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
“____” _____ 2025 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи контролю доступу та аудиту дій при роботі з мережевою базою даних”

Виконав здобувач вищої освіти
ІІ курсу, групи КІ-24М
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Кіблик І.О.
«____» _____ 2025 р.

Керівник проекту
кандидат фізико-математичних наук, доцент
_____ Петренюк В.І.
«____» _____ 2025 р.
Рецензент _____
_____

м. Кропивницький

АНОТАЦІЯ

Кіблик І.О. Дослідження та програмна реалізація системи контролю доступу та аудиту дій при роботі з мережевою базою даних. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2025.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи контролю доступу та аудиту дій при роботі з мережевою базою даних.

Метою розробки є дослідження та програмна реалізація системи контролю доступу та аудиту дій при роботі з мережевою базою даних.

Об'єктом дослідження є процес контролю доступу та аудиту дій при роботі з мережевою базою даних.

Предметом дослідження є методи контролю доступу та аудиту дій при роботі з мережевою базою даних.

Методи дослідження базуються на методах побудови баз даних та захисту інформації, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи контролю доступу та аудиту дій при роботі з мережевою базою даних.

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 Visual C#.

Ключові слова: комп'ютерна інженерія, доступ, аудит, база даних

ABSTRACT

Kiblyk I.O. Research and software implementation of the access control and audit system for actions when working with a network database. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2025.

In this final qualification work for the second (master's) level of higher education, software has been developed, which is intended for the access control and audit system for actions when working with a network database.

The purpose of the development is the research and software implementation of the access control and audit system for actions when working with a network database.

The object of the research is the process of access control and audit of actions when working with a network database.

The subject of the research is the methods of access control and audit of actions when working with a network database.

The research methods are based on methods of database construction and information protection, methods of mathematical statistics, methods of software development.

The result of the work is a software implementation of an access control system and audit of actions when working with a network database.

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 OS.

The program was developed in the Visual C# environment.

Keywords: computer engineering, access, audit, database

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

КСЗ	–	комплексна система захисту
ПЕОМ	–	персональна електронно-обчислювальна машина
СУБД	–	система управління базами даних
PGP	–	Pretty Good Privacy

ВСТУП

Актуальність теми. Контроль доступу до бази даних, або контроль доступу до БД – це метод, що дозволяє доступ до конфіденційної інформації компанії лише групам користувачів, яким дозволено доступ до таких даних, та обмежує доступ неавторизованим особам для запобігання витокам даних у системах баз даних. Контроль доступу до бази даних у СУБД включає два основні компоненти: автентифікацію та авторизацію.

Автентифікація – це спосіб підтвердження особи особи під час доступу до вашої бази даних. Важливо пам’ятати, що автентифікації користувача недостатньо для забезпечення безпеки даних. Авторизація, яка встановлює, чи є рівень доступу користувача або контроль доступу до даних відповідним, є додатковим рівнем захисту. Зрештою, без автентифікації та авторизації немає безпеки даних.

Кожна компанія сьогодні, у якої є співробітники, що взаємодіють з даними, а отже, кожна організація, повинна встановити контроль доступу до даних.

Після того, як ми розглянули питання «Що таке контроль доступу?», важливо зазначити, що ці засоби контролю впроваджуються для захисту ресурсів від несанкціонованого, незаконного доступу та забезпечення того, щоб суб’єкти могли отримувати доступ до об’єктів лише за допомогою безпечних, попередньо затверджених процедур.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи контролю доступу та аудиту дій при роботі з мережевою базою даних.

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

– Огляд існуючих систем контролю доступу та аудиту дій при роботі з мережевою базою даних.

– Дослідження системи контролю доступу та аудиту дій при роботі з мережевою базою даних.

– Програмна реалізація системи контролю доступу та аудиту дій при роботі з мережевою базою даних.

Об'єктом дослідження є процес контролю доступу та аудиту дій при роботі з мережевою базою даних.

Предметом дослідження є методи контролю доступу та аудиту дій при роботі з мережевою базою даних.

Методи дослідження базуються на методах побудови баз даних та захисту інформації, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод контролю доступу та аудиту дій при роботі з мережевою базою даних.

– Розроблено вітчизняний продукт контролю доступу та аудиту дій при роботі з мережевою базою даних, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі контролю доступу та аудиту дій при роботі з мережевою базою даних.

Достовірність наукових результатів підтверджена теоретичними викладеннями, даними комп'ютерного моделювання, коректними дослідженнями параметрів на функціонуючій обчислювальній мережі, а також відповідністю отриманих результатів окремим результатам, наведеним у науковій літературі.

Робота апробована на LVII Науково-технічній конференції здобувачів вищої освіти LV науково-технічної конференції «Наука в ЦНТУ: основні досягнення та перспективи розвитку» (2025 р.), основні положення випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти надруковані у статті збірника праць молодих науковців ЦНТУ, випуск №15.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи контролю доступу та аудиту дій при роботі з мережевою базою даних, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Фактори автентифікації:

– Пароль або PIN-код.

– Біометричне вимірювання (сканування відбитків пальців та сітківки).

– Картка або ключ .

Для комп'ютерної безпеки контроль доступу включає авторизацію, автентифікацію та аудит об'єкта, який намагається отримати доступ. Моделі контролю доступу мають суб'єкта та об'єкта.

Компоненти контролю доступу:

– Автентифікація: Автентифікація – це процес перевірки особи користувача. Автентифікація користувача – це процес перевірки особи користувача, коли цей користувач входить до комп’ютерної системи.

– Авторизація: Авторизація визначає ступінь доступу до мережі та типи послуг і ресурсів, доступних автентифікованому користувачеві. Авторизація – це метод забезпечення дотримання політик.

– Доступ: Після успішної автентифікації та авторизації їхня особа перевіряється, що дозволяє їм отримати доступ до ресурсу, до якого вони намагаються увійти.

– Керування: Організації можуть керувати своєю системою контролю доступу, додаючи та видаляючи автентифікацію та авторизацію для користувачів і систем. Керування цими системами може бути складним у сучасних IT-системах, які поєднують хмарні сервіси та фізичні системи.

– Аудит: Метод аудиту контролю доступу дозволяє організаціям дотримуватися цього принципу. Це дозволяє їм збирати дані про діяльність користувачів та аналізувати їх для виявлення можливих порушень доступу.

Контроль доступу передбачає визначення користувача на основі його облікових даних, а потім надання відповідного рівня доступу після підтвердження. Облікові дані використовуються для ідентифікації та автентифікації користувача, включаючи паролі, PIN-коди, токени безпеки та навіть біометричні сканування. Багатофакторна автентифікація (MFA) підвищує безпеку, вимагаючи від користувачів підтвердження за допомогою кількох методів. Після підтвердження особи користувача політики контролю доступу надають певні дозволи, дозволяючи користувачеві продовжувати роботу. Організації використовують кілька методів контролю доступу залежно від своїх потреб.

1.2 Область застосування

Залежно від вимог до відповідності та рівнів безпеки інформаційних технологій, які необхідно захистити, використовуються різні моделі контролю доступу. В основному контроль доступу буває двох типів:

– Контроль фізичного доступу: Фізичний контроль доступу обмежує вхід до кампусів, будівель, кімнат та фізичних IT-активів.

– Логічний контроль доступу: Логічний контроль доступу обмежує підключення до комп'ютерних мереж, системних файлів і даних.

Проблеми контролю доступу:

– Розподілені IT-системи: Сучасні IT-системи часто поєднують Інтернет та локальні мережі. Ці системи можуть бути розподілені географічно та складатися з різних пристроїв, активів та віртуальних машин. Доступ дозволено до всіх цих пристроїв, і відстеження їх може бути складним завданням.

– Управління політиками: Розробники політик в організації створюють політики, а IT-відділ перетворює заплановані політики на код для впровадження. Координація між цими двома групами є важливою для підтримки системи контролю доступу в актуальному стані та її належного функціонування.

– Моніторинг та звітність: Організації повинні постійно перевіряти системи контролю доступу, щоб гарантувати дотримання корпоративних політик та нормативних актів. Будь-які порушення або зміни повинні бути виявлені та негайно повідомлені.

– Моделі контролю доступу: Механізми контролю доступу забезпечують різні рівні точності. Вибір правильної стратегії контролю доступу для вашої організації дозволяє вам збалансувати прийнятну безпеку з ефективністю роботи співробітників.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи контролю доступу та аудиту дій при роботі з мережевою базою даних, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

Основні категорії користувачів

Користувачів СУБД можна розбити на три категорії:

– Адміністратор сервера баз даних. Він відає установкою, конфігуруванням сервера, реєстрацією користувачів, груп, ролей і т.п. Адміністратор сервера має ім'я ingres. Прямо або побічно він має всі привілеї, які мають або можуть мати інші користувачі.

– Адміністратори бази даних. До цієї категорії ставиться будь-який користувач, що створив базу даних, і, отже, що є її власником. Він може надавати іншим користувачам доступ до бази й до об'єктів, що втримуються в ній. Адміністратор бази відповідає за її збереження й відновлення. У принципі в організації може бути багато адміністраторів баз даних. Щоб користувач міг створити базу й стать її адміністратором, він повинен одержати (імовірно, від адміністратора сервера) привілей creatdb.

– Інші (кінцеві) користувачі. Вони оперують даними, що зберігаються в базах, у рамках виділених їм привілеїв.

У наступних розділах буде детально проаналізована система привілеїв СУБД INGRES. Тут ми відзначимо тільки, що адміністратор сервера баз даних, як самий привілейований користувач, має потребу в особливому захисту. Компрометація його пароля фактично означає компрометацію сервера й всіх баз, що зберігаються на ньому, даних.

Доручати адміністрування різних баз даних різним людям має сенс тільки тоді, коли ці бази незалежні й стосовно них не прийде проводити погоджену

політику виділення привілеїв або резервного копіювання. У такому випадку кожний з адміністраторів буде знати рівно стільки, скільки необхідно.

Можна провести аналогію між користувачем ingres і адміністраторами баз даних з одного боку, і суперкористувачем операційної системи (root у випадку ОС UNIX) і службовими користувачами (в ОС UNIX це можуть бути bin, lp, uucp і т.д.) з іншої сторони. Введення службових користувачів дозволяє адмініструвати функціональні підсистеми, не одержуючи привілеїв суперкористувача. Точно так само інформацію, що зберігається на сервері баз даних, можна розділити на відсіки, так що компрометація адміністратора одного відсіку не означає обов'язкової компрометації іншого.

Привілеї безпеки

Привілеї безпеки завжди виділяються конкретному користувачеві (а не групі, ролі або всім) під час його створення (оператором CREATE USER) або зміни характеристик (оператором ALTER USER). Таких привілеїв п'ять:

– Security – право управляти безпекою СУБД і відслідковувати дії користувачів. Користувач із цим привілеєм може підключатися до будь-якої бази даних, створювати, видаляти й змінювати характеристики користувачів, груп і ролей, передавати права на доступ до баз даним іншим користувачам, управляти записом реєстраційної інформації, відслідковувати запити інших користувачів і, нарешті, запускати INGRES-Команди від імені інших користувачів. Привілей security необхідний адміністраторові сервера баз даних, а також особі, персонально відповідальному за інформаційну безпеку. Передача цього привілеї іншим користувачам (наприклад, адміністраторам баз даних) збільшує число потенційно слабких місць у захисті сервера баз даних.

– Createdb – право на створення й видалення баз даних. Цим привілеєм, крім адміністратора сервера, повинні володіти користувачі, яким приділяється роль адміністраторів окремих баз даних.

– Operator – право на виконання дій, які традиційно відносять до компетенції оператора. Маються на увазі запуск і зупинка сервера, збереження й

відновлення інформації. Крім адміністраторів сервера й баз даних цим привілеєм доцільно наділити також адміністратора операційної системи.

– Maintain_locations – право на керування розташуванням баз адміністратори сервера баз даних і операційної системи.

– Trace – право на зміну стану прапорів відлагоджувального трасування. Даний привілей корисний адміністраторові сервера баз даних і інших знаючих користувачів при аналізі складних, незрозумілих ситуацій.

Привілеї доступу

Привілеї доступу виділяються користувачам, групам, ролям або всім за допомогою оператора GRANT і вилучаються за допомогою оператора REVOKE. Ці привілеї, як правило, привласнює власник відповідних об'єктів (він же – адміністратор бази даних) або власник привілеї security (звичайно адміністратор сервера баз даних).

Перш ніж привласнювати привілеї групам і ролям, їх (групи й ролі) необхідно створити за допомогою операторів CREATE GROUP і CREATE ROLE.

Для зміни состава групи служить оператор ALTER GROUP.

Оператор DROP GROUP дозволяє видаляти групи, щоправда, тільки після того, як спустошений список членів групи.

Оператор ALTER ROLE служить для зміни паролів ролей, а DROP ROLE – для видалення ролей.

Нагадаємо, що створювати й видаляти іменовані носії привілеїв, а також змінювати їхні характеристики може лише користувач із привілеєм security. При здійсненні подібних дій необхідно мати підключення до бази даних iidbdb, у якій зберігаються відомості про суб'єктів і їхні привілеї.

Привілеї доступу можна підрозділити відповідно до видів об'єктів, до яких вони ставляться. У СУБД INGRES таких видів п'ять:

– Таблиці й подання.

– Процедури.

– Бази даних.

– Сервер баз даних.

– Події.

Присвоювання привілеїв доступу виробляється за допомогою оператора GRANT.

Стосовно до таблиць і подань можна управляти наступними правами доступу:

– SELECT – право на вибірку даних

– INSERT – право на додавання даних

– DELETE – право на видалення даних

– UPDATE – право на відновлення даних (можна вказати певні стовпці, дозволені для відновлення)

– REFERENCES – право на використання зовнішніх ключів, що посилаються на дану таблицю (можна вказати певні стовпці)

За замовчуванням користувач не має ніяких прав доступу до таблиць і подань – їх необхідно передати за допомогою операторів GRANT.

Стосовно процедури можна надати право на виконання. При цьому не потрібно піклуватися про виділення прав доступу до об'єктів, оброблюваним процедурою – їхня наявність не обов'язково. Таким чином, процедури баз даних є зручним засобом надання контрольованого доступу для виконання строго певних дій над даними.

Права доступу до бази даних як до єдиного цілого може надавати її адміністратор або користувач із привілеєм security. Ці "права" насправді встановлюють ряд обмежень на використання бази даних, тобто по суті є заборонними. Мається на увазі обмеження на число операцій уведення/виводу або число рядків, що повертаються одним запитом, обмеження права створення таблиць і процедур і т.п. За замовчуванням користувач не соромиться кількісними лімітами й одержує право на створення об'єктів у базі.

Відзначимо, що при створенні бази даних вказується її статус – загальна або особиста. Це впливає на які мається на увазі права доступу до бази. За

замовчуванням право на підключення до загальної бази надається всім. Право на підключення до особистої бази потрібно передавати явно. Право на підключення необхідно для виконання всіх інших операцій з базою й об'єктами, що втримувалися в ній.

Привілеї (які в цьому випадку точніше було б назвати обмеженнями) QUERY_IO_LIMIT і QUERY_ROW_LIMIT перевіряються на підставі оцінок, виданих оптимізатором запитів. Якщо оптимізатор пророкує, що запит перевищить відведений ліміт числа операцій уведення виводу або рядків, що повертаються, він (запит) відкидається. Накладення подібних кількісних обмежень перешкоджає монополізації сервера одним клієнтом і може використовуватися як один з інструментів підтримки високої готовності.

Оборотний увага на наступну цікаву обставину. За замовчуванням всі користувачі мають право створювати процедури в базах даних. Якби вони при цьому автоматично діставали права на виконання, то змогли б здійснити по суті будь-яку операцію з даними, оскільки виконання процедури не вимагає прав доступу до оброблюваних об'єктів. На щастя, для передачі привілеї доступу до об'єктів, і в, частковості, для надання права на виконання процедури, треба бути не тільки власником об'єкта, але й адміністратором бази даних. Ми бачимо, наскільки обережно потрібно ставитися до надання привілеїв виконання стосовно нових, неперевірених процедур. У принципі досить одного "троянського коня", щоб скомпрометувати всю базу даних. Процедури є настільки ж гнучким, але й настільки ж небезпечним засобом, що й UNIX-Програми з бітом переустановки діючого ідентифікатора користувача.

Відзначимо, що заборонні привілеї в принципі можна накласти на адміністратора бази даних або адміністратора сервера, однак вони не будуть мати сили. Тим самим зловмисник не має змоги обмежити права доступу адміністраторів. Зокрема, він не може створити особисту "секретну" базу даних, до якої не буде мати доступ користувач ingres. Правда, у подібного положення є й

зворотний бік – компрометація пароля адміністратор сервера надає зловмисникові необмежені права доступу до всіх баз даних.

Права доступу до сервера поширюються на всі бази даних, що обслуговуються даним серверам. Набір цих прав той же, що й для окремих баз даних.

Привілеї, явно певні для окремих баз, мають пріоритет над привілеями сервера.

Механізм подій докладно розглянутий в [1]. Тут ми відзначимо лише, що стосовно подій є два привілеї – RAISE і REGISTER. Перша дозволяє збуджувати події, друга – реєструватися для їхнього одержання.

Оператор GRANT може містити необов'язкову частину, принципово важливу для захисту СУБД.

Подібний оператор GRANT передає не тільки зазначені в ньому привілеї, але й права на їхню подальшу передачу. Очевидно, що використання конструкції WITH GRANT OPTION веде до децентралізації контролю над привілеями й містить потенційну погрозу безпеки даних.

Для скасування привілеїв, виданих раніше (як дозвільних, так і заборонних), служить оператор REVOKE.

Одержання інформації про привілеї

Важливо не тільки давати й відбирати привілеї, але й мати інформацію про те, якими правами доступу володіє кожний із суб'єктів. Подібні дані можна одержати за допомогою функції dbmsinfo, а також шляхом аналізу вмісту таблиць у базі даних iidbdb.

Функція dbmsinfo повертає права доступу до бази, що ставляться до поточного підключення. Можна довідатися імена діючі групи й ролі, значення кількісних обмежень, наявність привілеїв для створення таблиць і процедур і т.п.

Таблиці iiusergroup, iirole і iidbprivileges бази даних iidbdb містять відповідно, список груп і їхній состав, перелік ролей разом із зашифрованими

паролями й відомості про привілеї доступу до баз даних. Так, таблиця iiusergroup складається із трьох стовпців:

– Groupid – ім'я групи,

– Groupmem – ім'я члена групи,

– Reserve – резервний стовпець.

Засобами SQL із цих таблиць можна витягти необхідну інформацію.

Використання подань для керування доступом

СУБД надають специфічний засіб керування доступом – подання. Подання дозволяють зробити видимими для суб'єктів певні стовпці базових таблиць (реалізувати проекцію) або відібрати певні рядки (реалізувати селекцію). Не надаючи суб'єктам прав доступу до базових таблиць і сконструювавши підходящі подання, адміністратор бази даних захистить таблиці від несанкціонованого доступу й постачить кожного користувача своїм баченням бази даних, коли недоступні об'єкти як би не існують.

Ієрархія прав доступу

Оператор GRANT і інші засоби керування доступом СУБД дозволяють реалізувати наступні види обмеження доступу:

– Операційні обмеження (за рахунок прав доступу SELECT, INSERT, UPDATE, DELETE, застосовних до всім або тільки деякими стовпцями таблиці).

– Обмеження за значеннями (за рахунок механізму подань).

– Обмеження на ресурси (за рахунок привілеїв доступу до баз даних).

При обробці запиту СУБД спочатку перевіряє права доступу до об'єктів. Якщо операційні обмеження виявляються порушеними, запит відкидається з видачею відповідної діагностики. Порушення обмежень на значення впливає тільки на кількість результуючих рядків; ніякої діагностики при цьому не видається (див. попередній пункт). Нарешті, після обліку двох попередніх обмежень, запит надходить на обробку оптимізаторові. Якщо той виявить перевищення обмежень на ресурси, запит буде відкинутий з видачею відповідної діагностики.

На ієрархію привілеїв можна подивитися й з іншого погляду. Кожний користувач, крім, власних, має привілеї PUBLIC. Крім цього, він може входити в різні групи й запускати додатка з певними ролями. Як співвідносяться між собою права, надані різним іменованим носіям привілеїв?

Ієрархія авторизації виглядає для СУБД INGRES у такий спосіб:

– Роль (вищий пріоритет).

– Користувач.

– Група.

– PUBLIC (нижчий пріоритет).

Для кожного об'єкта, до якого здійснюється доступ, INGRES намагається відшукати в ієрархії привілей, що ставиться до запитуваного виду доступу (SELECT, EXECUTE і т.п.). Наприклад, при спробі доступу до таблиці з метою відновлення, INGRES перевіряє привілеї ролі, користувача, групи й всіх користувачів. Якщо хоча б на одному рівні ієрархії привілей UPDATE є, запит передається для подальшої обробки. У противному випадку використовується яке мається на увазі право доступу, що пропонує відкинути запит.

Розглянемо докладніше трактування обмежень на ресурси. Нехай, наприклад, на всіх чотирьох рівнях ієрархії специфіковані свої обмеження на число результуючих рядків запиту (привілей QUERY_ROW_LIMIT).

Якщо користувач у момент початку сеансу роботи із СУБД задав і роль, і групу, буде використане обмеження, що накладається роллю (1700). Якби привілей QUERY_ROW_LIMIT для ролі була відсутня, або користувач не задав роль на початку сеансу роботи, користувач зміг би одержувати результати не більш ніж з 1500 рядків і т.п. Якби привілей QUERY_ROW_LIMIT взагалі не була специфікована на жодному рівні ієрархії, СУБД скористалася б яким мається на увазі значенням, що у цьому випадку означає відсутність обмежень на число результуючих рядків.

Звичайно використовувана роль і група задаються, відповідно, як аргументи опцій -R і -G у командному рядку запуску додатка.

Якщо опція -G відсутній, застосовується яка мається на увазі група користувача, якщо така є.

Мітки безпеки й примусовий контроль доступу

В "Критеріях оцінки надійних комп'ютерних систем", стосовно до систем рівня безпеки B, описаний механізм міток безпеки, реалізований у версії INGRES/Enhanced Security (INGRES з підвищеною безпекою). Застосовувати цю версію на практиці має сенс тільки в сполученні з операційною системою й іншими програмними компонентами того ж рівня безпеки. Проте, розгляд реалізації міточної безпеки в СУБД INGRES цікаво з пізнавальної точки зору, а сам підхід, заснований на поділі даних по рівнях таємності й категоріям доступу, може виявитися корисним при проектуванні системи привілеїв численних користувачів стосовно більших масивів даних.

У СУБД INGRES/Enhanced Security до кожної реляційної таблиці неявно додається стовпець, що містить влучні безпеки рядків таблиці. Мітка безпеки складається із трьох компонентів:

– Рівень таємності. Зміст цього компонента залежить від додатка. Зокрема, можливий традиційний спектр рівнів від "абсолютно секретно" до "несекретно".

– Категорії. Поняття категорії дозволяє розділити дані на "відсіки" і тим самим підвищити надійність системи безпеки. У комерційних додатках категоріями можуть служити "фінанси", "кадри", "матеріальні цінності" і т.п. Нижче призначення категорій роз'ясняється більш докладно.

– Області. Є додатковим засобом розподілу інформації на відсіки. На практиці компонентів "область" може дійсно мати географічний сенс, позначаючи, наприклад, країну, до якої ставляться дані.

Кожний користувач СУБД INGRES/Enhanced Security характеризується ступенем благонадійності, що також визначається влучної безпеки, привласненої даному користувачеві. Користувач може одержати доступ до даних, якщо ступінь

його благонадійності задовольняє вимогам відповідної мітки безпеки. Більш точно:

– Рівень таємності користувача повинен бути не нижче рівня таємності даних.

– Набір категорій, заданих у мітці безпеки даних, повинен цілком утримуватися в мітці безпеки користувача.

– Набір областей, заданих у мітці безпеки користувача, повинен цілком утримуватися в мітці безпеки даних.

Розглянемо приклад. Нехай дані мають рівень таємності "конфіденційно", належать категорії "фінанси" і ставляться до областей "Україна" і "ЄС". Далі, нехай ступінь благонадійності користувача характеризується влучної безпеки з рівнем таємності "абсолютно секретно", категоріями "фінанси" і "кадри", а також областю "Україна". Такий користувач одержить доступ до даних. Якби, однак, у мітці користувача була зазначена тільки категорії "кадри", у доступі до даним нього було б відмовлене, незважаючи на його "зовсім секретний" рівень.

Спеціальний привілей, DOWNGRADE, дозволяє змінювати мітки безпеки, асоційовані з даними. Подібна можливість необхідна, наприклад, для корекції міток, по тимі або інших причинах які виявилися неправильними.

Представляється природним, що СУБД INGRES/Enhanced Security допускає не тільки сховане, але і явне включення міток безпеки в реляційні таблиці. З'явився новий тип даних, security label, що підтримує відповідні операції порівняння.

INGRES/Enhanced Security – перша СУБД, що одержала сертифікат, еквівалентний атестації на клас безпеки B Імовірно, мітки безпеки поступово ввійдуть у стандартний репертуар систем керування базами даних.

Підтримка цілісності даних у СУБД

Для комерційних організацій забезпечення цілісності даних принаймні не менш важливо, чим забезпечення конфіденційності. Звичайно, неприємно, коли хтось підглядає за сумами на рахунках клієнтів, але набагато гірше, коли в

процесі перекладу грошей з рахунку на рахунок частина суми зникає в невідомому напрямку.

Відомо, що головними ворогами баз даних є не зовнішні зловмисники, а помилки встаткування, адміністраторів, прикладних програм і користувачів. Захист від подібних помилок – головна тема цього розділу.

З погляду користувача СУБД, основними засобами підтримки цілісності даних є обмеження й правила.

Обмеження

Обмеження можуть ставитися до таблиць або окремих стовпців. Обмеження на стовпці задаються при створенні таблиці, в операторах CREATE TABLE

Табличні обмеження ставляться до групи стовпців і можуть задаватися як при створенні таблиці, так і пізніше, за допомогою оператора ALTER TABLE.

Посилальні обмеження відповідають за цілісність зв'язків між таблицями. Подібне обмеження вимагає, щоб кожному значенню в стовпці або групі стовпців однієї таблиці відповідало рівно одне значення в іншій таблиці. Назва обмеження пояснюється тим, що такі значення відіграють роль посилань між таблицями в реляційній моделі.

Обмеження всіх видів накладаються власником таблиці й впливають на результат наступних операцій з даними. Перед завершенням виконання SQL-Оператора виробляється перевірка наявних обмежень. При виявленні порушень СУБД сигналізує про ненормальне завершення й анулює внесені оператором зміни.

Відзначимо, що для накладення посилального обмеження необхідно мати привілей REFERENCES стосовно таблиці, на яку робиться посилання (dept у прикладі вище).

Обмеження можна не тільки накладати, але й скасовувати. При цьому між обмеженнями можуть існувати залежності, і скасування одного з них може зажадати ліквідації інших (посилальних) обмежень, що залежать від первісного.

Слово cascade означає, що варто видалити також всі обмеження, прямо або побічно залежні від dept_unique. У цьому випадку буде вилучене обмеження empref. Якщо замість cascade указати restrict, тобто зробити спробу видалити тільки обмеження dept_unique, СУБД зафіксує помилку. Тим самим забезпечується цілісність системи обмежень.

У СУБД INGRES робиться спроба примирити контроль обмежень і ефективність функціонування. При масовому копіюванні даних контроль обмежень відключається. Це значить, що необхідно доповнювати копіювання запуском процедури глобальної перевірки цілісності.

Кластерна організація сервера баз даних

Ми будемо розуміти під кластером конфігурацію з декількох комп'ютерів (вузлів), що виконують загальний додаток (таке, наприклад, як сервер баз даних). Звичайно кластер містить також кілька дискових підсистем, спільно використовуваних вузлами-комп'ютерами, і надлишкові зв'язки між компонентами. Із зовнішньої точки зору кластер виглядає як єдине ціле, а наявність декількох вузлів сприяє підвищенню продуктивності й стійкості до відмов.

У справжньому розділі буде розглянута розробка компанії Sun Microsystems, Inc – SPARCcluster PDB Server (паралельний сервер баз даних на основі SPARC-Кластера).

Апаратна організація SPARCcluster PDB Server

У мінімальній конфігурації SPARCcluster PDB Server складається із двох вузлів SPARCserver 1000, двох дискових підсистем SPARCstorage Array і консолі кластера (SPARCclassic). Вузли- комп'ютери з'єднуються між собою за допомогою швидкого Ethernet (100 Мбіт/с), дискові підсистеми підключаються через оптоволоконі канали. У могутнішій конфігурації замість SPARCserver 1000 може використовуватися SPARCcenter 2000, а число дискових підсистем здатно досягати 32 (до 1 Тб дискового простору). Кожний вузол кластера – це багатопроцесорний комп'ютер, до якого, крім інших, підключені накопичувачі на

DAT-стрічках (або автозавантажувач касет з такими стрічками). Всі зв'язки з комп'ютерами й дисковими підсистемами продубльовані.

Подібна апаратна архітектура забезпечує стійкість до відмов (ніяка одиночна відмова не викликає зупинки роботи кластера в цілому). У той же час надлишкові компоненти (комп'ютери, дискові підсистеми) аж ніяк не обмежуються роллю гарячого резерву – вони повністю задіяні в процесі звичайної роботи.

Вся апаратура влаштована так, що допускає заміну в гарячому режимі, без зупинки інших компонентів кластера.

Програмна організація SPARCcluster PDB Server

Якщо розглядати програмну організацію SPARCcluster PDB Server у контексті надійної роботи баз даних, необхідно звернути увагу ще на один компонент – фронтальну машину, на якій виконується який-небудь монітор транзакцій, наприклад, TUXEDO. З урахуванням цього доповнення програмна організація здобуває наступний вид.

Розглянемо компоненти програмного забезпечення SPARCcluster PDB Server.

Стійкий до відмов розподілений менеджер блокувань (Fault Tolerant Distributed Lock Manager, FT-DLM) управляє паралельним доступом до баз даних, установлюючи й знімаючи блокування. Крім того, FT-DLM нейтралізує наслідку відмов, знімаючи блокування, установлені вузлом, що вийшли з ладу. FT-DLM взаємодіє із сервером Oracle для підтримки неблокуємих операцій читання й для блокування на рівні рядків при записі в таблиці. У результаті забезпечується цілісність і серіализація транзакцій у сполученні з паралельною роботою вузлів кластера й з паралельним доступом до декількох дискових підсистем.

Розподіленість менеджера блокувань означає, що на кожному вузлі кластера працює свій екземпляр FT-DLM і що FT-DLM уміє динамічно реконфігурувати себе (як при виході вузлів з ладу, так і при додаванні нових

вузлів). У результаті вихід з ладу одного вузла не означає краху всього сервера баз даних – сервер живий, поки працює хоча б один менеджер блокувань.

У розглянутому контексті основне призначення розподіленого менеджера томів – підтримка дзеркалювання дисків з тим важливим доповненням, що пристрою, що становлять пари, можуть належати різним дисковим підсистемам.

Підсистема виявлення й нейтралізації відмов постійно відслідковує доступність ресурсів, що становлять кластер. При виявленні несправності запускається процес реконфігурації, що ізолює, що вийшов з ладу компонентів при збереженні працездатності кластера в цілому (з виходом з ладу диска справляється менеджер томів).

Підсистема керування кластером складається із трьох інструментів із графічним інтерфейсом: консолі кластера, менеджера томів і менеджера сервера Oracle. Їхня інтеграція забезпечує централізоване оперативне керування всіма ресурсами кластера.

Нейтралізація відмови вузла

Розглянемо, як в SPARCcluster PDB Server реалізована нейтралізація самого неприємного з відмов – відмови вузла. Програмне забезпечення вживає при цьому наступні дії:

– Підсистема виявлення відмов виявляє вузол, що вийшов з ладу.

– Створюється нова конфігурація кластера, без вузла, що відмовив. Цей процес займає 1 – 2 хвилини, протягом яких обробка транзакцій припиняється.

Менеджер блокувань робить відновлення:

– Підтверджені транзакції від вузла, що відмовив (транзакції, про успішне завершення яких інші вузли кластера не встигли довідатися) накочуються вперед і деблокуються. а

– Непідтверджені транзакції від вузла, що відмовив, відкочуються й також деблокуються.

У цей період транзакції обробляються справними вузлами, але, імовірно, трохи повільніше, ніж звичайно.

– Монітор транзакцій повторно направляє в кластер непідтверджені транзакції.

– Вузол, що вийшов з ладу, ремонтується й знову запускається.

– Створюється нова конфігурація кластера, що включає в себе відремонтований вузол.

Відзначимо, що всі дії, крім ремонту компонента, що відмовив, виконуються в автоматичному режимі й не вимагають втручання обслуговуючого персоналу. Варто враховувати, однак, що якщо наступна поломка трапиться до закінчення ремонту, кластер може на якийсь час стати непрацездатним, тому затягувати ремонт не рекомендується.

Строго говорячи, SPARCcluster PDB Server не підтримує одну з важливих кластерних функцій – із зовнішньої точки зору кластер не виглядає як єдине ціле. Прикладні програми можуть прямо підключатися до його вузлів, і тоді відмови вузлів вимагають нейтралізації на прикладному рівні. У той же час використання моніторів транзакцій дозволяє згладити цей недолік, забезпечуючи до того ж балансування завантаження між вузлами.

Тиражування даних

У контексті інформаційної безпеки тиражування можна розглядати як засіб підвищення доступності даних. Стала легендою історія про бакалійника із Сан-Франциско, що після руйнівного землетрусу відновив свою базу даних за 16 мінут, перекачавши з іншого міста попередньо протиражовану інформацію.

Розвинені можливості тиражування надає СУБД INGRES. Їм присвячена стаття [2]. Тут ми розглянемо можливості іншого популярного сервера СУБД – Informix OnLine Dynamic Server (OnLine-DS) На відміну від попереднього розділу, мова йтиме про звичайні (а не кластерних) конфігураціях.

В Informix OnLine-DS 1 підтримується модель тиражування, що складає в повнім відображенні даних з основного сервера на вторинні.

У конфігурації серверів Informix OnLine-DS з тиражуванням виділяється один основний і ряд вторинних серверів. На основному сервері виконується й

читання, і відновлення даних, а всі зміни передаються на вторинні сервери, доступні тільки на читання. У випадку відмови основного сервера вторинний автоматично або вручну переводиться в режим доступу на читання й запис. Прозорий перенапрямок клієнтів при відмові основного сервера не підтримується, але воно може бути реалізоване в рамках додатків.

Після відновлення основного сервера можливий сценарій, при якому цей сервер стає вторинним, а колишній вторинному, котрий уже функціонує в режимі читання-запису, надається статус основного; клієнти, які підключені до нього, продовжують роботу. Таким чином, забезпечується безперервна доступність даних.

Тиражування здійснюється шляхом передачі інформації з журналу транзакцій (логічного журналу) у буфер тиражування основного сервера, звідки вона пересилається в буфер тиражування вторинного сервера. Таке пересилання може відбуватися або в синхронному, або в асинхронному режимі. Синхронний режим гарантує повну погодженість баз даних – жодна транзакція, зафіксована на основному сервері, не залишиться незафіксованої на вторинному, навіть у випадку збоїти основного сервера. Асинхронний режим не забезпечує абсолютної погодженості, але поліпшує робочі характеристики системи.

Побічний позитивний ефект тиражування – можливість винести переважно на вторинний сервер ресурсномісткі додатки підтримки прийняття рішень. У цьому випадку вони можуть виконуватися з максимальним використанням засобів паралельної обробки, не придушуючи додатків оперативної обробки транзакцій, зосереджених на основному сервері. Це також можна розглядати як фактор підвищення доступності даних.

Погрози, специфічні для СУБД

Головне джерело погроз, специфічних для СУБД, лежить у самій природі баз даних. Основним засобом взаємодії із СУБД є мова SQL – потужний непроцедурний інструмент визначення й маніпулювання даними. Збережені процедури додають до цього репертуару керуючі конструкції. Механізм правил

дає можливість вишиковувати складні, важкі для аналізу ланцюжка дій, дозволяючи попутно неявним образом передавати право на виконання процедур, навіть не маючи, строго говорячи, повноважень на це. У результаті потенційний зловмисник одержує у свої руки потужний і зручний інструментарій, а весь розвиток СУБД спрямоване на те, щоб зробити цей інструментарій ще могутніше й зручніше.

Ми розглянемо кілька погроз, що виникають при використанні зловмисником засобів мови SQL.

Одержання інформації шляхом логічних виводів

Приведемо приклад – з'ясування набору первинних ключів таблиці при наявності тільки привілеї INSERT (без привілеї SELECT). Якщо набір можливих значень ключів приблизно відомий, можна намагатися вставляти нові рядки з "цікавими" ключами й аналізувати коди завершення SQL-Операторів. Як ми бачили з попереднього приклада, сам факт присутності певного ключа в таблиці може бути досить інформативним.

Якщо для реалізації контролю доступу використовуються подання, і ці подання допускають модифікацію, за допомогою операцій модифікації/вставки можна одержати інформацію про вміст базових таблиць, не розташовуючи прямим доступом до них.

Основним засобом боротьби з подібними погрозами, крім ретельно проектування моделі даних, є механізм розмноження рядків. Суть його в тім, що до складу первинного ключа, явно або неявно, включається мітка безпеки, за рахунок чого з'являється можливість зберігати в таблиці кілька екземплярів рядків з однаковими значеннями "змістовних" ключових полів. Найбільш природне розмноження рядків реалізується в СУБД, що підтримують влучні безпеки (наприклад, в INGRES/Enhanced Security), однак і стандартними SQL-засобами можна одержати задовільне рішення.

Продовжуючи медичну тематику, розглянемо базу даних, що складає з однієї таблиці із двома стовпцями: ім'я пацієнта й діагноз. Передбачається, що

ім'я є первинним ключем. Кожна з рядків таблиці ставиться до одному із двох рівнів таємності – високому (HIGH) і низькому (LOW). Відповідно, і користувачі підрозділяються на два рівні благонадійності, які ми також будемо називати високим і низьким.

До високого рівня таємності ставляться відомості про пацієнтів, що перебувають під наглядом правоохоронних органів або страждаючих специфічних захворювань. На низькому рівні розташовуються дані про інших пацієнтів, а також інформація про деяких "секретних" пацієнтів з "маскувальним" діагнозом.

Звернемо увагу на те, що відомості про пацієнта на прізвище Іванов присутні на обох рівнях, але містять різні діагнози.

Ми хочемо реалізувати таку дисципліну доступу, щоб користувачі з низьким рівнем благонадійності могли маніпулювати тільки даними на своєму рівні й не мали можливості зробити які-небудь виводи про присутність у секретній половині відомостей про конкретних пацієнтів. Користувачі з високим рівнем благонадійності повинні мати доступ до секретної половини таблиці, а також до інформації про інших пацієнтів.

Ми бачимо, що на відміну від систем з міточною безпекою, стандартні SQL-сервери надають досить великовагові засоби для реалізації механізму розмноження рядків. Проте, ці засоби не так погані, як може здатися на перший погляд. Можна сподіватися, що оптимізатор SQL-запитів, що входить у комплект будь-який сучасної СУБД, зробить час доступу до подання PatientInfo порівнянним із часом добування рядків з базових таблиць.

Неважко зрозуміти, що боротьба з одержанням інформації шляхом логічного виводу актуальна не тільки для медичних баз даних і що вона (боротьба) вимагає кропіткої праці при проектуванні моделі даних і ієрархії привілеїв, а також при реалізації видимих користувачам подань.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Програмне забезпечення написане мовою Visual C#. Ця мова обрана виходячи з наступних міркувань. Visual C# – строго типізована об'єктно-орієнтована мова, призначена для розробки різноманітних безпечних і потужних додатків, виконуваних у середовищі .NET Framework. Мовою Visual C# можна розробляти звичайні клієнтські додатки Windows, веб-служби XML, розподілені компоненти, додатки типу " сервер-клієнт", додатки баз даних і багато яких інших. В Visual C# є розширений редактор коду, конструктори зі зручним користувальницьким інтерфейсом, вбудований відладник і багато інших засобів, покликані спростити розробку додатків мовою Visual C# версії 5.0 і .NET Framework версії 4.5.

Синтаксис Visual C# дуже виразний, але простий у вивченні. Усі, хто знаком з мовами C, C++ або Java з легкістю взнають синтаксис із фігурними дужками, характерний для мови Visual C#. Розроблювачі, що знають кожну із цих мов, як правило, зможуть домогтися ефективної роботи з мовою Visual C# за дуже короткий час. Синтаксис Visual C# робить простіше те, що було складно в C++, і забезпечує потужні можливості, такі як типи значень Nullable, перерахування, делегати, лямбда-вираження й прямий доступ до пам'яті, чого немає в Java. Visual C# підтримує універсальні методи й типи, забезпечуючи більше високий рівень безпеки й продуктивності, а також ітератори, що дозволяють при реалізації колекцій класів визначати власне поводження ітерації, що може легко використовуватися в клієнтському коді. В Visual C# 5.0 вираження LINQ (Language-Integrated Query) роблять строго-типізований запит першокласною конструкцією мови.

Як об'єктно-орієнтована мова, Visual C# підтримує поняття інкапсуляції, спадкування й поліморфізму. Всі змінні й методи, включаючи метод `Main` – крапку входу додатка – інкапсулюється у визначення класів. Клас може

успадковувати безпосередньо з одного родового класу, але може реалізовувати будь-яке число інтерфейсів. Для методів, які перевизначають віртуальні методи в батьківському класі, необхідно ключове слово override, щоб виключити випадкове повторне визначення. У мові Visual C# структура схожа на полегшений клас: це тип, що розподіляється по стопках, що реалізує інтерфейси, але не підтримуюче спадкування.

На додаток до основних описаних об’єктно-орієнтованих принципів, мова Visual C# спрощує розробку компонентів програмного забезпечення завдяки декільком інноваційним конструкціям мови, у число яких входять наступні:

– Інкапсульовані підписи методів, називані делегатами, які підтримують строго-типізовані повідомлення про події.

– Властивості, що виступають у ролі методів доступу для закритих змінних-членів.

– Атрибути з декларативними метаданими про типи під час виконання.

– Вбудовані коментарі XML-документації.

– LINQ (Language-Integrated Query), що пропонує вбудовані можливості запитів у різних джерелах даних.

Якщо буде потрібно забезпечити взаємодію з іншим програмним забезпеченням Windows, таким як об'єкти COM або власні бібліотеки DLL Win32, у мові Visual C# можна використовувати процес, що називається "Interop". Процес Interop дозволяє програмам на Visual C# виконувати практично будь-які дії, які може виконувати вихідний додаток на C++. Мова Visual C# підтримує навіть покажчики й поняття "небезпечного" коду для тих випадків, коли прямий доступ до пам'яті має вкрай важливе значення.

Процес побудови Visual C# у порівнянні з C і C++ простий і є більше гнучким, чим в Java. Немає окремих файлів заголовка, а методи й типи не потрібно повідомляти в певному порядку. У вихідному файлі Visual C# може бути визначене будь-яке число класів, структур, інтерфейсів і подій.

Архітектура платформи .NET Framework

Програма мовою Visual C# виконується в середовищі .NET Framework – інтегрованому компоненті Windows, що містить віртуальну систему виконання (середовище CLR) і уніфікований набір бібліотек класів. Середовище CLR являє собою комерційну реалізацію корпорацією Майкрософт інфраструктури CLI, що є міжнародним стандартом, який лежить в основі створення середовищ виконання й розробки, у яких забезпечується тісна взаємодія між мовами й бібліотеками.

Вихідний код, написаний мовою Visual C#, компілюється в проміжну мову (IL) у відповідності зі специфікацією CLI. Код IL і ресурси, такі як растрові зображення й рядки, зберігаються на диску у файлі, що виконується, називаному складанням, з розширенням EXE або DLL у більшості випадків. Складання містить маніфест із відомостями про типи складання, версії, мови й регіональні параметри та вимоги безпеки.

При виконанні програми на Visual C# складання завантажується в середовище CLR залежно від відомостей у маніфесті. Далі, якщо вимоги безпеки дотримані, середовище CLR виконує JIT-компіляцію для перетворення коду IL в інструкції машинного коду. Середовище CLR також надає інші служби, що відносяться до автоматичного збору сміття, обробки виключень і керуванню ресурсами. Код, виконуваний середовищем CLR, іноді називають "керованим кодом" у протиставлення "некерованому коду", що компілюється в машинний код, призначений для певної системи. Далі показані відносини під час компіляції й час виконання між файлами з вихідним кодом Visual C#, бібліотеками класів .NET Framework, складаннями й середовищем CLR.

Взаємодія між мовами є ключовою особливістю .NET Framework. Оскільки код IL, створюваний компілятором Visual C# відповідає специфікації CTS, код IL на основі Visual C# може взаємодіяти з кодом, створюваним версіями мов Visual Basic, Visual C++, Visual J# платформи .NET Framework і ще більш ніж 20 CTS-сумісних мов. В одному складанні може бути кілька модулів, написаних

на різних мовах платформи .NET Framework, і типи можуть посилатися один на одного, як якби вони були написані на одній мові.

Крім служб часу виконання, в.NET Framework також є велика бібліотека, що складається з більш ніж 4000 класів, організованих по просторах імен, які забезпечують різноманітні корисні функції для будь-яких дій, починаючи від введення й виведення файлів для керуванням рядками для розбивки XML, і закінчуючи елементами керування Windows Forms. У звичайному додатку мовою Visual C# бібліотека класів .NET Framework інтенсивно використовується для "устрою" коду.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи контролю доступу та аудиту дій при роботі з мережевою базою даних.

В процесі розробки випускної кваліфікаційної роботи за другим (магістерським) 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) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Найвідоміші приклади контролю доступу до бази даних включають:

– Дискреційний контроль доступу (DAC). Власник даних надає доступ до моделей DAC. DAC – це метод призначення прав доступу на основі правил, визначених користувачем.

– Обов'язковий контроль доступу (MAC). У MAC людям дозволено доступ на основі дозволу на доступ до інформації, розробленого з використанням недискреційної парадигми. MAC позначає політику, яка призначає дозволи доступу на основі правил центрального органу влади.

– Контроль доступу на основі ролей (RBAC). RBAC використовує фундаментальні принципи безпеки, такі як «мінімальні привілеї» та «розділення привілеїв», щоб надавати доступ залежно від ролі користувача. Як результат, той, хто хоче отримати доступ до інформації, може отримати доступ лише до даних, необхідних для виконання його функції.

– Контроль доступу на основі атрибутів (ABAC). Кожен ресурс і користувач в ABAC отримує набір атрибутів. Цей динамічний підхід визначає доступ до ресурсів на основі порівняння характеристик користувача, таких як час доби, місцезнаходження та місцезнаходження.

Як працюють системи контролю доступу до баз даних

Системи контролю доступу до баз даних працюють з трьох сторін: користувача, адміністратора та інфраструктури:

– Користувач: Коли співробітник бажає увійти до зони обмеженого доступу, він повинен надати свої облікові дані. Запит на розблокування подається на зчитувач карток, який надсилає інформацію до блоку контролю доступу, що згодом авторизує користувача та відкриває двері.

– Адміністратор: Система контролю доступу має панель керування або портал на адміністративній стороні. Адміністратори офісу, IT-менеджери та керівники служби безпеки можуть використовувати портал керування, щоб вказати, хто має доступ до приміщення та за яких умов.

– Системна інфраструктура: Інфраструктура системи контролю доступу включає електричні замки, зчитувачі карт, стан дверей для моніторингу руху та пристрої запитів на вихід, всі з яких повідомляють на панель керування та сервер.

Ось деякі з найкращих практик використання системи контролю доступу до бази даних:

– Зосередьтеся на доступі до конфіденційних даних. Однією з реалій управління великою компанією є збір величезних обсягів конфіденційних даних, які потім зберігаються та обробляються в базах даних. Як наслідок, бази даних є поширеною мішенню для кібератак. Як наслідок, критично важливо зосередитися на безпеці даних. Почніть із запиту до вашого IT-відділу про створення базового плану поточних рівнів доступу та політик. Завдяки цьому ви зможете виявити недоліки у ваших існуючих процесах та викрити будь-яких серйозних порушників, таких як той, хто керує бізнесом зі свого робочого місця.

– Шифрування даних. Дані під час передачі або дані в русі активно переміщуються з однієї області в іншу, наприклад, через Інтернет або приватну мережу. З іншого боку, дані в стані спокою – це інформація, яка не переміщується активно з одного пристрою на інший або з однієї мережі в іншу, наприклад, інформація, що зберігається на жорсткому диску, ноутбуці, флеш-накопичувачі, або архівована чи збережена іншим способом. Сучасні підприємства повинні захищати конфіденційні дані як під час передачі, так і в стані спокою, оскільки кіберзлочинці розробляють нові способи компрометації систем та крадіжки даних. Шифрування є популярним інструментом для захисту даних під час передачі та в стані спокою, і воно відіграє важливу роль у захисті даних.

– Освіта для всіх зацікавлених сторін даних. Це може здатися

несподіванкою, але співробітники організації несуть найбільші ризики для її кібербезпеки. Хоча співробітники компанії є найбільшими носіями ризиків, вони також мають найбільшу користь для кібербезпеки організації. Завдяки постійній освіті та комплексній програмі навчання з безпеки, співробітники можуть забезпечити додаткову безпеку, виступаючи в ролі ще одного рівня захисту.

– Застосовуйте доктрину найменших привілеїв. Гарною відправною точкою для встановлення контролю доступу є використання Доктрини найменших привілеїв, яка по суті базується на принципі, що людина не повинна мати доступу до чогось, якщо їй не потрібно з цим працювати.

– Аудит та моніторинг. Аудит та моніторинг – це гарні заходи для забезпечення безпеки контролю доступу до бази даних. Оскільки співробітники більш схильні перевіряти обмеження доступу, коли ніхто не спостерігає, компанії можуть нагадувати своїм співробітникам, що їхня діяльність з доступу до даних контролюється.

3.2 Розробка структурної схеми

Можливі ризики й приклади реальних інцидентів

До баз даних постійно прикута пильна увага зловмисників, як внутрішніх, так і зовнішніх. Щодня в українських компаніях відбувається безліч інцидентів порушення політик безпеки.

Приклад атаки зсередини

Адміністратор безпеки у фінансовій організації, використовуючи свої права, здійснював переміщення коштів між рахунками. Операція виконувалася в самій базі даних, при цьому журнал реєстрації дій відключався, або вироблялося його чищення.

Приклад атаки ззовні

Дані при їхній передачі між підрозділами компанії не захищалися за допомогою криптографії, залишаючи можливість їхнього прослуховування й зняття копії з мережного встаткування.

Безпечне конфігурування

Створення автоматизованого середовища забезпечення безпечної роботи з базами даних повинне містити в собі: виявлення баз даних, сканування по профілях безпеки, фіксація конфігурацій і т.п.

Навіть у випадку якщо база даних пройшла процедуру настроювання параметрів безпеки, це не виключає постійного контролю її стану, через те, що користувачі своїми діями можуть знижувати захищеність, неусвідомлено відкриваючи можливості для проникнення зловмисника й експлуатації уразливостей.

Технічні засоби забезпечення безпеки баз даних

Забезпечення безпеки Web-додатків

Через те що Web-додатки є невід'ємною частиною практично будь-якої бази даних, заходи, спрямовані на їхній захист, дозволяють за порівняно короткий строк, без впливу на сам процес роботи з базами даних, підвищити їхня захищеність від проникнень із зовнішнього середовища.

Можливості дозволяють реалізувати практично будь-яку конфігурацію, у тому числі й для високонавантажених систем.

Фізичний поділ компонентів БД, контроль доступу на мережному рівні

Поділ компонентів баз даних на мережному рівні повинне вироблятися за допомогою міжмережевих екранів, що забезпечують контроль доступу з перевіркою стану сесії й бажано з перевіркою користувачів по їх обліковим даним (інтеграція з LDAP, Active Directory та ін.).

Крім реалізації функцій поділу, бажано щоб міжмережевий екран мав функціонала виявлення й запобігання вторгнень.

Наявність наочних засобів адміністрування й візуалізації подій у міжмережевих екранах, є безсумнівним плюсом через прямий вплив на операційні витрати, пов'язані з розслідуванням інцидентів. Як можливі варіанти для рішення даного завдання є продукти компанії Check Point Software Technologies, PaloAlto Networks, Cisco Systems, Інфотекс, Код Безпеки й т.п.

Контроль доступу й аудит дій користувачів і адміністраторів

Реалізація функцій по контролі доступу, поділу прав, і виконанню аудита всіх дій, може бути виконана убудованими засобами БД. Однак, слід зазначити, що включення функцій по убудованому аудиті, як того вимагають завдання по забезпеченню безпеки (тобто не тільки базовий набір команд), приводить до зростання навантаження на апаратні ресурси від 10 до 30% залежно від бази даних, що у свою чергу тягне неефективну розтрату дорогих апаратних ресурсів. Реалізація функцій аудита убудованими засобами БД, у кожному разі залишає можливість адміністраторові відключити цього аудита.

У зв'язку із цим найбільш раціональним шляхом є винесення завдань по аудиті на зовнішню систему, що забезпечує запис всіх дій, виконуваних у БД користувачами й адміністраторами.

У загальному виді, рішення зможе складатися з декількох компонентів різних функцій, що забезпечують реалізацію, зокрема:

Сервер керування – для керування й збору даних із всіх компонентів рішення;

Шлюз – який реалізує функції розмежування доступу й аудита для мережних обігів;

Агент – який реалізує функції розмежування доступу й аудита для операцій, виконуваних безпосередньо із БД (даний спосіб роботи повинен бути максимально обмежений).

Звичайний міжмережевий екран не дозволяє працювати з даними на рівні sql – запитів, розбираючи їх і вишиковуючи для них профіль захисту.

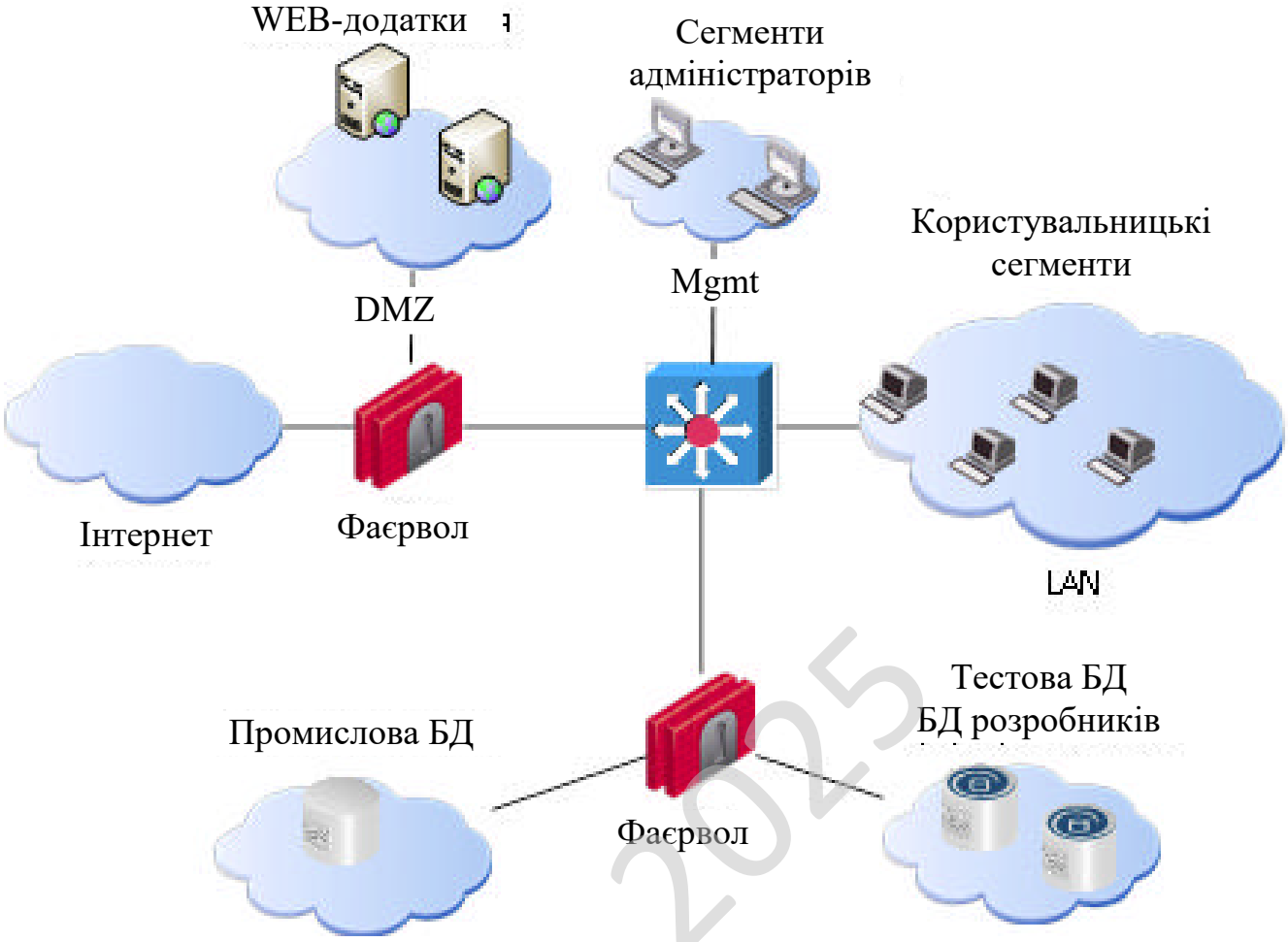

Рисунок 3.1 – Структурна схема системи

Поділ компонентів

Поділ бази даних на середовища розробки, тестування й промислове середовище, є обов'язковою вимогою й невід'ємною частиною всіх рекомендацій з безпеки. Однак це, у свою чергу народжує проблеми своєчасного одержання відновлень, даних із промислового середовища в тестову й розробки, швидкого відновлення й обслуговування декількох копій середовища розробки й тестування (через те, що ті самі відновлення можуть перевірятися як одночасно різними співробітниками й розроблювачами, так і для різних станів однієї й тої ж бази даних). Таким чином, швидке перенесення даних з одного середовища в іншу, відновлення копій баз даних і їхнє адміністрування починає істотно впливати на операційні витрати.

Вирішити дані проблеми можна за допомогою розробленої програми, що також дозволяє паралельно вирішити завдання маскування даних у середовищі тестування й розробки.

Розроблена програма дозволяє віртуалізувати базу даних, швидко й ефективно працювати з копіями баз даних і інформацією в ній (відновлюючи/відкочуючи/видаляючи дані й стан бази даних на бажаний момент часу), а також забезпечити зміна даних стерпних із промислової бази даних у середовища розробки й тестування виконавши тим самим їх маскування.

Немаловажним завданням є перевірка встановлюваних на базу даних і її компоненти відновлень ПЗ. Необхідність постійного відновлення баз даних і її компонент із однієї сторони може бути викликана вимогами бізнесу (по додаванню нового функціонала), а з іншої сторони вимогами безпеки (усунення виявлених уразливостей). Вирішити дане завдання в тому числі можливо шляхом відпрацьовування змін на віртуальних копіях бази даних.

Аналіз захищеності

Постійний контроль за змінами, що відбуваються з базами даних і їхніх компонентів, аналіз їхньої захищеності й схильності уразливостям повинен бути виділений в окремий процес, здійснюваний адміністраторами компанії, адміністраторами баз даних і розроблювачами. Для виявлення й аналізу уразливостей у базах даних варто використовувати сканери безпеки, при цьому їхнє застосування повинне виконуватися в строго погоджені технологічні “вікна” с дотриманням вимог по попередньому резервуванню поточних конфігурацій у базі даних. Сканування за допомогою технічного інструментарію повинне виконуватися по заздалегідь підготовлених профілях сканування актуальним для виконуваного завдання. Результати сканування повинні розбиратися й інтерпретуватися відповідними фахівцями, щоб з однієї сторони максимально точно настроїти профіль сканування, а з інший чітко позначити виявлені недоліки в конфігураціях баз даних, уразливості й відсутні відновлення.

Результатом проведеної роботи повинне з'явитися підтримка незмінності настроювань безпеки в базах даних і її компонентів, виявлення відхилень від затверджених профілів безпеки й своєчасне усунення що виявляються уразливостей шляхом відновлення відповідних компонентів.

Виявлення відхилень у поводженні користувачів/адміністраторів

Рішення, що аналізують профіль поводження користувачів і адміністраторів, дозволяють у тому числі ефективно боротися із шахрайськими діями зловмисників навіть у тому випадку, якщо в них за якимись причинами виявилися легітимні облікові записи. Як подібні рішення можуть застосовуватися системи UBA (User Behavioral Analysis), принцип роботи яких будуватися на підставі складання профілю поводження для кожного контрольованого суб'єкта при його операціях з об'єктами доступу. При цьому варто враховувати факти спрацьовування подібних систем у випадках появи додаткових повноважень у суб'єктів доступу, зміни їхніх прав і додавання нових об'єктів з якими здійснюється робота. Через вищесказаний найбільш доцільним є застосування систем UBA у зв'язуванні з іншими системами, що фіксують правомірність зміни прав суб'єктів доступу, створення нових об'єктів доступу й т.п.

3.3 Розробка функціональної схеми

На рисунку 3.2 зображена функціональна схема системи. Нижче розглянемо її більш докладно.

Функціональна схема складається з наступних блоків:

– Головне вікно програми контролю доступу та аудиту дій при роботі з розподіленою базою даних.

– Блок розмежування доступу.

– Блок менеджеру паролів.

– Блок журналювання подій.

– Допомога.

– Блоки шифрування та дешифрування інформації згідно алгоритму 3DES.

Розглянемо ці блоки більш детально.

Головне вікно програми контролю доступу та аудиту дій при роботі з розподіленою базою даних. Головне вікно призначене для швидкого доступу до основних функцій програми й меню. Програма складається з головного вікна, розташованого у верхній частині екрана й набору незалежних дочірніх вікон. Розташування й розміри вікон можна змінювати за допомогою миші. Також існує можливість закрити непотрібні дочірні вікна (знову відобразити їх можна шляхом вибору відповідних пунктів у меню натисканням на аналогічні кнопки в головному вікні програми). Всі зроблені зміни збережуться в наступному сеансі роботи. Призначення всіх кнопок у програмі пояснюється спливаючими підказками: підведіть покажчик миші до будь-якої кнопки й затримаєте його – з'явиться спливаюча підказка із призначенням кнопки. Головне меню надає доступ до основних списків і функцій системи.

Блок розмежування доступу. Призначений для організації безпечного доступу співтовариства користувачів до захищених ресурсів. Члени цього співтовариства, використовуючи програму, одержують визначені переваги. Це дає наступні можливості:

– Надавати користувачам доступ до інформації (наприклад, групи структурних схем, адресні довідники відділів або пошук співробітників) і ресурсам (наприклад, устаткування або облікові записи у внутрішніх системах), у яких вони бідують, буквально з першого дня.

– Синхронізувати кілька паролів з одним ім'ям користувача для всіх систем.

– При необхідності оперативно змінювати або відзивати права на доступ (наприклад, при переході співробітника в іншу групу або при звільненні).

– Підтримувати відповідність урядовим постановам.

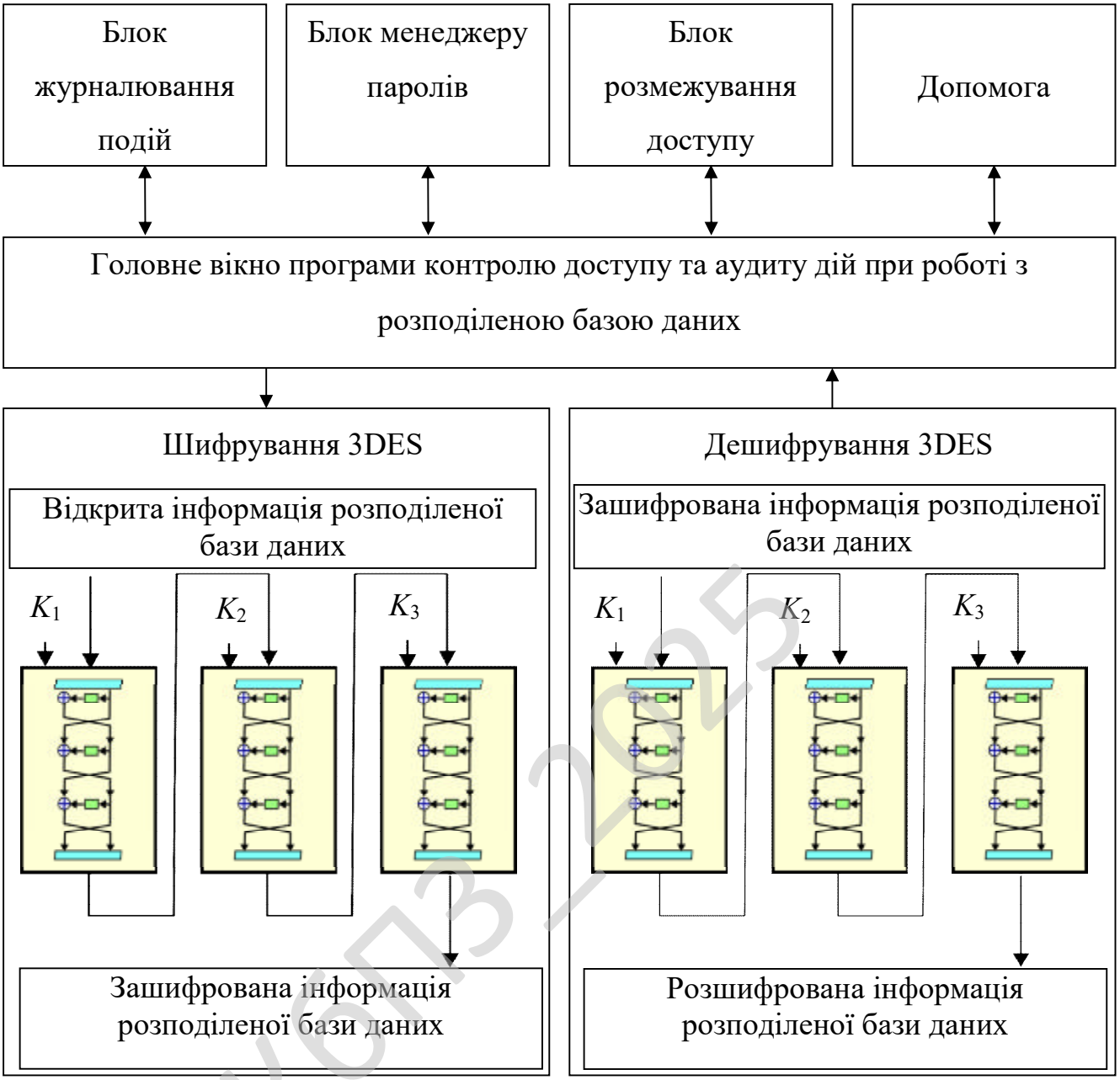

Рисунок 3.2 – Функціональна схема системи

У цей блок включені наступні можливості.

Самообслуговування облікового запису, що дозволяє:

– відображати структурні схеми;

– повідомляти про додатки, пов'язані з користувачем, для адміністратора;

– змінювати дані профілю;

– виконувати пошук у каталозі;

– змінювати пароль, відповідь на запит-відповідь пароля і його підказку;

– переглядати стан політики й синхронізації пароля;

– створювати облікові записи для нових користувачів і груп (при наявності відповідних повноважень).

Запити й твердження, що дозволяють:

– запитувати ресурси;

– перевіряти підтвердження запитів на ресурси;

– працювати із призначеними завданнями підтвердження інших запитів на ресурси;

– виконувати запити й твердження в якості чиєїсь довіреної особи або делегата;

– призначати кого-небудь ще довіреною особою або делегатом (при наявності відповідних повноважень);

– управляти всіма цими функціями запитів і підтверджень в інтересах Вашої групи (при наявності відповідних повноважень);

– при необхідності для кожного запиту або підтвердження надавати цифровий підпис.

Ролі, що дозволяють виконувати наступні дії:

– запитувати призначення ролей і управляти процесом підтвердження запитів на призначення ролей;

– перевіряти стан Ваших запитів ролей;

– визначати ролі і їхні взаємини;

– визначати обмеження поділу обов'язків (SoD) і управляти процесом підтвердження у випадках, коли користувач запитує перевизначення обмеження;

– переглядати довідник ролей;

– переглядати докладні звіти, у яких перераховані ролі й обмеження поділу обов'язків, визначені в довіднику, а також поточний стан призначення ролей, виключення поділу обов'язків і повноваження користувача.

Модуль "Дотримання" дозволяє:

– Запитувати підтвердження профілю користувача.

– Запитувати підтвердження поділу обов'язків (SoD).

– Запитувати підтвердження призначення функцій.

– Запитувати підтвердження призначення користувача.

Блок менеджеру паролів. Надає можливості не тільки для простого збереження паролів, але й для повноцінної роботи з ними. Програма підтримує роботу з декількома аккаунтами, і працювати з нею можуть трохи користувачів. При цьому бази даних кожного користувача шифруються.

Додаткові можливості:

– Система пошуку по базі даних.

– Підтримка макросів.

– Можливість резервного копіювання бази даних.

– Можливість швидкого перемикання між користувачами.

– Швидкий доступ до часто використовуваних функцій.

– Генератор паролів.

– Можливість роздруківки паролів.

Блок журналювання подій. Призначений для запису у журнал усіх подій, які відбуваються у системі. Журнал дій користувачів містить форму для запуску архівації журналу. Форма архівації являє собою кнопку "Очистити журнал" і поле з датою "по:". Дату можна встановлювати будь-яку, але не раніше, ніж поточна дата мінус 1 місяць, щоб у системі завжди зберігалися дані про дії користувачів як мінімум за місяць.

Після натискання кнопки "Очистити журнал" у Системі генерується текстовий файл із архівом журналу за обраний період. Файл зберігається в зашифрованому виді, а посилання на цей файл показуються адміністраторові. Після створення файлу запису журналу за обраний період віддаляються з бази даних. У випадку помилки при створенні або збереженні файлу, записи не видаляються.

Допомога. Блок призначений про надання допомоги по роботі з системою, а також для надання інформації про розробників системи, версію та дату випуску.

Блоки шифрування та дешифрування інформації згідно алгоритму 3DES. Призначені для шифрування та дешифрування інформації, до якої користувач має доступ, згідно прав доступу.

Розглянувши усі блоки функціональної схеми перейдемо до розгляду діаграми взаємодії процесів, які відбуваються у системі.

Розглянувши усі блоки функціональної схеми перейдемо до розгляду діаграми взаємодії процесів, які відбуваються у системі.

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.3. При детальному її розгляді можна побачити як саме проходе взаємодія у розробленій системі. Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

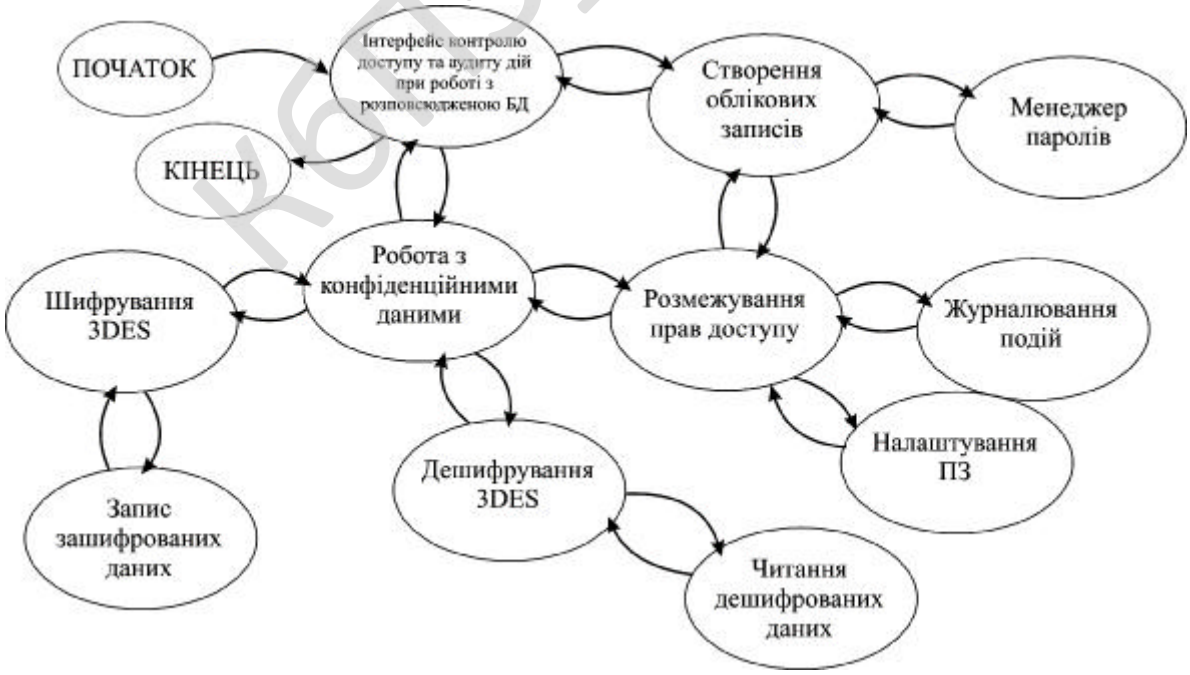

Рисунок 3.3 – Діаграма взаємодії процесів

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.

Діаграми потоків даних містять чотири типи елементів:

– Процеси які являють собою трансформацію даних в рамках описуваної системи.

– Сховища даних (репозиторії).

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

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

Redmine – вільне серверне ПЗ для управління проектами та відстежування помилок. До системи входить календар-планувальник та діаграми Ганта для візуального представлення ходу робіт за проектом та строків виконання. Redmine написано на мові Ruby і є ПЗ розробленим з використанням відомого веб-фреймворку Ruby on Rails, що означає легкість в розгортанні системи та її адаптації під конкретні вимоги. Для кожного проекту можна вести свої вікі та форуми.

Функціональні можливості:

– Ведення декількох проектів.

– Гнучка система доступу з використанням ролей.

– Система відстеження помилок.

– Діаграми Ганта та календар.

– Ведення новин проекту, документів та управління файлами.

– Сповіщення про зміни за допомогою RSS-потоків та електронної пошти.

– Власна Wiki для кожного проекту.

– Форуми для кожного проекту.

– Облік часових витрат.

– Налаштування власних (custom) полів для задач, затрат часу, проектів та користувачів.

– Легка інтеграція із системами керування версіями (SVN, CVS, Git, Mercurial, Bazaar и Darcs).

– Створення записів про помилки на основі отриманих листів.

– Підтримка LDAP автентифікації.

– Можливість самореєстрації нових користувачів.

– Багатомовний інтерфейс (у тому числі українська мова).

– Підтримка СКБД: MySQL, PostgreSQL, SQLite.

Діаграма Ганта (Gantt chart, також стрічкова діаграма, графік Ганта) – це популярний тип діаграм, який використовується для ілюстрації плану, графіка робіт за будь-яким проектом. Є одним з методів планування та управління проектами.

Діаграма Ганта являє собою відрізки (графічні плашки), розміщені на горизонтальній шкалі часу. Кожен відрізок відповідає окремому завданню або підзадачі. Завдання і підзадачі, складові плану, розміщуються по вертикалі. Початок, кінець і довжина відрізка на шкалі часу відповідають початку, кінцю і тривалості завдання. На деяких діаграмах Ганта також показується залежність між завданнями.

Діаграма може використовуватися для представлення поточного стану виконання робіт: частина прямокутника, що відповідає завданню, заштриховується, відзначаючи відсоток виконання завдання; показується вертикальна лінія, що відповідає моменту «сьогодні».

Часто діаграма Ганта використовується спільно з таблицею зі списком робіт, рядки якої відповідають окремо взятій задачі, зображеній на діаграмі, а стовпці містять додаткову інформацію про задачу.

Система відстеження помилок Багтрекер – прикладна програма для допомоги розробникам програмного забезпечення (програмістам, тестувальникам тощо) враховувати і контролювати помилки, знайдені у програмах, питання щодо функціональності, рішення та оновлення, побажання користувачів, а також стежити за процесом їх виконання.

Кожному, хто розробляв програмні продукти, добре знайоме співвідношення «20/80» – останні 20 % роботи тривають 80 % часу.

Як це не парадоксально, але нічого дивного в цій пропорції немає, адже саме на завершальній стадії починається тестування проекту, коли виявляються помилки, і що більший проект, то більше буде знайдено помилок.

Водночас досить часто виявляється, що більшість цих помилок були відомі та могли бути виправлені з меншими витратами на попередніх стадіях роботи, але не були вчасно описані, а потім загубилися серед інших важливих завдань.

Отже, система відстеження помилок у найпростішому варіанті – це процес, що включає в себе виявлення помилки, її опис, виправлення і перевірку цього виправлення, тобто процес «стеження» за багом протягом всього як його життєвого циклу, так і життєвого циклу розробки в цілому.

Сукупність інформації про дефект. Головний компонент такої системи – база даних, що містить відомості про виявлені дефекти. Ці відомості можуть включати в себе:

– номер (ідентифікатор) дефекту;

– хто повідомив про дефект;

– дата і час виявлення дефекту;

– версія продукту, в якій виявлено дефект;

– серйозність (критичність) дефекту та пріоритет рішення;

– опис кроків для відтворення дефекту (неправильної поведінки програми);

– відповідальний за усунення дефекту;

– обговорення можливих рішень та їх наслідків;

– поточний стан виправлення дефекту;

– версії продукту, в якій дефект виправлений.

Крім того, розвинені системи надають можливість прикріплювати файли, які допомагають описати проблему, наприклад, дамп пам'яті або скріншот.

Використання. Основна перевага систем відстеження помилок полягає в забезпеченні чітких централізованих оглядів, запитів на розробку (включаючи помилки і виправлення) та їх стан. У корпоративному середовищі, системи відстеження помилок можуть бути використані для генерації звітів по

продуктивності програмістів виправлення помилок. Однак, це може іноді приводити до неточних результатів, тому що різні помилки можуть мати різні ступені пріоритету та серйозності, що пов'язано з складністю їх фіксації.

Життєвий цикл дефекту. Як правило, система відстеження помилок використовує той чи інший варіант «життєвого циклу» помилки, стадія якого визначається поточним станом помилки.

Типовий життєвий цикл дефекту:

1. Новий – дефект зареєстрований тестувальником.

2. Призначений – призначений відповідальний за виправлення дефекту.

3. Дозволений – дефект переходить назад у сферу відповідальності тестувальника. Як правило, супроводжується резолюцією, наприклад:

– Виправлено (виправлення включені у версію таку-то).

– Дубль (повторює дефект, що вже знаходиться в роботі).

– Не виправлено (працює відповідно до специфікації, має занадто низький пріоритет, виправлення відкладено до наступної версії тощо).

– «В мене все працює» (запит додаткової інформації про умови, в яких дефект проявляється).

4. Далі тестувальник проводить перевірку виправлення, залежно від чого дефект або знову переходить у стан «Призначений» (якщо він описаний як виправлений, але не виправлений), або у стан «Закрито».

5. Відкрито повторно – дефект знайдено знову в іншій версії.

Система може надавати адміністраторові можливість налаштування користувачі, які можуть переглядати і редагувати помилки залежно від їх стану, переводити їх в інший стан або видаляти.

У корпоративному середовищі, система відстеження помилок може використовуватися для отримання звітів, що показують продуктивність програмістів при виправленні помилок.

Однак, часто такий підхід не дає достатньо точних результатів через те, що різні помилки мають різну ступінь серйозності та складності. При цьому

серйозність проблеми прямо не стосується складності її усунення.

Під час роботи над магістерською дипломною роботою було створено блок-схеми. 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.

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 елементу. Вхід в елемент позначається лінією, що входить зазвичай у верхню вершину елементу. 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 є першоджерелами стратегії розвитку ПЗ. 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 оптимізації програми високого рівня, також те, що при розробці програми слід надати особливу увагу модулю контролю доступу та аудиту дій при роботі з мережевою базою даних, модулю обробки помилок програми і основному модулю.

На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми.

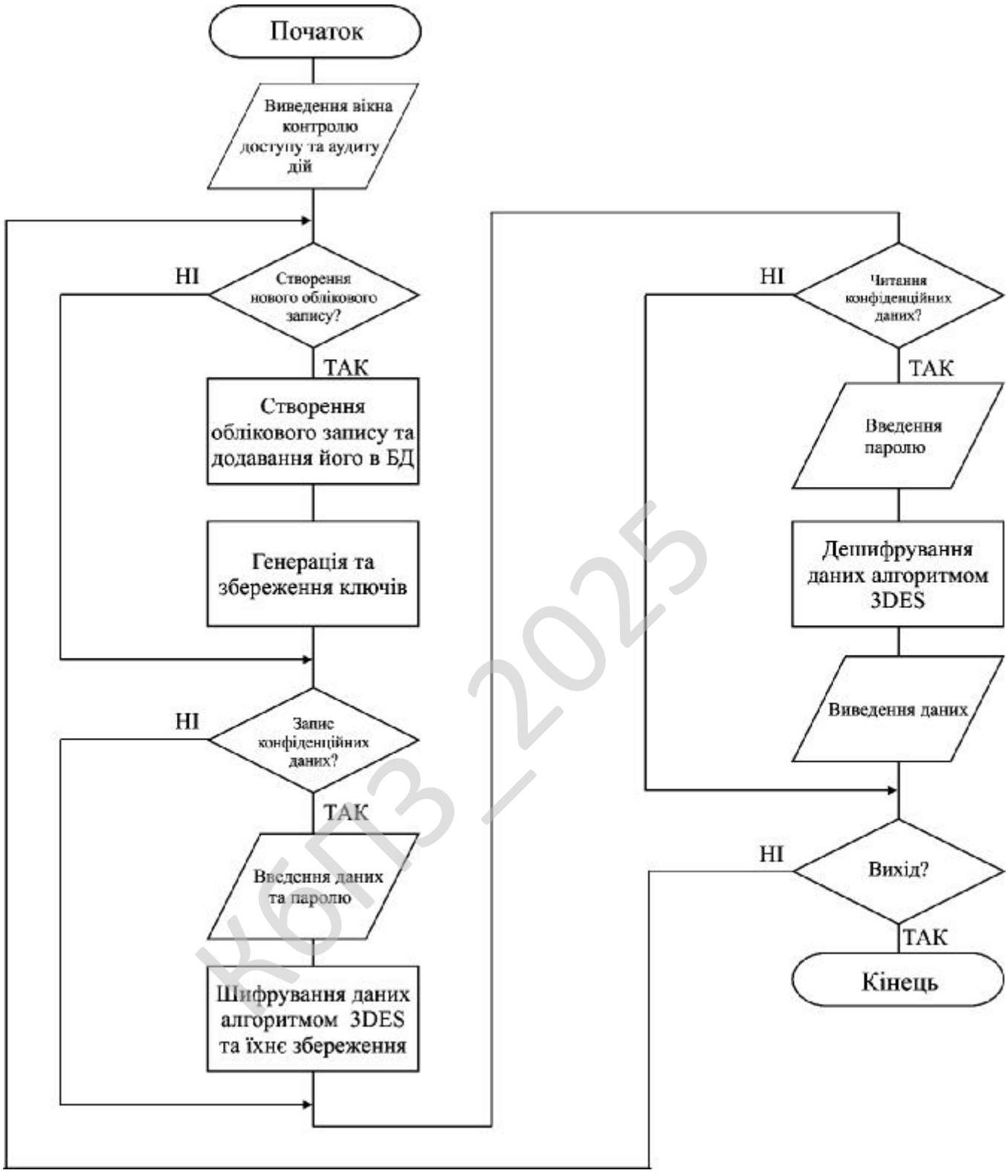

Рисунок 4.1 – Блок-схема основної програми

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 підпрограми 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.

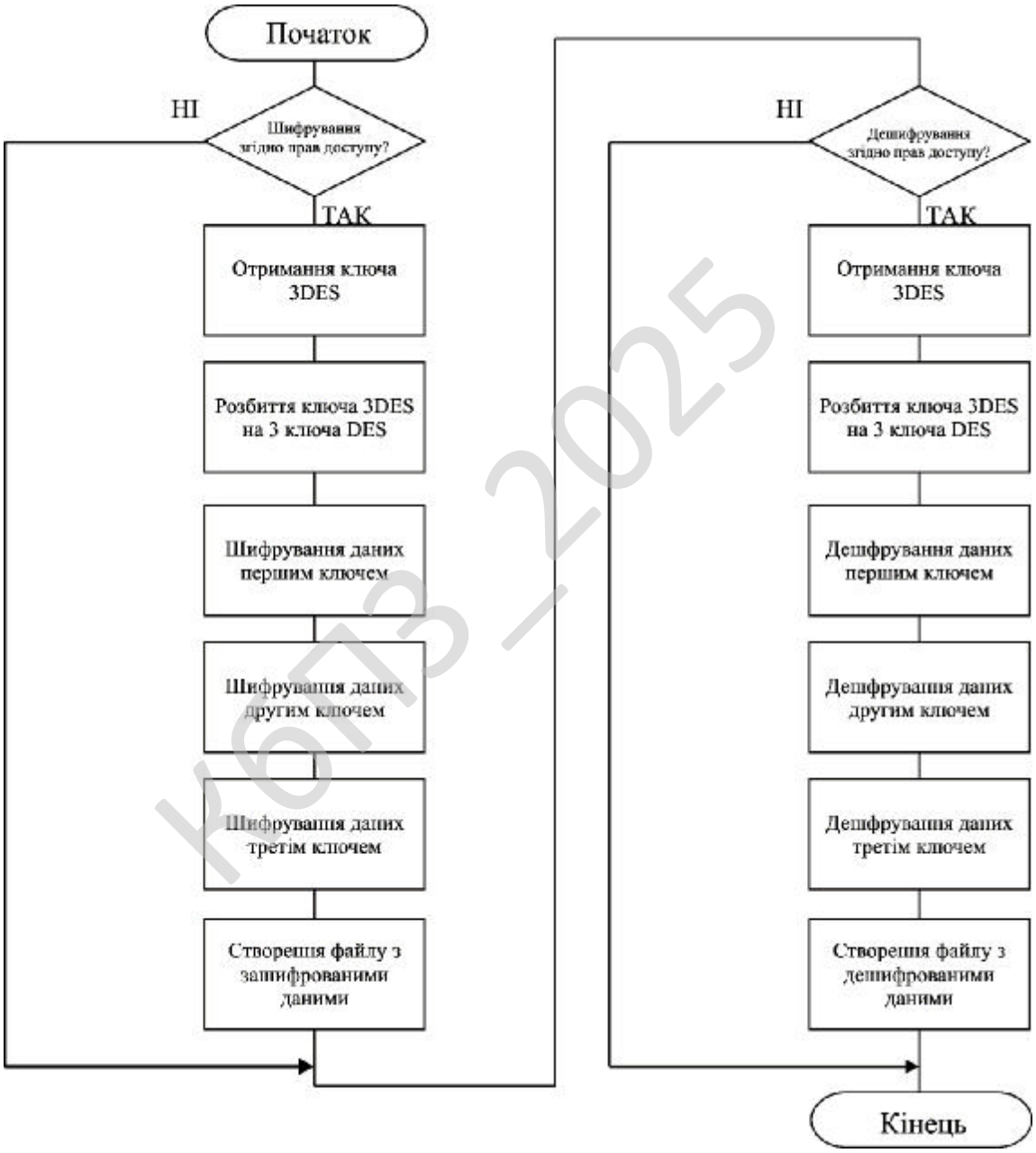

Рисунок 4.2 – Блок-схема роботи підпрограми

Незважаючи на те що я працював над ПЗ один в реалізації програми я використовував підходи пришвидшення розробки на основі методологій Agile.

Гнучка́ розробка програмного забезпечення (Agile software development, agile-методи) – клас методологій розробки програмного забезпечення, що базується на ітеративній розробці, в якій вимоги та розв'язки еволюціонують через співпрацю між самоорганізовуваними багатофункціональними командами.

Гнучка розробка – найкращий засіб для підвищення продуктивності розробників програмного забезпечення.

Більшість гнучких методологій націлені на мінімізацію ризиків, шляхом зведення розробки до серії коротких циклів, що мають назву ітерацій, які зазвичай тривають один-два тижні. Кожна ітерація сама по собі виглядає як програмний проект в мініатюрі, і включає всі завдання, необхідні для видачі мінімального приросту за функціональністю: планування, аналіз вимог, проектування, кодування, тестування і документування. Хоча окрема ітерація, як правило, недостатня для випуску нової версії продукту, мається на увазі те, що гнучкий програмний проект готовий до випуску наприкінці кожної ітерації. Після закінчення кожної ітерації, команда виконує переоцінку пріоритетів розробки.

Agile акцентує увагу на безпосередньому спілкуванні «віч-на-віч». Більшість agile команд розташовані в одному офісі, його іноді називають bullpen. Як мінімум вона включає і «замовників» (замовники, які визначають продукт, також це можуть бути менеджери продукту, бізнес аналітики або клієнти). Офіс може також включати тестувальників, дизайнерів інтерфейсу, технічних авторів і менеджерів.

Основною метрикою agile методів є робочий продукт. Віддаючи перевагу безпосередньому спілкуванню, agile-методи зменшують обсяг письмової документації в порівнянні з іншими методами. Це привело до критики цих методів як недисциплінованих.

Agile – родина процесів розробки, а не єдиний підхід в розробці програмного забезпечення, і визначається Agile Manifesto. Agile не включає практик, а визначає цінності та принципи, якими керуються успішні команди.

Agile Manifesto розроблений і прийнятий 17 розробниками 11-13 лютого 2001 року на лижному курорті The Lodge at Snowbird в горах Юти. Маніфест підписали представники наступних методологій Extreme programming, Scrum, DSDM, Adaptive software development, Crystal Clear, Feature driven development, Pragmatic Programming. Agile Manifesto містить 4 основні ідеї та 12 принципів. Примітно, що Agile Manifesto не містить практичних порад.

Основні ідеї:

– Особистості та їхні взаємодії важливіші, ніж процеси та інструменти.

– Робоче програмне забезпечення важливіше, ніж повна документація.

– Співпраця із замовником важливіша, ніж контрактні зобов'язання.

– Реакція на зміни важливіша, ніж дотримання плану.

Принципи, які роз'яснює Agile Manifesto:

– задоволення клієнта за рахунок ранньої та безперебійної поставки коштовного програмного забезпечення;

– вітання змін вимог навіть наприкінці розробки (це може підвищити конкурентоспроможність отриманого продукту);

– часта поставка робочого програмного забезпечення (кожен місяць або тиждень або ще частіше);

– тісне, щоденне спілкування замовника з розробниками впродовж всього проекту;

– проектом займаються мотивовані особистості, які забезпечені потрібними умовами роботи, підтримкою і довірою;

– рекомендований метод передачі інформації – особиста розмова (віч-на-віч);

– робоче програмне забезпечення – найкращий вимірювач прогресу;

– спонсори, розробники та користувачі повинні мати можливість підтримувати постійний темп на невизначений термін;

– постійну увагу поліпшенню технічної майстерності та зручному дизайну;

– простота – мистецтво не робити зайвої роботи;

– найкращі технічні вимоги, дизайн та архітектура виходять у самоорганізованої команди;

– постійна адаптація до мінливих обставин.

Маніфест та Принципи гнучкої розробки містять високорівневі ідеї щодо того, як потрібно вибудовувати процес розробки програмного забезпечення, щоб успішно завершувати проекти й створювати команди, в яких приємно та цікаво працювати.

Документи визначають, що потрібно для цього зробити, але не говорять, як це зробити. По-іншому й не могло бути, оскільки Маніфест та Принципи народилися внаслідок консенсусу представників різних (хоча й споріднених) напрямів, які могли знайти спільну основу лише на рівні базових цінностей та принципів.

Критика. Багато керівників проектів, що працюють у традиційних методологіях на кшталт «водоспаду», критикують agile-методи.

Один з повторюваних пунктів критики: при agile-підході часто нехтують створенням «дорожньої карти» розвитку продукту, так само як і управлінням вимогами, в процесі якого і формується така «карта». Гнучкий підхід до управління вимогами не має на увазі далекосяжних планів (по суті, управління вимогами просто не існує в даній методології), а має на увазі можливість замовника раптом і несподівано наприкінці кожної ітерації виставляти нові вимоги, що часто суперечать архітектурі вже створеного і поставленого продукту. Таке іноді призводить до катастрофічних «авралів» з масовим рефакторингом і переробками практично на кожній черговій ітерації.

Крім того вважається, що робота в agile мотивує розробників вирішувати всі прибулі завдання найпростішим і найшвидшим можливим способом, при

цьому часто не звертаючи уваги на коректність коду з точки зору вимог базової платформи (підхід «працює, та й добре», при цьому не враховується, що може перестати працювати при найменшій зміні або ж породити важкі до відтворення дефекти після реального розгортання у клієнта). Це призводить до зниження якості продукту і накопиченню дефектів.

Методології. Існують методології, які дотримуються цінностей і принципів заявлених в Agile Manifesto, деякі з них:

1. Agile Modeling – набір понять, принципів і прийомів (практик), що дозволяють швидко і просто виконувати моделювання і документування в проектах розробки програмного забезпечення. Не включає в себе детальну інструкцію з проектування, не містить описів, як будувати діаграми на UML.

Основна мета – ефективне моделювання і документування; але не охоплює програмування та тестування, не включає питання управління проектом, розгортання і супроводу системи. Однак включає в себе перевірку моделі кодом.

2. Agile Unified Process (AUP) спрощена версія IBM Rational Unified Process (RUP), розроблена Скоттом Амблером, яка описує просте і зрозуміле наближення (модель) для створення програмного забезпечення для бізнес-додатків.

3 Agile Data Method – група ітеративних методів розробки програмного забезпечення, в яких вимоги та рішення досягаються в рамках співпраці різних крос-функціональних команд.

4. DSDM заснований на концепції швидкої розробки додатків (Rapid Application Development, RAD). Являє собою ітеративний і інкрементний підхід, який надає особливого значення тривалій участі в процесі користувача/споживача.

5. Essential Unified Process (EssUP).

6. Екстремальне програмування (Extreme programming, XP).

7. Feature driven development (FDD) – функціонально-орієнтована розробка. Використовуване в FDD поняття функції або властивості (feature) Системи досить

близько до поняття прецеденту використання, використовуваному в RUP, істотна відмінність – це додаткове обмеження: «кожна функція повинна допускати реалізацію не більше, ніж за два тижні». Тобто якщо сценарій використання досить малий, його можна вважати функцією. Якщо ж великий, то його треба розбити на декілька відносно незалежних функцій.

8. Getting Real – ітераційний підхід без функціональних специфікацій, що використовується для веб-додатків. У даному методі спершу розробляється інтерфейс програми, а потім її функціональна частина.

9. OpenUP – це ітераційно-інкрементний метод розробки програмного забезпечення. Позиціюється, як легкий і гнучкий варіант RUP. OpenUP ділить життєвий цикл проекту на чотири фази: початкова фаза, фази уточнення, конструювання та передачі. Життєвий цикл проекту забезпечує надання зацікавленим особам та членам колективу точок ознайомлення і прийняття рішень впродовж усього проекту. Це дозволяє ефективно контролювати ситуацію і вчасно приймати рішення про задовільність результатів. План проекту визначає життєвий цикл, а кінцевим результатом є остаточний додаток.

10. Scrum встановлює правила керування процесом розробки та дозволяє використовувати вже існуючі практики кодування, коректуючи вимоги або вносячи тактичні зміни. Використання цієї методології дає можливість виявляти і усувати відхилення від бажаного результату на більш ранніх етапах розробки програмного продукту.

11. Бережлива розробка програмного забезпечення (lean software development). Використовує підходи з концепції бережливого виробництва.

Було використано підходи з використанням UML, 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. UML був створений для визначення,

візуалізації, проектування й документування в основному програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація.

UML може бути застосовано на всіх етапах життєвого циклу аналізу бізнес-систем і розробки прикладних програм. Різні види діаграм які підтримуються UML, і найбагатший набір можливостей представлення певних аспектів системи робить UML універсальним засобом опису як програмних, так і ділових систем.

Діаграми дають можливість представити систему (як ділову, так і програмну) у такому вигляді, щоб її можна було легко перевести в програмний код. Основною причиною використання мови UML є спілкування розробників між собою.

Крім того, UML спеціально створювалася для оптимізації процесу розробки програмних систем, що дозволяє збільшити ефективність їх реалізації у кілька разів і помітно поліпшити якість кінцевого продукту.

UML прекрасно зарекомендувала себе в багатьох успішних програмних проектах. Засоби автоматичної генерації кодів дозволяють перетворювати моделі мовою UML у вихідний код об’єктно-орієнтованих мов програмування, що ще більш прискорює процес розробки. Практично усі CASE-засоби (програми автоматизації процесу аналізу і проектування) мають підтримку UML. Моделі розроблені в UML, дозволяють значно спростити процес кодування і направити зусилля програмістів безпосередньо на реалізацію системи.

Діаграми підвищують супроводжуваність проекту і полегшують розробку документації.

UML необхідний:

– Керівникам проектів, які керують розподілом завдань і контролем за проектом.

– Проектувальникам інформаційних систем які розробляють технічні завдання для програмістів.

– Бізнес-аналітикам, які досліджують реальну систему і здійснюють інжиніринг і реінжиніринг бізнесу компанії.

– Програмістам які реалізовують модулі інформаційної системи.

При модифікації системи об'єктний підхід дозволяє легко включати в систему нові об'єкти і виключати застарілі без істотної зміни її життєздатності. Використання побудованої моделі при модифікаціях системи дає можливість усунути небажані наслідки змін, оскільки вони не ламають структури системи, а тільки змінюють поведінку об'єктів.

Також при розробці магістерської дипломної роботи було використано наступні підходи UML: діаграма діяльності (діаграми поведінки типу); діаграма прецедентів (діаграми поведінки типу); Діаграма класів; Діаграма компонент; Діаграма об'єктів; Діаграма розгортання.

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 це діаграма, на якій зображено відношення між акторами та прецедентами в системі. Також, перекладається як діаграма варіантів використання.

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.

При цьому нічого не говориться про те, яким чином буде реалізована взаємодія акторів із системою.

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).

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.

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. 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 зафарбовані 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 розгортування.

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.

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм SEED – у криптографії симетричний блоковий криптоалгоритм на основі Мережі Фейстеля, розроблений Корейським

агентством інформаційної безпеки (Korean Information Security Agency, KISA) в 1998 році. В алгоритмі використовується 128-бітний блок і ключ довжиною 128 біт. Алгоритм одержав широке поширення й використовується фінансовими й банківськими структурами, виробничими підприємствами й бюджетними установами Південної Кореї, оскільки 40-бітний SSL не забезпечує на даний момент мінімально необхідного рівня безпеки. Агентством по побудови баз даних та захисту інформації специфіковане використання шифру SEED у протоколах TLS і S/MIME. У той же час, алгоритм SEED не реалізований у більшості сучасних браузерів і інтернет-додатків, що утрудняє його використання в даній сфері поза межами Південної Кореї.

SEED являє собою мережу Фейстеля з 16 раундами, 128-бітовими блоками й 128-бітовим ключем. Алгоритм використовує дві 8×8 таблиці підстановки, які, як такі з Safer, виведені з дискретного зведення в ступінь (у цьому випадку, x^{247} і x^{251} – плюс деякі «несумісні операції»). Це є деякою подібністю с MISTY1 у рекурсивності його структури: 128-бітовий повний шифр – мережа Фейстеля з F-функцією, що впливає на 64-бітові половини, у той час як сама F-функція – Мережа Фейстеля, складена з G-функції, що впливає на 32-розрядні половини. Однак рекурсія не простягнеться далі, тому що G-функція – не Мережа Фейстеля. В G-функції 32-розрядне слово розглядають як чотири 8-бітових байта, кожний з яких проходить через одну або іншу таблицю підстановки, потім поєднується в помірковано комплексному наборі булевих функцій таким чином, що кожний біт виводу залежить від 3 з 4 вхідних байтів.

SEED має складний ключовий розклад, генеруючи тридцять два 32-розрядних додаткових символу, використовуючи G-функції на серіях обертань вихідного неопрацьованого ключа, комбінованого зі спеціальними раундовими константами (як в TEA) від «Золотого співвідношення» (англ. Golden ratio).

Згідно з дослідженнями KISA, алгоритм SEED «надійно протистоїть відомим атакам».

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розглянемо розроблене ПЗ контролю доступу та аудиту дій при роботі з мережевою базою даних яке зображено на рисунку 5.1. З рисунку можна побачити що інтерфейс головного вікна розподілено на наступні функціональні розділи:

– Навігаційне меню: Шифрування; Дешифрування; Параметри; Довідка.

– Розділу виведення результату роботи системи – поле введення-вивведення текстових даних.

– Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші.

– Функціональних кнопок ПЗ.

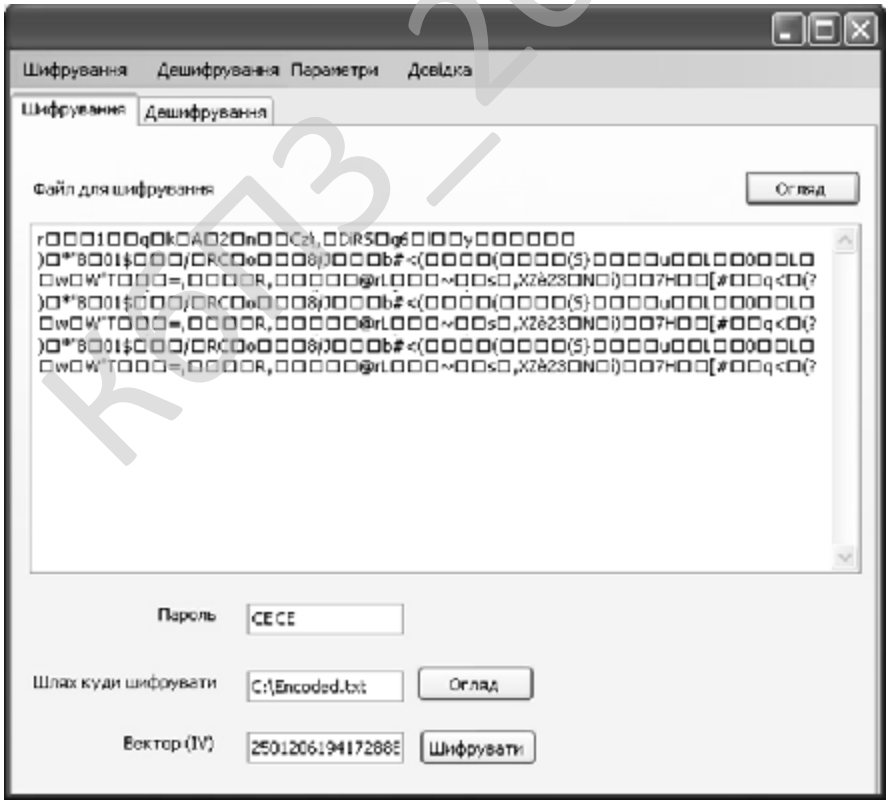

Рисунок 5.1 – Головне вікно ПЗ

Історично розвиток систем безпеки баз даних відбувалася як реакція на дії зловмисників. Ці зміни також були обумовлені загальним розвитком баз даних від рішень на мейнфреймах до хмарних сховищ.

Можна виділити наступні архітектурні підходи:

– повний доступ всіх користувачів до сервера БД;

– поділ користувачів на довірених і частково довірених засобами СУБД;

– введення системи аудита (балок дій користувачів) засобами СУБД;

– введення шифрування даних; винос засобів автентифікації за межі СУБД в операційні системи й проміжне ПЗ; відмова від повністю довіреного адміністратора даних.

Введення засобів захисту як реакції на погрози не забезпечує захист від нових способів атак і формує розрізнене подання про саму проблему забезпечення безпеки.

З обліком таких еволюційних особливостей з'явилося й існує велика кількість різнорідних засобів забезпечення безпеки, що в підсумку привело до відсутності розуміння комплексної безпеки даних. Відсутній загальний підхід до безпеки сховищ даних.

Ускладнюється й прогнозування майбутніх атак, а також розробка захисних механізмів. Більше того, для багатьох систем зберігається актуальність уже давно відомих атак, ускладнюється підготовка фахівців з безпеки.

Розроблена програма має дуже простий і інтуїтивно зрозумілий інтерфейс з користувачем. Кожен, хто в достатньому обсязі володіє операційним середовищем Windows без особливих складностей освоїть і цю програму, оскільки її інтерфейс інтуїтивно зрозумілий.

Якщо програма не видала ніяких помилок, і працює, то можна використовувати, інакше слід слідувати інструкціям, які пропонує програма.

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення.

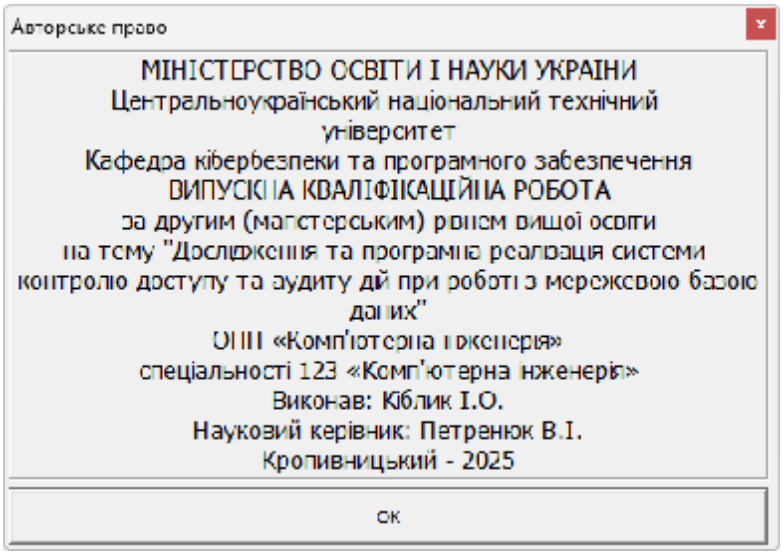

Рисунок 5.2 – Авторське право

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

– Обновлення.

– 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 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 ПЗ.

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.

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.

– У програмі можуть бути пропущені деякі маршрути.

– 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», чию 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 ITc).

– Набір, утворений такими вхідними даними, які демонструють дефекти програми (назвемо його OT).

Будь-який спосіб тестування «чорної скриньки» повинен:

– Виявити такі вхідні дані, які з високою ймовірністю належать набору ITc;

– Сформулювати такі очікувані результати, які з високою імовірністю є елементами набору OT.

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:

– Некоректних чи відсутніх функцій;

– Помилок інтерфейсу;

– Помилок у зовнішніх структурах даних або в доступі до зовнішньої бази даних;

– Помилок характеристик (необхідна ємність пам'яті і т.д.);

– Помилок ініціалізації та завершення.

Обрано умови розповсюдження – Freeware.

Це власницьке програмне забезпечення, котре можна Безоплатно використовувати протягом необмеженого терміну без обмежень у функціональності, і поширюване без сирцевих кодів.

Автори такого програмного забезпечення, як правило, хочуть «дати щось спільноті», але хочуть також контролювати його подальшу розробку. Іноді, коли програмісти вирішують припинити розробку, вони передають сирцевий код іншим програмістам, або ж спільності як вільне програмне забезпечення.

Дуже часто плутають поняття «безплатне програмне забезпечення» та «вільне програмне забезпечення», хоча вони суттєво відрізняються.

Безплатне програмне забезпечення можна безоплатно встановлювати та використовувати (іноді з певними обмеженнями, як, наприклад, «безплатне для домашнього або некомерційного вжитку»), в той час як вільне програмне забезпечення можна продавати за будь-яку суму, але при тому, у користувача, котрий його отримує, повинні бути права на вивчення, модифікацію та поширення сирцевих кодів одержаної програми.

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи контролю доступу та аудиту дій при роботі з мережевою базою даних.

Метою розробки є дослідження та програмна реалізація системи контролю доступу та аудиту дій при роботі з мережевою базою даних.

Об'єктом дослідження є процес контролю доступу та аудиту дій при роботі з мережевою базою даних.

Предметом дослідження є методи контролю доступу та аудиту дій при роботі з мережевою базою даних.

Методи дослідження базуються на методах побудови баз даних та захисту інформації, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод контролю доступу та аудиту дій при роботі з мережевою базою даних.

– Розроблено вітчизняний продукт контролю доступу та аудиту дій при роботі з мережевою базою даних, який має більш широкі можливості, на відміну від існуючих аналогів.

7 МАРКЕТИНГОВЕ ТА ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ ІТ-ПРОЄКТУ

7.1 Визначення цільової аудиторії кінцевого готового продукту

Результати дослідження та розробки системи контролю доступу та аудиту дій при роботі з мережевою базою даних можуть бути насамперед корисними для підприємств, які мають розгалужену ІТ-інфраструктуру й використовують сервери, мережеве обладнання та корпоративні сервіси для підтримки своєї діяльності. Для таких компаній стабільність і безперебійність роботи інформаційних систем є критично важливими, тому можливість своєчасного виявлення несправностей або перевантажень стає суттєвою конкурентною перевагою. Саме система моніторингу допомагає контролювати роботу мережевих пристроїв у режимі реального часу, виявляючи проблеми ще до того, як вони вплинуть на користувачів.

Особливий інтерес до таких систем можуть проявити ІТ-компанії, які займаються наданням послуг хостингу, розробкою програмного забезпечення або підтримкою клієнтів. Для них швидкість реагування на інциденти та якість технічного обслуговування є показниками репутації, а отже, від роботи системи моніторингу залежить рівень довіри клієнтів і лояльність користувачів. Такі підприємства часто працюють у середовищі, де навіть хвилинна затримка чи зупинка сервера призводить до фінансових збитків, тому автоматизація контролю за станом мережі – це не розкіш, а необхідність.

Крім комерційних компаній, результати дослідження будуть актуальними для державних структур, освітніх установ і організацій, які мають внутрішні мережі та зберігають великі обсяги інформації. У таких установах впровадження системи моніторингу підвищує ефективність роботи ІТ-відділів, зменшує ризик втрати даних і допомагає раціонально використовувати наявні ресурси.

Не менш важливим є значення цієї розробки для навчальних і наукових закладів. Вони можуть використовувати систему як навчальну платформу для підготовки фахівців у сфері інформаційних технологій. Студенти отримають можливість не лише спостерігати за реальною роботою системи моніторингу, а й аналізувати дані, моделювати різні ситуації та вчитися реагувати на інциденти. Таким чином, результати дослідження мають універсальний характер і можуть бути впроваджені як у бізнесі, так і в освіті.

7.2 Оцінка привабливості шляхом застосування методів експертних оцінок

Для оцінки привабливості програмного продукту було проведено експертне опитування серед фахівців у галузі IT-інфраструктури, адміністраторів систем і представників компаній, що мають досвід використання схожих рішень. Експертам було запропоновано оцінити систему за основними критеріями – функціональні можливості, надійність, простота впровадження, масштабованість, вартість експлуатації та потенційна економічна ефективність.

Більшість експертів високо оцінили саме інтелектуальну частину системи – можливість автоматичного сповіщення про інциденти, генерацію аналітичних звітів і прогнозування потенційних відмов обладнання. Особливо було відзначено, що система працює стабільно навіть при великому навантаженні й може адаптуватися до різних типів мережевої інфраструктури, що робить її універсальною.

За результатами оцінки середній рівень привабливості продукту склав 8,7 бала з 10 можливих. Експерти зазначили, що така система може мати великий попит серед середніх і великих підприємств, особливо якщо її вартість залишатиметься конкурентною. Також було підкреслено, що простота інтерфейсу та можливість кастомізації під конкретного користувача є суттєвими перевагами, які підвищують комерційний потенціал рішення.

Таким чином, метод експертних оцінок показав, що система має високу ринкову привабливість, відповідає актуальним потребам бізнесу та може стати успішним продуктом за умови належного маркетингового просування та підтримки користувачів.

7.3 Вибір методу оцінки вартості ПЗ

Для оцінки вартості розробки системи контролю доступу та аудиту дій при роботі з мережевою базою даних доцільно використовувати витратний метод. Він передбачає визначення всіх фактичних витрат, які були понесені під час створення програмного продукту, включаючи оплату праці розробників, витрати на апаратне забезпечення, ліцензії, тестування та впровадження. Такий підхід дозволяє точно визначити базову собівартість проєкту, що є особливо важливим для невеликих команд і стартапів.

Однак, у випадку комерційного впровадження, доцільно поєднати цей підхід із дохідним методом. Дохідний метод дає змогу оцінити майбутні вигоди, які підприємство отримає після впровадження системи. Наприклад, скорочення простоїв серверів, підвищення ефективності роботи персоналу та зменшення витрат на ручну діагностику мережі є прямими джерелами економічної вигоди.

Такий комбінований підхід дозволяє не лише визначити початкову вартість розробки, а й обґрунтувати економічну доцільність проєкту. Він допомагає потенційним інвесторам побачити не просто витрати, а реальні фінансові перспективи, які відкриває впровадження системи.

У результаті використання комбінованої моделі оцінки можна отримати повну картину вартості та окупності проєкту, що стане основою для прийняття управлінських рішень щодо його реалізації чи масштабування.

7.4 Розрахунок економічної ефективності від впровадження реалізованого ПЗ як фактору його привабливості

Компанія має розгалужену IT-інфраструктуру, яка включає сервери, мережеве обладнання, робочі станції, системи зберігання даних і корпоративні сервіси. До впровадження системи моніторингу контроль за станом мережі здійснювався вручну: адміністратори виявляли проблеми лише після звернень користувачів або повного виходу сервісів із ладу. Це призводило до простоїв, затримок у роботі та фінансових втрат. Основна мета впровадження системи мережевого моніторингу – забезпечити цілодобове автоматичне відстеження стану обладнання, серверів і додатків, оперативне реагування на інциденти, зниження кількості простоїв і запобігання критичним збоям у роботі IT-інфраструктури. Вхідні дані зафіксовано в таблиці 7.1.

Таблиця 7.1 – Вихідні дані для розрахунку

Показник	До впровадження	Після впровадження	Економічний ефект
Кількість простоїв серверів на рік	20 випадків	5 випадків	–15
Середня тривалість простою одного сервера	4 години	1 година	–3 години
Середні втрати підприємства за 1 годину простою	25 000 грн	5 000 грн	–20 000 грн
Витрати на ручну діагностику й усунення збоїв	300 000 грн/рік	150 000 грн/рік	–150 000 грн
Вартість впровадження системи моніторингу	—	—	450 000 грн
Річні витрати на підтримку системи	—	—	100 000 грн

Розрахунок економічного ефекту демонструє наступне: зменшення збитків від простоїв – 1 975 000 грн/рік, економія на технічному обслуговуванні – 150 000 грн/рік, сукупний річний ефект – 2 125 000 грн/рік, чистий ефект – 2 025 000 грн/рік, термін окупності (Payback Period) – 0,22 року (~2,5 місяці), коефіцієнт ефективності (ROI) – 450%.

Додаткові (немонетарні) переваги: підвищення стабільності IT-інфраструктури завдяки ранньому виявленню збоїв, зменшення навантаження на IT-персонал через автоматизацію моніторингу, покращення SLA (Service Level Agreement) і задоволеності користувачів, прогнозування потенційних проблем через аналітику та звітність у реальному часі, зростання репутації підприємства, адже мінімізуються ризики затримок у наданні послуг або збою критичних бізнес-процесів.

Таким чином, моніторинг стає не лише технічним інструментом, а й важливою складовою операційної надійності та конкурентоспроможності підприємства.

7.5 Пропозиція алгоритму просування проєкту розробки ПЗ

Просування системи моніторингу має будуватися на поетапному підході, що включає як технічну демонстрацію, так і інформаційне просування. На першому етапі варто створити пілотний проєкт і запропонувати його впровадження у невеликій кількості підприємств для збору відгуків і реальних кейсів. Це дозволить перевірити ефективність системи в реальних умовах і створити довіру до продукту.

Далі важливо забезпечити інформаційну присутність продукту – через участь у галузевих конференціях, IT-форумах, онлайн-презентаціях і спеціалізованих публікаціях. Саме через публічну експертну комунікацію формується репутація розробника та усвідомлення цінності рішення на ринку.

Наступним етапом є розширення партнерських зв'язків. Доцільно співпрацювати з IT-компаніями, які займаються інтеграцією корпоративних систем, адже вони можуть пропонувати продукт своїм клієнтам як частину комплексного рішення. Водночас слід розробити гнучку цінову політику – наприклад, ліцензування за кількістю пристроїв або модель передплати, що зробить продукт доступнішим для малого та середнього бізнесу.

Просування має супроводжуватися технічною підтримкою користувачів, оновленнями та навчанням персоналу. Це створює позитивний досвід використання продукту та сприяє формуванню довгострокових відносин із клієнтами. У підсумку правильна стратегія просування допоможе не лише збільшити продажі, а й побудувати впізнаваний бренд на ринку IT-рішень.

7.6 Оптимізація каналів збуту та шляхів реалізації ПЗ

Для оптимізації каналів збуту варто поєднати прямі продажі з цифровими платформами розповсюдження програмного забезпечення. Власний сайт компанії може стати не лише вітриною продукту, а й каналом комунікації з клієнтами, де вони зможуть отримати демо-версію, консультацію або підтримку. Це сприятиме зниженню витрат на маркетинг і збільшенню довіри. Додатково ефективним буде впровадження партнерської програми для системних інтеграторів і реселерів, які вже мають доступ до корпоративних клієнтів. Така модель дозволяє розширити охоплення ринку без суттєвих додаткових інвестицій.

Також можна запропонувати гібридну форму реалізації: ліцензування для великих компаній і модель SaaS (Software as a Service) для малого бізнесу. Це підвищить доступність системи та дозволить гнучко реагувати на потреби різних сегментів ринку. Ключовим напрямом оптимізації збуту є створення якісного сервісу після продажу – технічна підтримка, регулярні оновлення, аналітичні звіти. Усе це забезпечує стабільність роботи клієнта й стимулює його до подальшої співпраці.

7.7 Визначення ключових факторів успіху конкретного проєкту

Основним фактором успіху є стабільність і надійність системи. Якщо система моніторингу працює без збоїв і забезпечує реальну користь, вона швидко здобуває довіру користувачів. Технологічна якість продукту, його здатність масштабуватися й інтегруватися з іншими IT-рішеннями відіграють ключову роль у його життєздатності.

Другим важливим чинником є професійна команда розробників і технічної підтримки. Клієнти цінують не лише продукт, а й можливість отримати швидку допомогу у випадку проблем або питань. Від рівня компетенції фахівців залежить не лише якість обслуговування, а й довгострокові відносини з партнерами.

Не менш значущим є гнучкість системи – можливість адаптувати її під специфіку кожного клієнта. Різні компанії мають різну інфраструктуру, тому універсальне, але налаштовуване рішення стає перевагою.

І, нарешті, успіх будь-якого IT-проєкту визначається здатністю постійно вдосконалюватися. Регулярні оновлення, впровадження нових технологій і зворотний зв’язок із користувачами формують довіру й підтримують актуальність продукту на ринку. Саме ці чинники разом створюють основу для стабільного розвитку та комерційного успіху системи моніторингу.

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Електронно-обчислювальна машина (ЕОМ) відіграє важливу роль у житті сучасної людини. Кожного дня мільйони людей використовують ЕОМ для пошуку необхідної інформації, спілкуванні у соціальних мережах, перегляду новин, роботи тощо. Багато людей користуються ЕОМ у професійних цілях, оскільки завдяки ЕОМ з'явилося багато нових професій. Тому для розробника хмарних сервісів так важливо розробити зручний інтерфейс для зручного сприйняття інформації, та необхідний функціонал, який буде відповідати необхідним вимогам та навантаженням. Все це вимагає багато часу та великого навантаження з боку розробників. Тому так важливо слідкувати за умовами праці, в яких відбувається робочий процес. Оскільки захворювання можуть бути спричинені надмірним фізичним або розумовим навантаженням, через велику нервово-емоційну напругу, або через виробниче середовище. В даному розділі магістерської роботи проведемо аналіз основних чинників при роботі програміста.

Законом України "Про охорону праці" регламентуються загальні положення державної політики в галузі охорони праці, а конкретизуються ці положення нормативно-правовими актами про охорону праці, зокрема Наказом Міністерства соціальної політики України 14.02.2018 № 207, який зареєстровано в Міністерстві юстиції України 25 квітня 2018 р. за №508/31960 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями», яким затверджено нормативно-правовий акт з охорони праці НПАОП 0.00-7.15-18, «Правила охорони праці під час експлуатації електронно-обчислювальних машин», та «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98.

8.2 Шкідливі і небезпечні фактори при роботі з комп'ютером

Електронно-обчислювальна машина (ЕОМ) та інше обладнання є джерелами небезпеки ураження електричним струмом. Так як робота програміста характеризується істотним зоровим навантаженням, то вимагає належного освітлення. У приміщенні, в якому працюють люди (у т.ч. програмісти) необхідно створити належний мікроклімат, параметри якого регламентуються, Державними санітарними правилами і нормами, зокрема ДСанПіН 3.3.2.007-98.

На роботу програміста впливають наступні фактори: невідповідний мікроклімат приміщення (температура, вологість), недостатня освітленість робочої зони, підвищений рівень шуму та електромагнітного випромінювання, порушення іонного складу повітря, неправильна ергономічна організація робочого місця, ризики, пов'язані із погіршенням зору, порушенням фізичного стану, стресом тощо.

Шкідливими факторами при роботі з персональним комп'ютером є неіонізуюче випромінювання промислової частоти, збільшене нервово- емоційне навантаження на оператора, збільшення навантаження на органи зору та дрібні стереостатичні рухи кінцівок. Ці фактори можуть викликати у працівника певні розлади здоров'я, зокрема підвищення артеріального тиску, кон'юктивіти, тендовагініти та інші захворювання.

Комп'ютер, як і будь-який електричний прилад, особливо при його неправильному підключенні, може бути джерелом ураження оператора електричним струмом. Саме тому всі працівники, які працюють з персональним комп'ютером, повинні мати першу (або другу) групу допуску з електробезпеки.

Через наявність зазначених факторів працівники, які працюють з персональними комп'ютерами, підлягають попередньому та періодичному медичному огляду згідно з пунктом 6.2.3 додатку 4 до наказу Міністерства охорони здоров'я України «Про затвердження Порядку проведення медичних оглядів працівників певних категорій» від 21 травня 2007 року № 246.

8.3 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста

Оптимальна температура в приміщенні для праці має становити 20-24°С, відносна вологість – 40-60 %, атмосферний тиск – 750 мм. рт. ст., запиленість не повинна перевищувати 10 мг/м^3, швидкість руху повітря – 0,1 м/с.

Через те, що обчислювальна техніка є джерелом тепловиділення, організація мікроклімату потребує додаткових зусиль: кондиціонування, провітрювання, використання систем опалення тощо. Об’єм приміщень повинен передбачатися з урахуванням як мінімум 20 м^3 /на особу [4].

Монітори комп’ютерів є джерелом випромінювання, яке може зашкодити здоров’ю людини. Для убезпечення роботи з комп’ютером відстань від монітора повинна становити не менше 50 см, бажано використовувати монітори зі зниженим рівнем, скорочувати час безперервної роботи за комп'ютером (робити п’ятнадцяти хвилинні перерви після кожних півтори години праці). Також в приміщенні необхідно встановлювати іонізатори повітря, використовувати нейтралізатори та зволожувачі.

Комп’ютери та периферійні пристрої є джерелами шуму, висока інтенсивність якого може призвести до проблем з органами слуху та негативно впливати на психологічний стан. Рівень шуму на робочому місці не повинен перевищувати 50 дБА [5]. Для зменшення рівня шуму можна використовувати звукопоглинальні пристрої, а стіни приміщень з комп’ютерами можуть бути покриті звукопоглинальними матеріалами. Поряд із шумом часто виникає вібрація. Для зменшення рівня вібрації в приміщенні на поверхні необхідно встановлювати віброізолятори.

Ергономічні показники робочого місця програміста мають бути наступними: висота робочої поверхні повинна складати 720 мм, розмір поверхні має становити 1600 х 1000 мм; під столом повинен бути простір з розмірами по глибині 650 мм; стіл повинен мати підставку для ніг, розташовану під кутом

15°до поверхні; відстань клавіатури від краю столу має бути не більше 300 мм; відстань між очима й екраном повинна складати 40 – 80 см; стілець повинен мати підйомно-поворотний механізм; висота сидіння має регулюватися в межах 400 – 500 мм, глибина – не менше 380 мм, а ширина – не менше 400 мм, висота опорної поверхні спинки має бути не менше 300 мм, ширина – не менше 380 мм. Кут нахилу спинки стільця до площини сидіння повинен змінюватися в межах 90 – 110° [6].

Проведений аналіз показує, що показники мікроклімату в приміщенні відповідають установленим нормам. Штучне опалення застосовується у холодний період року.

В літню пору застосовується кондиціонер.

Для боротьби з пилом робляться регулярні провітрювання та вологі прибирання приміщенні.

У приміщенні знаходяться наступні джерела шуму: принтер Prinics PicKit M1 Smartphone Photo Printer White, електродвигуни вентиляторів ЕОМ.

Робота програміста передбачає постійний візуальній контакт з моніторами комп'ютерів, та, як наслідок, значне навантаження на зір. Традиційно, це зорова робота високої або середньої точності. Для зорової роботи високої точності загальне освітлення (розподіл світла у всьому об'ємі приміщення) має становити 300 лк, комбіноване освітлення (поєднання загального і місцевого освітлення) – 750 лк. Штучне освітлення повинно бути рівномірним та використовуватися в світлий і темний час доби. Джерелами штучного освітлення можуть слугувати люмінесцентні лампи. Правильне освітлення передбачає уникнення відблисків на екранах.

З 2019 року діють Державні будівельні норми України "Природне і штучне освітлення" – ДБН В.2.5-28:2018 [4], у яких прописані вимоги до використання всіх освітлювальних приладів, у т.ч. світлодіодних.

Працю працівника, який постійно працює за комп'ютером, згідно ДБН В.2.5-28:2018 [4], можна віднести до роботи з малою точністю (найменший

розмір об'єкта розрізнення від 1 до 5 мм) V-го розряду зорової роботи, з великою контрастністю об'єкта розрізнення (символів на екрані дисплея), з темним тлом (під розряд зорової роботи В). Приміщення можна віднести до 1-ої групи приміщень, у яких проводиться розрізнення об'єктів зорової роботи при фіксованому напрямку лінії зору того, що працює на робочу поверхню. Для такого типу приміщень і розряду зорової роботи нормоване значення коефіцієнта природної освітленості (КПО) робочої поверхні (при поєднаному, спільному освітленні), повинен становити не більше 1,5%, освітленість при штучному висвітленні повинна становити 300 Лк. [4], Крім того все поле зору повинно бути освітлено достатньо рівномірно – це основна гігієнічна вимога. Оскільки яскраве світло на ділянці периферійного зору значно збільшує напруженість очей і, як наслідок, призводить до їх швидкої стомлюваності, ступінь освітлення приміщення і яскравість екрану комп'ютера повинні бути приблизно однаковими.

8.4 Розробка заходів з умов поліпшення охорони праці

Згідно аналізу умов праці в розглянутому приміщенні, ми одержали наступні результати:

– розмірі приміщення, у розрахунку на одному працюючого, відповідають нормативам;

– мікроклімат відповідає нормативному значенню;

– акустичні умови роботи не перевищують нормативних значень;

Таким чином можна припустити, що основною причиною можливого зниження працездатності програміста є психофізіологічний фактор, тому основна пропозиція буде така: дотримання позитивної психологічної атмосфери в колективі та регламентованого режиму праці та відпочинку, організація робочого місця з урахуванням ергономічних вимог.

Рекомендовані заходи: регулярні періодичні наочні огляди персоналом шляхів для евакуації людей із приміщення, відповідно до плану евакуації (який

повинен розташовуватись на видному місці у приміщенні), включення до колективного договору мінімально можливого вмісту аптечок з обов'язковою наявністю масок-клапанів, або іншого спорядження для штучного дихання. Регулярна періодична перевірка параметрів заземлення та занулення (вимірювання опору ланцюга).

Регулярна наочне знайомство персоналу із шляхами для евакуації людей із приміщення відповідно до плану евакуації, забезпечення розподільних щитів спеціальними розетками з заземлюючими контактами; організація заземлення всіх приладів і пристроїв, які працюють при напрузі вище 36 В.

Так як при ураженні електричним струмом у людини може статися фібриляція шлуночків серця, в організації бажано мати дефібрилятор і підготовлений персонал для роботи з ним.

8.5 Розрахункова частина

Проведемо розрахунок штучного освітлення за методом коефіцієнта використання світлового потоку для приміщення ширина якого складає 6 м, довжина – 7 м, висота – 2,9 м.

У зазначеному приміщенні працює 4 людей.

Для того, щоб визначити потрібну кількість світильників, які повинні забезпечити нормований рівень освітленості, визначимо світловий потік, що падає на робочу поверхню за формулою [1]:

$$F = E \cdot S \cdot K \cdot Z / n,$$

де:

F – світловий потік, що розраховується, Лм;

E – нормована мінімальна освітленість, Лк; E = 300 Лк;

S – площа освітлюваного приміщення (у нашому випадку S=6 х 7 = 42 м^2);

K – коефіцієнт запасу, що враховує зменшення світлового потоку лампи в результаті забруднення світильників у процесі експлуатації (його значення

залежить від типу приміщення і характеру робіт, що проводяться в ньому, в нашому випадку $K = 1,5$);

Z – відношення середньої освітленості до мінімальної (зазвичай приймається рівним 1.1... 1.2, у нашому випадку $Z = 1,1$);

n – коефіцієнт використання світлового потоку, (відношення світлового потоку, що падає на розрахункову поверхню, до сумарного потоку від усіх ламп і обчислюється в долях одиниці [8]; залежить від характеристик світильника, розмірів приміщення, забарвлення стін і стелі, що характеризуються коефіцієнтами відбиття від стін ($\rho_{стін.}$) і стелі ($\rho_{стелі}$), значення коефіцієнтів дорівнюють $\rho_{стін} = 50\%$ і $\rho_{стелі} = 50\%$.

Обчислимо індекс приміщення за формулою:

$$i = S/(h\cdot (A+B)),$$

де:

S – площа приміщення, $S = 42$ м2;

h – розрахункова висота підвісу, $h = 2,9$ м (співпадає з висотою стелі, оскільки лампи освітлення закріплюються на стелі);

A – ширина приміщення, $A = 6$ м;

B – довжина приміщення, $B = 7$ м.

Підставимо всі значення у формулу та визначимо індекс приміщення:

$$i=1,4.$$

Знаючи індекс приміщення, за знаходимо $n = 0,29$ (з табличних даних коефіцієнтів використання світлового потоку (n) світильників з відповідним типом лампам) [8]. Підставимо всі значення у формулу, визначимо світловий потік: F=71689 Лм.

Для розрахунку будемо використовувати світлодіодні стельові панелі Delux LED Panel 41 44 Вт, світловий потік яких $F_л = 3600$ Лм.

Число ламп визначається за формулою:

$$N=F/F_л$$

де:

F – світловий потік,

$F_л$ – світловий потік однієї лампи.

Підставимо всі значення у формулу та визначимо індекс приміщення:

$$N= 71689 / 3600 =19,9 \text{ шт.}$$

Приймаємо необхідну кількість світлодіодних світильників 20 шт.

Висновки до розділу

Дотримання всіх необхідних умов праці не лише сприяє збереженню здоров'я працівників, а також підвищує ефективність виробництва в цілому.

З цих міркувань було здійснено аналіз умов праці, призначеного для праці програмістів, проведено розгляд небезпечних та шкідливих факторів, що негативно впливають на програмістів під час роботи. Виконано розрахунок штучного освітлення, як одного з ключових факторів впливу на працездатність та здоров'я програміста. Розроблено заходи з умов поліпшення охорони праці.

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи контролю доступу та аудиту дій при роботі з мережевою базою даних.

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

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання дослідження методів контролю доступу та аудиту дій при роботі з мережевою базою даних.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем контролю доступу та аудиту дій при роботі з мережевою базою даних.

– Досліджена система контролю доступу та аудиту дій при роботі з мережевою базою даних.

– На основі отриманих результатів досліджень створена програмна реалізація системи контролю доступу та аудиту дій при роботі з мережевою базою даних.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання контролю доступу та аудиту дій при роботі з мережевою базою даних.

Проведено аналіз предметної галузі в ході якого були виявлені об'єкти, взаємодія яких носить істотний характер для функціональної діяльності предметної галузі, і їхні основні характеристики; побудована алгоритм і вибраний середовище розробки.

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

Програма реалізована на мові високого рівня Visual C#. Дана мова програмування дозволяє найбільш ефективно обробляти дані. 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 SEED.

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.

Проведено маркетингове та економічне обґрунтування ІТ-проєкту, що дозволило визначити ключові фактори успіху даного проекту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Кіблик І.О. Дослідження та програмна реалізація системи контролю доступу та аудиту дій при роботі з мережевою базою даних // Збірник праць молодих науковців ЦНТУ. – Вип. 15. – Кропивницький: ЦНТУ, 2025.

2. Alyssa Miller. Cybersecurity Career Guide. Manning Publications. 2022. 368 p.

3. Awais Rashid, Howard Chivers, George Danezis, Emil Lupu, Andrew Martin. CyBOK The Cyber Security Body of Knowledge. The National Cyber Security Centre. 2019. 854 p.

4. Loren Kohnfelder. Designing Secure Software. No Starch Press. 2022. 332 p.

5. Samir Kumar Rakshit. Ethical Hacker's Penetration Testing Guide. BPB Online. 2022. 509 p.

6. Corey J. Ball. Hacking APIs. No Starch Press. 2022. 353 p.

7. Kevin Beaver. Hacking for Dummies. John Wiley & Sons. 2022. 419 p.

8. Mark S. Merkow. Practical Security for Agile and DevOps. CRC Press. 2022. 236 p

9. Derek Fisher. Application Security Program Handbook. Manning Publications. 2021. 155 p.

10. Cameron Wyatt PH.D. Kali Linux Tutorial. Independently published. 2021. 60 p.

11. Alex Matrosov, Eugene Rodionov, Sergey Bratus. Rootkits and Bootkits. No Starch Press. 2019. 450 p.

12. 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.

13. 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

14. Kuznetsov, O., Frontoni, E., Kuznetsova, K., Arnesano, M., Smirnov, O. «A secure biometric authentication architecture for blockchain-driven cyber-physical systems». *Security and Privacy of Cyber Physical Systems Emerging Trends Technologies and Applications*, 2025, pp. 193–224.

15. 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.

16. 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.

17. 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.

18. 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.

19. 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.

20. 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.

21. 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.

22. Смірнова Т.В., Конопліцька-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.

23. 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.

24. 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.

25. 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.

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

27. 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

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

29. 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.

30. Smirnov, O., Lakhno, V., Akhmetov, B., Chubaievskyi, V., Khorolska, K., Bebeshko, B. «Selection of a Rational Composition of İnformation 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.

31. 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.

32. 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 hозвитку інформаційних технологій (ІПШРІТ-2023)»* м.Черкаси 6 грудня 2023 року – Черкаси: ЧДТУ.– 2023. – С.251-252.

33. 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.

34. 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.

35. 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.

36. 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,

37. 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.

38. 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.

39. 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.

40. 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.

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

42. 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

43. 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.

44. 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.

45. 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.*

46. 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.

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

48. 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.

49. 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.

50. 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.

51. 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.

52. 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.

53. 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.