Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації

Комплексна бакалаврська дипломна робота на тему:
«Засіб поширення візуальної новели. Частина 2. Захищені бази даних»
08-20.БДР.017.00.000

Виконав: студент 4 курсу групи 1БС-19б
Спеціальності 125 Кібербезпека
_____ Іван САВЧУК

Керівник: к. т. н., доцент, доцент каф ЗІ
_____ Юрій БАРИШЕВ
« 16 » червня 2023 р.

Рецензент: к. т. н., доцент, доцент каф. ПЗ
_____ Наталя БАБЮК
« 16 » червня 2023 р.

Допущено до захисту
Завідувач кафедри ЗІ
д. т. н., професор
_____ Володимир ЛУЖЕЦЬКИЙ
« 16 » червня 2023 р.

Вінниця ВНТУ – 2023 року

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації
Рівень вищої освіти І (бакалаврський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 125 Кібербезпека
Освітньо-професійна програма – Безпека інформаційних і комунікаційних систем

ЗАТВЕРДЖУЮ
Завідувач кафедри ЗІ,
д. т. н., професор
_____ В. А. Лужецький
«20» березня 2023 року

З А В Д А Н Н Я
НА КОМПЛЕКСНУ БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Савчуку Івану Борисовичу

1. Тема роботи: «Засіб поширення візуальної новели. Частина 2. Захищені бази даних»

керівник роботи: Баришев Юрій Володимирович, к. т. н., доцент кафедри ЗІ, затверджено наказом ректора ВНТУ від 20 березня 2023 року №67.

2. Строк подання студентом роботи 17 червня 2023 р.

3. Вихідні дані до роботи:

- архітектура – клієнт-серверна;
- вид баз даних - реляційні;
- нефункціональні вимоги – кросплатформений застосунок сумісний з Windows, Linux, MacOS;

4. Зміст текстової частини: Вступ. 1. Аналіз методів захисту баз даних комп'ютерних ігор. 2. Структура бази даних засобу поширення візуальних новел. 3. Методи захисту даних. 4. Тестування захищеної бази даних. Висновки.Список використаних джерел. Додатки

5. Перелік ілюстративного матеріалу: аналіз загроз комп'ютерним іграм, аналіз методів захисту баз даних, узагальнена архітектура засобу поширення візуальних новел, DR-діаграма нормалізованої бази даних, узагальнений алгоритм роботи бази даних, модель розмежування прав доступу, алгоритм захисту даних на стороні

сервера, результати блокового тестування, результати інтеграційного тесту результат мануального тестування.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Баришев Ю. В., к. т. н., доц. каф. ЗІ	20.05.23	08.05.23
2	Баришев Ю. В., к. т. н., доц. каф. ЗІ	20.05.23	15.05.23
3	Баришев Ю. В., к. т. н., доц. каф. ЗІ	20.05.23	21.05.23
4	Баришев Ю. В., к. т. н., доц. каф. ЗІ	20.03.23	10.06.23

7. Дата видачі завдання 20 березня 2023 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів комплексної бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітки
1	Аналіз завдання. Вступ	20.03.23-26.03.23	
2	Аналіз інформаційних джерел за напрямком бакалаврської дипломної роботи	27.03.23-09.04.23	
3	Розробка рішень, моделей, алгоритмів	10.04.23-23.04.23	
4	Практична реалізація, моделювання, експериментування, результати	24.04.23-21.05.23	
5	Висновки	22.05.23-24.05.23	
6	Оформлення пояснювальної записки	25.05.23-31.05.23	
7	Попередній захист БДР	01.06.23-15.06.23	
8	Виправлення зауважень, підготовка ілюстративного матеріалу	16.06.23-19.06.23	
9	Представлення БДР до захисту, рецензування	20.06.23-23.06.23	

Студент _____ Іван САІ

Керівник роботи _____ Юрій БАРИ

АНОТАЦІЯ

Комплексна бакалаврська дипломна робота складається з 82 сторінок формату А4, на яких є 38 рисунків, 3 таблиць, список використаних джерел містить 32 найменування.

Комплексна бакалаврська дипломна робота присвячена розробці захищеної бази даних для засобу поширення візуальних новел. В результаті аналізу загроз та методів захисту баз даних, обрано такі методи захисту, як рольове розмежування, логування та аудит з використанням процедур та тригерів. Використання наведених методів передбачає побудову комплексного захисту для бази даних. Розроблені алгоритми та методи спрямовані на запобігання несанкціонованому доступу до баз даних та забезпеченню безпеки даних засобу. Після розробки проведено блокове та інтеграційне тестування захищеної бази даних для засобу поширення візуальних новел.

Ключові слова: загрози комп’ютерним іграм, захист застосунків від піратства, захист баз даних, рольове розмежування прав доступу, моніторинг.

ABSTRACT

The complex bachelor thesis consists of 82 of pages of A4 format, on which there are 38 of figures, 3 of tables, the list of used sources contains 32 names.

The complex bachelor's diploma thesis is devoted to the development of a secure database for the means of visual novels distribution. The following protection methods were chosen due to analysis of existing threats and database protection methods,: role-based access control, monitoring and auditing using stored procedures and triggers. The usage of the above-mentioned methods involves the construction of comprehensive protection for the database. The developed algorithms and methods are aimed at preventing unauthorized access to databases and ensuring the security of the tool's data. After development, block and integration testing of a secure database for a visual novel distribution tool was conducted.

Key words: computer game threats, application protection from piracy, database protection, role-based access control, monitoring.

ЗМІСТ

ВСТУП

Еволюція комп’ютерних технологій, включаючи прогрес в апаратному та програмному забезпеченні, відіграла значну роль у формуванні індустрії ігор. Постійне збільшення обчислювальних можливостей дозволило розробникам створювати більш складні та витончені ігри, пропонуючи гравцям безпрецедентний рівень інтерактивності та реалізму. Цей швидкий прогрес призвів до створення захопливих і реалістичних ігор, фанатами яких є мільйони гравців різного віку та статі у всьому світі. Оскільки технології продовжують прогресувати, розробники ігор постійно розширюють межі можливого, створюючи візуально приголомшливу графіку, складну ігрову механіку та захопливу історію для того щоб отримати прихильність з боку користувача і як наслідок великі фінансові надходження.

Розробка ігор напряму пов’язана з великою кількістю даних, які зберігаються у базах даних ігор. Починаючи від графічних об’єктів, які дарують гравцю приємну картинку та позитивні враження від гри, закінчуючи даними про сюжет, сцени та інформацію про користувачів. Бази даних сучасних ігор зберігають та обробляють велику кількість персональних даних користувачів, зокрема такі: паролі, псевдоніми, реальні імена, поштові скриньки, дані платіжних систем, у деяких випадках, дані про місце знаходження. Таке різноманіття конфіденційної інформації не залишається без уваги хакерів.

Мотиви зламу, як і їх методи можуть бути різними. Переважно злами відбуваються з таких причин, як бажання отримати переваги у грі над іншими користувачами, отримання персональних даних облікових записів гравців для подальшого використання у власних цілях, або перепродажу, отримання витоків з бази даних з розробками, частинами коду або графічного матеріалу для шантажування компаній розробників. У будь-якому випадку наслідки завжди однакові. Хакерські атаки на бази даних ігрової індустрії завжди несуть великі збитки для компаній розробників, як з фінансової, так і з репутаційного боку.

Щорічно ігрова індустрія та хакери змагаються у розробці стійких методів захисту та їх зламу. Компанії-гіганти витрачають багато коштів на розробку як інтегрованих таких як: шифрування, обфускація, розмежування прав доступу, а також навісних засобів захисту від зламу до прикладу системи моніторингу, проактивне тестування безпеки, античіт системи, щоб забезпечити цілісність, конфіденційність та доступність своїх продуктів. З тим самим хакери розробляють нові методи зламу, знаходячи нові вразливості в системах. Саме тому постає актуальна задача захисту ігор та їх даних на основі комплексного підходу з використанням різних методів захисту. Лише у такому випадку розробники можуть знизити ймовірність успішних кібератак на їх продукти.

Отже об’єктом дослідження є процес захисту баз даних.

Предметом дослідження є методи та засоби захисту баз даних.

Метою даної комплексної бакалаврської диплосної роботи є покращення захисту бази даних засобу для поширення візуальних новел.

Для досягнення мети потрібно розв’язати такі задачі:

- проаналізувати основні загрози комп'ютерним іграм;
- проаналізувати методи захисту баз даних;
- розробити узагальнену архітектуру засобу;
- розробити захищену базу даних;
- розробити метод розмежування прав доступу;
- розробити алгоритм тригерів та процедур для підвищення захищеності;
- провести тестування.

Результати роботи доповідались на LII Науково-технічній конференції факультету інформаційних технологій та комп'ютерної інженерії (2023) конференції та опубліковані як тези та матеріал доповіді [1].

1 АНАЛІЗ МЕТОДІВ ЗАХИСТУ БАЗ ДАНИХ КОМП'ЮТЕРНИХ ІГОР

1.1 Аналіз загроз комп’ютерним іграм

У сучасному світі комп’ютерні ігри все частіше можна зустріти у комп’ютерах звичайних користувачів. З кожним роком світовий ігровий ринок заповнюється новинками, які перевершують тогорічні релізи. Нині щорічно гейм-індустрія приносить розробникам близько 200 мільярдів доларів [2]. З розвитком комп’ютерних систем та збільшенням їх обчислювальної потужності комп’ютерні ігри стають все більш складнішими, як у своїй функціональній побудові, так і в масштабності даних, які вони містять. Таким чином можна розглянути такі типи даних, які можуть зацікавити зловмисників:

– інформація про обліковий запис користувача – зловмисники можуть атакувати бази даних, щоб отримати імена користувачів, паролі, адреси електронної пошти та іншу інформацію, пов’язану з обліковим записом [3];

– ігрова валюта та віртуальні активи – багато ігор мають власні віртуальні валюти, предмети чи активи, які мають цінність в економіці гри. Зловмисники можуть націлитися на бази даних, щоб викрасти або маніпулювати цими віртуальними активами для особистої вигоди. Потім вони можуть продати або обміняти їх на зовнішніх платформах за реальну валюту;

– профілі гравців і дані про прогрес – профілі гравців часто містять цінну інформацію, таку як досягнення, статистика, рейтинги та прогрес у грі. Зловмисники можуть атакувати бази даних, щоб отримати доступ до цих даних і маніпулювати ними, що може порушити чесну конкуренцію, вплинути на репутацію гравців або навіть призвести до спроб вимагання;

– інтелектуальна власність та ігрові активи – бази даних ігор можуть містити власні ігрові активи, зокрема ілюстрації, аудіофайли, фрагменти коду та іншу інтелектуальну власність. Зловмисники можуть намагатися викрасти або змінити

ці активи, що потенційно може призвести до порушення авторських прав, несанкціонованого розповсюдження або навіть створення підроблених версій гри;

– аналітика та дані про поведінку користувачів – розробники ігор часто збирають дані про поведінку гравців, шаблони ігрового процесу та вподобання користувачів для аналітики та вдосконалення. Зловмисники можуть націлитися на ці бази даних, щоб отримати інформацію про поведінку користувачів, яку можна використовувати для цілеспрямованих фішингових кампаній;

– технічна інформація та вразливості – бази даних також можуть містити технічну інформацію про інфраструктуру гри, конфігурації сервера або версії програмного забезпечення. Зловмисники можуть використати цю інформацію для виявлення вразливостей, здійснення цілеспрямованих атак на певні ігрові сервери чи компоненти або отримання несанкціонованого доступу до основної інфраструктури.

Щоб захистити особисті дані гравців і гарантувати веселу та неупереджену гру, надзвичайно важливо захистити бази даних. Розробники ігор повинні розуміти ці негативні фактори. У динамічному просторі комп’ютерних ігор вкрай важливо вивчити потенційні ризики, які можуть поставити під загрозу безпеку та конфіденційність баз даних ігор. Розробники щорічно намагаються захистити свої продукти, але злам та захист – сфери, що вічно конкурують. На сучасні ігри впливає низка загроз:

– Витік даних і проблеми з конфіденційністю становлять велику загрозу для безпеки конфіденційних даних користувачів. Комп’ютерні ігри часто збирають і зберігають особисту та конфіденційну інформацію гравців, таку як імена користувачів, паролі, адреси електронної пошти, платіжні дані та статистику ігрового процесу. Витоки даних можуть статися через уразливості в ігрових системах, цілеспрямовані атаки або компрометацію сторонніх служб, які використовуються грою. Коли бази даних скомпрометовано, особиста інформація гравців та про гру стає відкритою, що призводить до проблем із конфіденційністю та можливої крадіжки особистих даних. Здебільшого такого типу загроза стосується переважно ігор які знаходяться на стадії релізу. Від такого типу загроз

потерпали такі відомі ігри як: The Last of Us, Cyberpunk 2077 [4]. На стадії релізу, зловмисники отримали доступ до даних гри та розповсюджували вихідний код у мережі інтернет [5]. Це потенційно створювало ризики безпеці та впливало на процес розробки гри.

– Шахрайство та експлуатація становлять серйозну загрозу для чесності та цілісності комп'ютерних ігор. Хакери та зловмисники використовують різні методи, зокрема aimbots, wallhacks, speed hacks і сценарії, щоб отримати несправедливі переваги або порушити ігровий процес для інших [6]. Ці дії часто передбачають несанкціоновані модифікації ігрових клієнтів, впровадження шкідливого коду або маніпуляції з ігровими даними, що зберігаються в базах даних. Гра, яка вірогідно найбільше потерпає від такого типу загрози є Counter-Strike:Global Offencive [7].

– DDoS атаки можуть серйозно вплинути на доступність і продуктивність онлайн-комп'ютерних ігор. Зловмисники заповнюють ігрові сервери надзвичайною кількістю трафіку, роблячи їх недоступними для законних гравців. Це порушує ігровий процес, розчаровує користувачів і потенційно призводить до втрати прибутку для розробників ігор. DDoS-атаки, націлені на ігрову індустрію, охоплюють 37% усіх DDoS-атак [8]. Від даних атак сильніше всіх постраждала гра Rainbow Six Siege, розроблена компанією Ubisoft [9].

– Злам облікових записів і неавторизований доступ. Однією з суттєвих загроз для комп'ютерних ігор є злам облікових записів гравців, що призводить до несанкціонованого доступу до баз даних ігор. Хакери використовують такі методи, як фішинг, соціальна інженерія або атаки грубою силою, щоб отримати облікові дані для входу гравців. У разі успіху зловмисник отримує доступ до баз даних гри, що потенційно може призвести до крадіжки даних, модифікації або збою в роботі ігрових служб. Одна з найпопулярніших загроз, від якої потерпають розробники ігор. Гарним прикладом масового зламу облікових записів, є злам конфіденційних даних користувачів компанії Ubisoft. У 2013 році Ubisoft зазнала масового зламу особистих даних, що вплинуло на облікові записи користувачів на їхній ігровій онлайн-платформі Uplay. Порушення призвело до компрометації імен

користувачів, адрес електронної пошти та зашифрованих паролів багатьох користувачів Uplay [10].

– Інсайдерські загрози стосуються ризиків, створених особами з авторизованим доступом до баз даних гри, які зловживають своїми правами зі зловмисною метою. Це може включати такі дії, як викрадення даних, несанкціоноване маніпулювання даними або витік конфіденційної інформації. Інсайдери можуть включати розробників ігор, адміністраторів або персонал служби підтримки клієнтів, які мають законний доступ до баз даних [11].

Для запобігання таким загрозам, розробники ігор повинні приділяти велику увагу безпеці та захисту даних. Це означає впровадження надійних механізмів шифрування, забезпечення безпеки мережевих протоколів, аудиту та моніторингу системи, а також своєчасну виправлення вразливостей.

1.2 Аналіз методів захисту баз даних

Стійкі заходи безпеки мають вирішальне значення для забезпечення безпеки бази даних у швидкоплинному цифровому світі, особливо в індустрії комп’ютерних ігор. Враховуючи величезні обсяги даних, які збираються, зберігаються та обробляються розробниками ігор і організаціями, важливо захистити бази даних від несанкціонованого доступу, витоку даних та інших загроз безпеці.

Віддаючи пріоритет конфіденційності, цілісності та доступності своїх даних за допомогою ефективних протоколів безпеки, розробники ігор можуть краще захистити свої цінні активи. Обсяг, що постійно зростає, і складність даних, що зберігаються в базах даних, створюють як проблеми, так і можливості. З одного боку, величезна кількість даних дає цінну інформацію та можливості для інновацій, дозволяючи розробникам ігор надавати гравцям більш персоналізований та привабливий досвід від гри. З іншого боку, цей масив даних також привертає увагу зловмисників, які прагнуть використати вразливі місця та отримати

несанкціонований доступ до конфіденційної інформації. Існує велика кількість стратегій захисту баз даних зокрема у сфері ігрової індустрії зокрема таких, як:

- шифрування;
- резервне копіювання;
- контроль доступу;
- аудит;
- системи виявлення вторгнення.

Шифрування є основним методом захисту даних у базах даних. Це передбачає перетворення конфіденційної інформації в незрозумілу форму. Шифрування може бути реалізовано на різних рівнях, наприклад, шифрування повної бази даних, шифрування на рівні поля або шифрування даних під час передачі. Використовуючи стійкі алгоритми шифрування та методи керування ключами, розробники ігор можуть забезпечити конфіденційність і цілісність вмісту своїх баз даних [12].

Регулярне резервне копіювання та стратегії аварійного відновлення є важливими компонентами захисту бази даних. Створення резервних копій баз даних і їх зберігання мінімізує наслідки втрати даних у разі системних збоїв або зловмисних атак. Впровадження планів аварійного відновлення, включаючи процедури резервного копіювання та тестування їх ефективності, може допомогти забезпечити доступність бази даних і цілісність даних [13].

Контроль доступу передбачає керування дозволами та привілеями, наданими користувачам або ролям, які мають доступ до бази даних, гарантуючи, що лише авторизовані особи можуть отримати доступ до певних даних. Контроль доступу на основі ролей (RBAC) — це широко поширений підхід до реалізації контролю доступу в ігровій індустрії. За допомогою RBAC права доступу призначаються на основі попередньо визначених ролей. Розробники ігор визначають різні ролі, наприклад адміністраторів, модераторів і гравців, і призначають відповідні дозволи та привілеї кожній ролі. Це гарантує, що користувачі можуть виконувати лише дії та отримувати доступ до даних, які відповідають їхнім конкретним ролям, мінімізуючи ризик несанкціонованих дій. Рольове розмежування прав доступу

допомагає покращити безпеку, підзвітність і відповідність бази даних, спрощуючи керування нею. Застосовуючи заходи контролю доступу, розробники ігор можуть захистити свої бази даних від несанкціонованого доступу, пом’якшити можливі порушення і зберегти конфіденційність, цілісність і доступність даних гри[14].

Також як додатковий захід захисту використовуються білі списки у системах управління баз даних, їх використання дозволить підключення лише з певних IP-адрес або імен хостів. Це гарантує, що лише авторизовані клієнти можуть підключитися до сервера і отримати доступ до пов’язаних баз даних.

Аудит бази даних надають засоби моніторингу та запису дій у базі даних. Увімкнувши широкі можливості аудиту, розробники ігор та баз даних до них можуть відстежувати та аналізувати доступ до бази даних, модифікації та системні події. Ця інформація є важливою для виявлення підозрілих дій, попередження потенційних інцидентів та допомоги під час відновлення резервних копій. Регулярний перегляд і аналіз журналів аудиту можуть допомогти у виявленні вразливостей і посиленні заходів безпеки [15].

Системи виявлення та запобігання вторгненням (IDPS) відіграють вирішальну роль у виявленні та пом’якшенні атак на бази даних. IDPS відстежує діяльність бази даних у режимі реального часу, виявляє аномальну поведінку та запускає сповіщення або автоматичні відповіді для запобігання потенційним порушенням. Впроваджуючи рішення IDPS із розширеними можливостями виявлення загроз, такими як виявлення аномалій, виявлення на основі сигнатур та аналіз поведінки, розробники ігор можуть покращити свою здатність виявляти потенційні атаки та оперативно реагувати на них [16]. Впроваджуючи дані методи захисту розробники ігор можуть створити надійну систему безпеки, щоб захистити бази даних своїх ігор від потенційних загроз і збільшити безпеку гри для користувачів.

1.3 Аналіз засобів захисту комп’ютерних ігор

Забезпечення безпеки та цілісності комп’ютерних ігор є критично важливим аспектом ігрової індустрії. З поширенням шахрайства та хакерства, появою нових

методів для обходу чесної гри, які стрімко розвиваються розробники ігор дедалі більше покладаються на різноманітні програми захисту які з кожним роком дедалі ефективніше реалізують чесний ігровий процес і захист даних гри та гравців. Аналіз засобів захисту, та ефективне їх використання дозволяє розробникам створити стратегію захисту власних розробок, та комфортної чесної гри для їх користувачів.

Захист цілісності ігрових файлів забезпечується за рахунок використання методу управління цифровими правами, які спрямовані на захист від несанкціонованого копіювання, розповсюдженню та використанню ігор. Як правило дані системи вимагають від користувача введення спеціального ключа (цифрової ліцензії). Розробники часто використовують рішення управління цифровими правами для захисту своїх ігор від піратства та несанкціонованого розповсюдження [17].

На програмному рівні розробники часто використовують обфускацію коду – це техніка яка погіршує його розуміння і, як наслідок, зменшує привабливість його подальшого використання. Обфускація передбачає застосування різних перетворень до коду, таких як перейменування змінних і функцій, додавання непотрібного коду та зміна структур потоку керування без зміни поведінки програми. Даний засіб бути ефективним засобом захисту від несанкціонованого дослідження та неавторизованих модифікацій [18].

Захищені мережеві протоколи. Захищені мережеві протоколи, такі як SSL/TLS (Secure Sockets Layer/Transport Layer Security), використовуються для захисту ігрових комунікацій від підслуховування, втручання та атак типу "людина посередині". Ці протоколи забезпечують механізми шифрування та автентифікації, гарантуючи, що дані, що передаються між ігровим клієнтом і сервером, є безпечними та не можуть бути перехоплені чи змінені зловмисниками. Використання захищених мережевих протоколів збільшує захист конфіденційних даних гравців [19].

Одним з найпоширеніших методів захисту від шахрайства під час ігрового процесу, є використання AntiCheat систем. Ці системи використовують різні

методи, такі як виявлення на стороні клієнта, перевірка на стороні сервера та аналіз поведінки, щоб ідентифікувати та стримувати шахраїв. На сьогоднішній день системи такого роду поділяються на такі види:

– Easy Anti-Cheat – система створена для захисту від чітів, яка інтегрується з ігровими движками для виявлення та запобігання шахрайству в багатокористувацьких іграх. Система сканує ігрове середовище та здійснює профілактичне чищення від проникнень хакерів та гравців, які намагаються зламати системи або обійти правила гри, щоб отримати незаслужені переваги. Дана система використовується у багатьох сучасних іграх, таких, як: Apex Legends, Fortnite, War Thunder та інші [20].

– BattlEye — це вдосконалене програмне забезпечення проти чітів, яке використовується в популярних онлайн-іграх. Під час ігрового процесу система надсилає файли із системи користувача на свої сервери, хоча вони не містять особистої інформації та здійснює перевірку на чесність користувача, у випадку якщо користувач здійснює шахрайські дії, система розпізнає його та блокує [21].

– Valve Anti-Cheat (VAC) — система захисту від чітів. VAC в основному використовується для виявлення та запобігання шахрайству в іграх, що розповсюджуються через Steam. Вона використовує метод «хвилі» збираючи свіжі дані про чіти та їх постачальників, а потім проводить масові блокування шахраїв. Даний метод ускладнює шахраям визначати тригери на які реагує система [22].

Провівши аналіз засобів захисту в комп’ютерних іграх, для полегшення порівняльного аналізу їх використання у різних продуктах доцільно представити все у вигляді таблиці 1.1., де буде розглянута низка ігор, рік релізу та їх методи захисту, такі як:

– Easy Anti-Cheat;
– BattlEye;
– Valve Anti-Cheat;
– захист від копіювання;
– управління цифровими правами;
– SSL/TLS захист.

Таблиця 1.1 – Порівняльний аналіз систем захисту в іграх.

Гра	Рік	Наявність кооперативу	EAC	BE	VAC	Захист від копіювання	УПЦ	SSL/TLS
Call of Duty: Warzone [23]	2020	+	+	+	-	+	-	+
Valorant [24]	2020	+	+	-	-	+	-	+
Rainbow Six Siege [25]	2015	+	+	+	-	+	+	+
Counter-Strike: Global Offensive	2012	+	+	+	+	-	-	+
Apex Legends [26]	2019	+	+	+	-	+	-	+
The Witcher 3: Wild Hunt [27]	2015	-	-	-	-	+	+	+
Skyrim [28]	2011	-	-	-	-	+	-	-
Far Cry series [29]	2004	-	-	-	-	+	-	-
Minecraft [30]	2011	+	-	-	-	-	-	+
Red Dead Redemption 2 [31]	2018	+	-	-	-	+	+	+
Dark Souls III [32]	2016	+	-	-	-	+	+	+

Як можна побачити, реалізація систем захисту в комп'ютерних іграх істотно відрізняється. Деякі ігри демонструють надійні заходи безпеки, тоді як інші не мають достатнього комплексного захисту. Переважно ігри які не мають кооперативного режиму гри, або він не є популярним відповідно мають не достатньо надійний захист, зокрема від шахрайства. Також можна помітити, що з

роками кількість методів захисту збільшується. Як можна побачити такі ігри як Counter-Strike: Global Offensive, Call of Duty: Warzone, Apex Legends активно впроваджують нові системи захисту, незважаючи на те що їх реліз відбувся досить давно, розробники продовжують активно підтримувати та впроваджувати новий захист для таких гейм-гігантів. Ефективно використовуючи ці інструменти, розробники ігор можуть пом’якшити несанкціоновану діяльність, захистити досвід гравців і підтримувати цілісність своїх ігор.

1.4 Постановка задачі

Виходячи з результатів аналізу атак, було помічено, що переважний вектор атак спрямований на несанкціонований доступ до баз даних ігор. Дана переважаюча тенденція підкреслює серйозну проблему в ігровій індустрії, оскільки зловмисники постійно атакують ігрові дані, створюючи суттєву загрозу для користувачів зокрема на дані, які містять конфіденційну інформацію, або можуть змінити баланс гри у випадку їх несанкціонованої зміни. Ці спроби несанкціонованого доступу не лише порушують цілісність ігрових систем, а й наражають гравців на потенційне викрадення особистих даних, інші форми кіберзлочинів. Переважна кількість спроб отримання несанкціонованого доступу базується на отриманні доступу до секцій бази даних, які недоступні звичайному користувачу шляхом присвоєння шахраю ролей, які йому не належать.

Успішні атаки мають довготривалі наслідки, як для розробників, так для користувачів, завдаючи непоправної шкоди з боку довіри та репутації розробників. Також це суттєво впливає на загальне відношення потенційних користувачів до гри. За відсутності стійких засобів та методів реалізації безпеки бази даних розробники зустрічаються з циклічною боротьбою з зловмисниками, які прагнуть використати вразливості та отримати доступ до конфіденційної інформації користувачів або даних які змінюють баланс у грі.

Таким чином необхідно розробити модель розмежування прав доступу у базі даних для засобу поширення візуальних новел. Завдяки коректному розмежуванню прав доступу можна суттєво зменшити ризик спроб отримання зловмисником

несанкціонованого доступу. Також необхідно налаштувати аудит подій завдяки якому, можна відслідковувати та записувати всі дії та операції, що виконуються в системі. Це є важливим заходом безпеки, так, як це дозволяє виявляти будь-які спроби несанкціонованого доступу, підозрілі дії або потенційні порушення. Журнал аудиту може надати цінну інформацію про послідовність подій і допомогти визначити джерело або характер будь-яких інцидентів безпеки.

Крім того, регулярне резервне копіювання даних має важливе значення для забезпечення цілісності та доступності бази даних. Резервне копіювання служить запобіжним засобом від втрати даних через різні фактори, такі як апаратні збої, програмні збої, людські помилки або зловмисні атаки. Регулярно створюючи резервні копії бази даних, критично важливу інформацію можна захистити, дозволяючи швидко відновлювати її у разі будь-яких непередбачених обставин.

1.5 Висновки з розділу

Аналіз загроз комп’ютерним іграм показав, що на сьогодні індустрія розробки ігор значною мірою потерпає від шахрайських дій з боку зловмисників, націлених на отримання несанкціонованого доступу. Аналіз даних які містяться у базах даних ігор дозволив виявити основні об’єкти атак.

Огляд засобів захисту комп’ютерних ігор продемонстрував, що нині велика кількість розробників активно використовують, вбудовані засоби захисту як обфускація коду та управління цифровими правами, які зменшують ймовірність копіювання продукту та його модифікації на стороні коду. Також розробники активно використовують сторонні засоби, такі, як античіти які виконують постійний моніторнинг ігрового процесу та зменшують ризик появи шахрайських інцидентів з боку гравців.

На основі отриманих даних проведено аналіз методів захисту баз даних. Розглянуто основні стратегії захисту баз даних від несанкціонованого доступу. Даний аналіз продемонстрував, що основним методом захисту від отримання несанкціонованого доступу з боку системи управління базами даних є коректне рольове розмежування, яке зменшить ризик отримання доступу до секцій баз даних

непередбачених для конкретної ролі. На основі цього аналізу була поставлена задача комплексної бакалаврської роботи, яка зосереджена на створені захищеної бази даних шляхом створення системи розмежування ролей.

2 СТРУКТУРА БАЗИ ДАНИХ ЗАСОБУ ПОШИРЕННЯ ВІЗУАЛЬНИХ НОВЕЛ

2.1 Узагальнена архітектура засобу

Візуальні новели дедалі більше набирають популярність, як засіб інтерактивного оповідання, поєднуючи елементи літератури, мистецтва та цифрових технологій. За їхніми розповідями та візуальними ефектами ховається складна архітектурна структура, яка сприяє безперебійній взаємодії між гравцями та віртуальним ігровим світом. Серед архітектурних шаблонів, що використовуються у засобах поширення візуальних новел, було обрано клієнт-серверну модель, яка характеризується використанням двох різних баз даних: одна розташована на стороні клієнта, а друга — на стороні сервера.

Компоненти клієнта охоплюють низку важливих елементів, які сприяють візуальному відображенню та ігровому процесу в цілому, включаючи такі ключові компоненти:

– інтерфейс користувача (GUI) – клієнтська програма містить графічний інтерфейс, за допомогою якого гравці взаємодіють із засобом поширення візуальних новел;

– локальна база даних – наявна на пристрої клієнта, де зберігаються пов’язані з іграми дані, такі як збереження, налаштування та відстеження прогресу;

– HTTP-клієнт відповідає за передачу даних до ядра, які отримуються від серверу, шляхом надсилання на нього запитів. Ігрове ядро у свою чергу обробляє отримані дані від графічного інтерфейсу на клієнта.

Серверна сторона візуальної новели відіграє ключову роль у забезпеченні безперебійного ігрового процесу та взаємодії, який містить наступний набір ключових компонентів:

– база даних сервера – окрема база даних яка підтримує на стороні сервера, де зберігаються спільні дані, як-от, інформація про підписки, контент історій для гравців;

– рівень репозиторіїв надсилає запити до бази даних та формує із отриманих даних повноцінні об’єкти;

– сервісний рівень є проміжним рівнем між контролерами та репозиторіями, створений для зв’язку між ними, відповідає за логіку обробки отриманої інформації;

– контролери містять прикладні програмні інтерфейси (API) для RESTful сервіса та для архітектурного шаблону модель-представлення-контролер для взаємодії із запитами з браузера.

Таким чином, загальну архітектуру засобу можна представити у вигляді комплексної та взаємопов'язаної структурної схеми, що включає в себе вищеперераховані компоненти та модулі наведеної на рисунку 2.1.

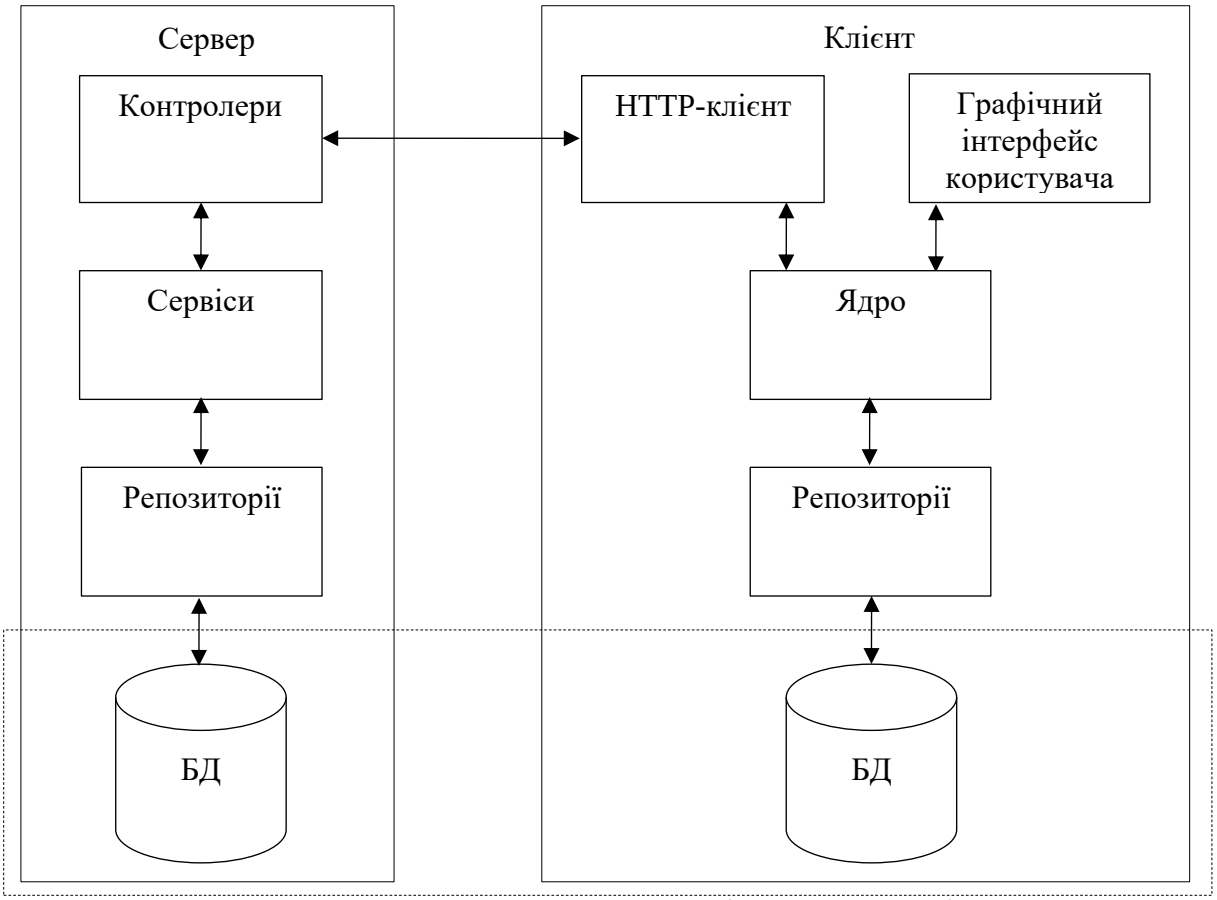


Рисунок 2.1 – Узагальнена архітектура засобу

Наведена архітектура забезпечує можливість кешування даних на стороні клієнта, що сприяє масштабованості застосунку. Крім того, наявність бази даних на стороні клієнта дозволяє зменшити навантаження на сервер.

2.2 Аналіз предметної області бази даних

З огляду загальної архітектури засобу поширення візуальних новел випливає необхідність створення одразу двох взаємопов’язаних баз даних. Необхідно визначити дані які будуть зберігатися у кожній з цих баз даних, а також зв’язки між ними. Перша база даних буде створена на стороні серверу яка містить такі дані:

– інформація про користувача, що передбачає такі дані, як облікові записи користувачів, дані для входу;

– інформація про підписку забезпечує збереження даних, пов’язаних з підписками користувачів, зокрема, такі як типи підписок, відомості про оплату та тривалість підписки;

– історії та їх вміст зокрема, діалоги, профілі персонажів, шляхи розгалуження сюжету, також тут міститимуться елементи мультимедійного відображення, такі як зображення та аудіо;

– показники для подальшого аналізу, зокрема, результати аналізу поведінки гравців, показники залученості у ту чи іншу історію та статистику ефективності для покращення візуальної новели та можливості рекомендування інших новел.

Друга база даних буде створена на стороні клієнта, її задача полягає у зменшенні навантаження на сервер та збереженні необхідних даних, які у разі переривання зв’язку забезпечать комфортну гру. База даних на стороні клієнта містить такі компоненти:

– збереження ігрового прогресу дозволяє гравцям зберігати та завантажувати свій прогрес у певних точках візуальної новели. Це включає збережені варіанти, досягнення прогресу сюжетної стрічки та іншу відповідну інформацію про стан гри;

– інформація про персоналізацію та налаштування включатиме такі параметри, як швидкість тексту, параметри аудіо, налаштування мови та візуальні параметри.

Розподіл даних між серверною та клієнтською базами даних забезпечить ефективну роботу системи та знизить навантаження на сервер. Також, це

забезпечує більшу надійність, оскільки деякі дані зберігаються локально на пристрої користувача, у разі втрати зв'язку з сервером або інших проблем.

Отже, створення двох взаємопов’язаних баз даних для засобу поширення візуальних новел є необхідним кроком, що дозволяє забезпечити ефективну роботу системи, збереження важливих даних та зручну взаємодію з користувачами.

2.3 Ідентифікація основних сутностей та їх атрибутів

Для побудови визначених баз даних, необхідно ідентифікувати основні сутності та атрибути до них. Так, як на сервері міститься основний масив даних доцільно розпочати ідентифікацію з нього. Для коректної роботи засобу на сервері мають бути присутні такий перелік основних сутностей:

- users;
- image_data;
- personages;
- edges;
- nodes.

Users створена для збереження інформації про користувача, його особистих даних, зокрема такі: нікнейм, логін, пароль, назву ролі, інформація про підписку, дозволені історії та аватар, який відображатиметься в особистому кабінеті користувача.

Image_data створена для зберігання всіх мультимедійних елементів, включаючи персонажів, сцени та інші графічні елементи. Наведена сутність містить у собі такі атрибути як ідентифікатор, назву, тип та зображення.

Засіб насичений великою кількістю різноманітних персонажів для збереження інформації про них доцільно реалізувати сутність Personages. Вона міститиме такі атрибути, як: ідентифікатор, назву, емоцію та сторону у якій персонаж розташовується відносно сцени.

Для визначення вмісту сцени яка відображається користувачу створена сутність Nodes вона оперує тим, який персонаж, фон, текст буде розташований на

екрані та варіанти відповідей користувача під час діалогу. Сутність Edges відповідає за сценарій, визначаючи черговість сцен, які відображаються під час проходження сюжету. Вона містить такі атрибути, як ідентифікатор, тип, інформацію про попередню та наступну сцену. Розроблені сутності з визначеними атрибутами наведено на рисунку 2.2.

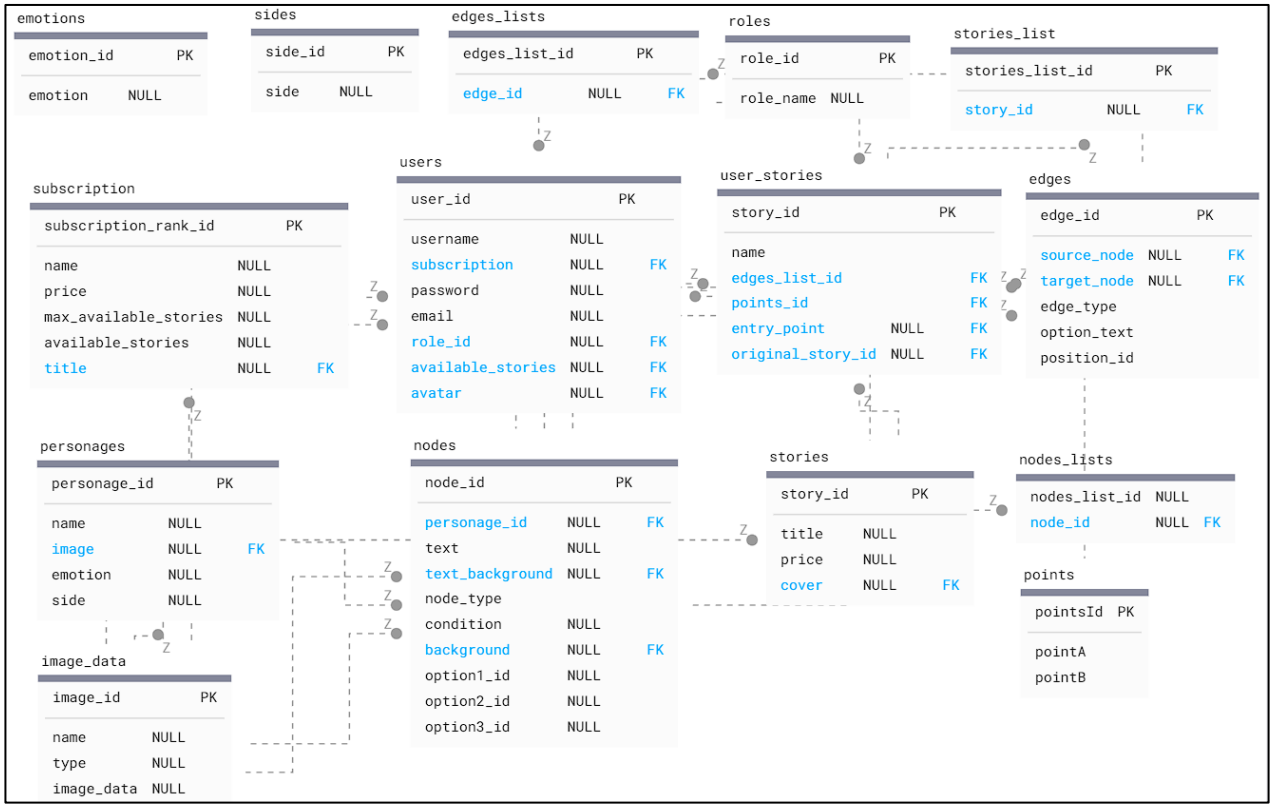


Рисунок 2.2 – Сутності та атрибути бази даних сервера

База даних на стороні клієнта містить значно меншу кількість сутностей, зокрема: user, image_data, settings. User міститиме в собі інформацію про користувача, та його аватар. Images_data виконуватиме зберігатиме по аналогії до серверної бази даних, мультимедійні дані. Це забезпечить безперервну гру навіть у випадку переривання зв’язку з сервером. Settings міститиме в собі дані про персональні налаштування користувача, для того, щоб не налаштовувати засіб кожного разу після запуску. Реалізовані сутності також можна відобразити у вигляді схеми наведеної на рисунку. 2.3.

Рисунок 2.3 – Структура даних бази даних клієнта

Основна частина даних зберігається на сервері. На стороні клієнта обробляється мала частка даних, які відповідають здебільшого за персоналізацію користувача та його особисті налаштування.

2.4 Нормалізація бази даних

Нормалізація відіграє вирішальну роль у проектуванні бази даних, забезпечуючи ефективне зберігання даних, покращені показники захисту цілісність даних і спрощене обслуговування бази даних. Дотримання принципів нормалізації, дозволяє створювати надійні та масштабовані бази даних, які відповідають потребам програми та її користувачів. Враховуючи масштаб бази даних сервера, доцільно провести нормалізацію саме для неї.

Сутності початкової бази даних містять у собі велику кількість атрибутів, які доцільно розділити на декілька таблиць. Сегмент Users ненормалізованої бази даних має вигляд, наведений на рисунку 2.4.

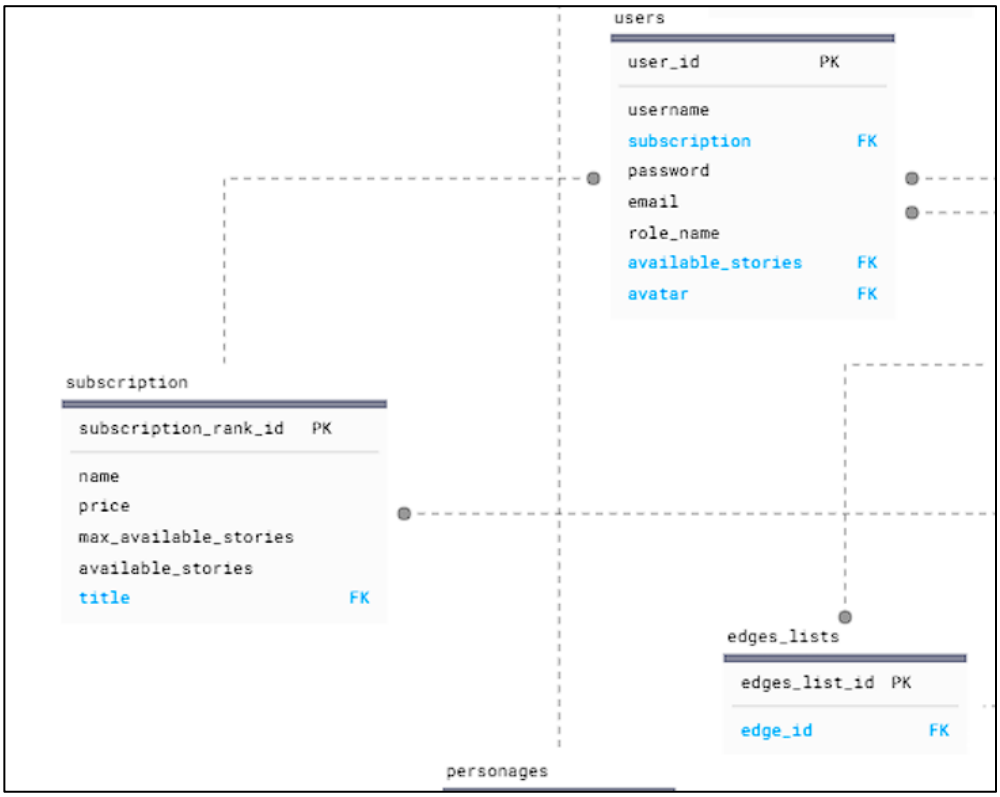


Рисунок 2.4 – Сегмент Users бази даних

Таблиця Users буде розділена на таблиці з інформацією про підписки користувача, ролі, таблицю у якій будуть міститись дані про дозволені до перегляду історії та таблицю з конкретною інформацією про історію та інші, які необхідні відповідно до вимог засобу поширення візуальних новел. Також необхідно встановити відповідні зв’язки між атрибутами у таблицях. Вигляд нормалізованого сегменту Users з встановленими зв’язками наведена на рисунку 2.5.

Таким чином усі таблиці мають первинний ключ, кожен стовпець у таблиці містить атомарні значення, жодна таблиця не містить повторів груп або масивів таким чином була досягнута перша нормальна форма. Таблиці не мають часткових залежностей, стовпці в кожній таблиці залежать від усього первинного ключа, а не лише від його частини за рахунок цього сегмент бази даних був нормалізований до другої нормальної форми. Ні в одній таблиці немає транзитивних залежностей та неключові стовпці залежать лише від первинного ключа, а не від інших неключових стовпців так сегмент бази даних був нормалізований до третьої нормальної форми, що відповідає вимогам засобу для поширення візуальних новел.

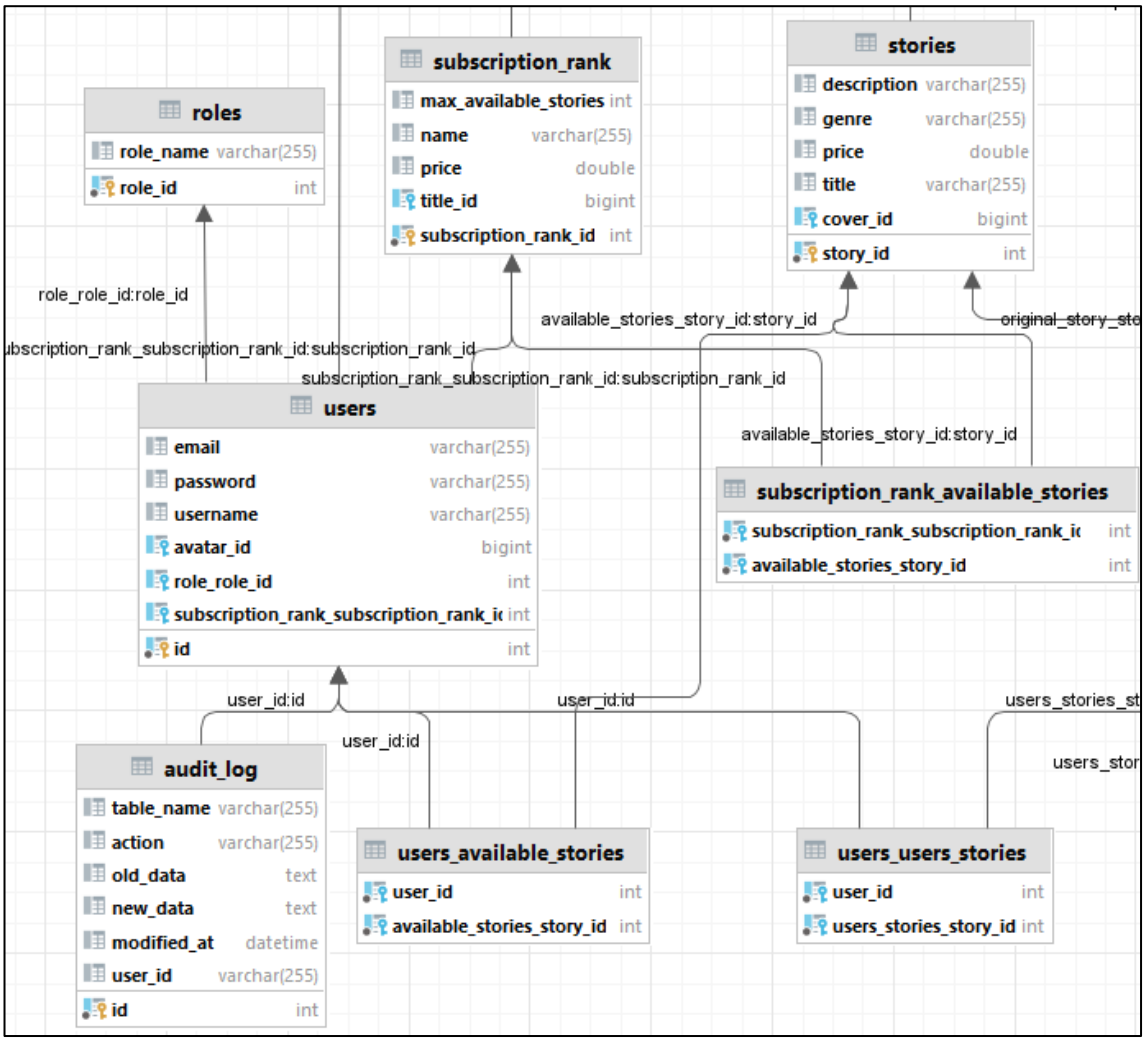


Рисунок 2.5 – Нормалізований сегмент Users бази даних

Наступним сегментом для нормалізації був сегмент Nodes який визначає основні елементи сцени яку спостерігає користувач. Сегмент Nodes ненормалізованої бази даних має вигляд, наведений на рисунку 2.6.

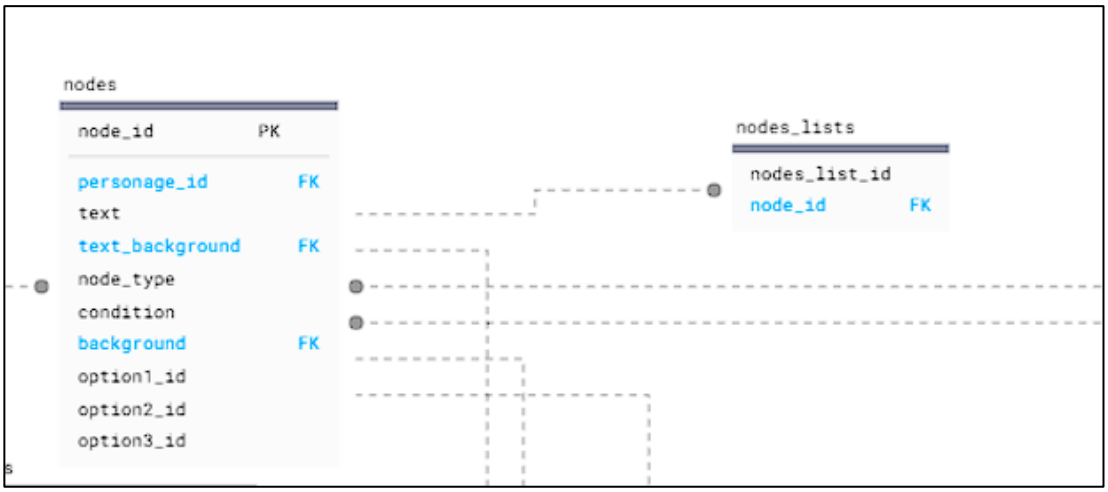


Рисунок 2.6 – сегмент Nodes бази даних

Аналогічно до сегменту Users, його було модифіковано. Даний сегмент зберігає інформацію про вміст сцени. Нормалізований сегмент був розділений на такі таблиці, як nodes для збереження інформації про вміст конкретної сцени, conditions, у якій буде збережена інформація про умову, three_choice_node та two_choice_node у яких зберігається інформація про варіативність відповідей у діалогах та інші відповідно до потреб засобу. Вигляд нормалізованого сегменту Nodes з встановленими зв'язками має вигляд, представлений на рисунку 2.7.

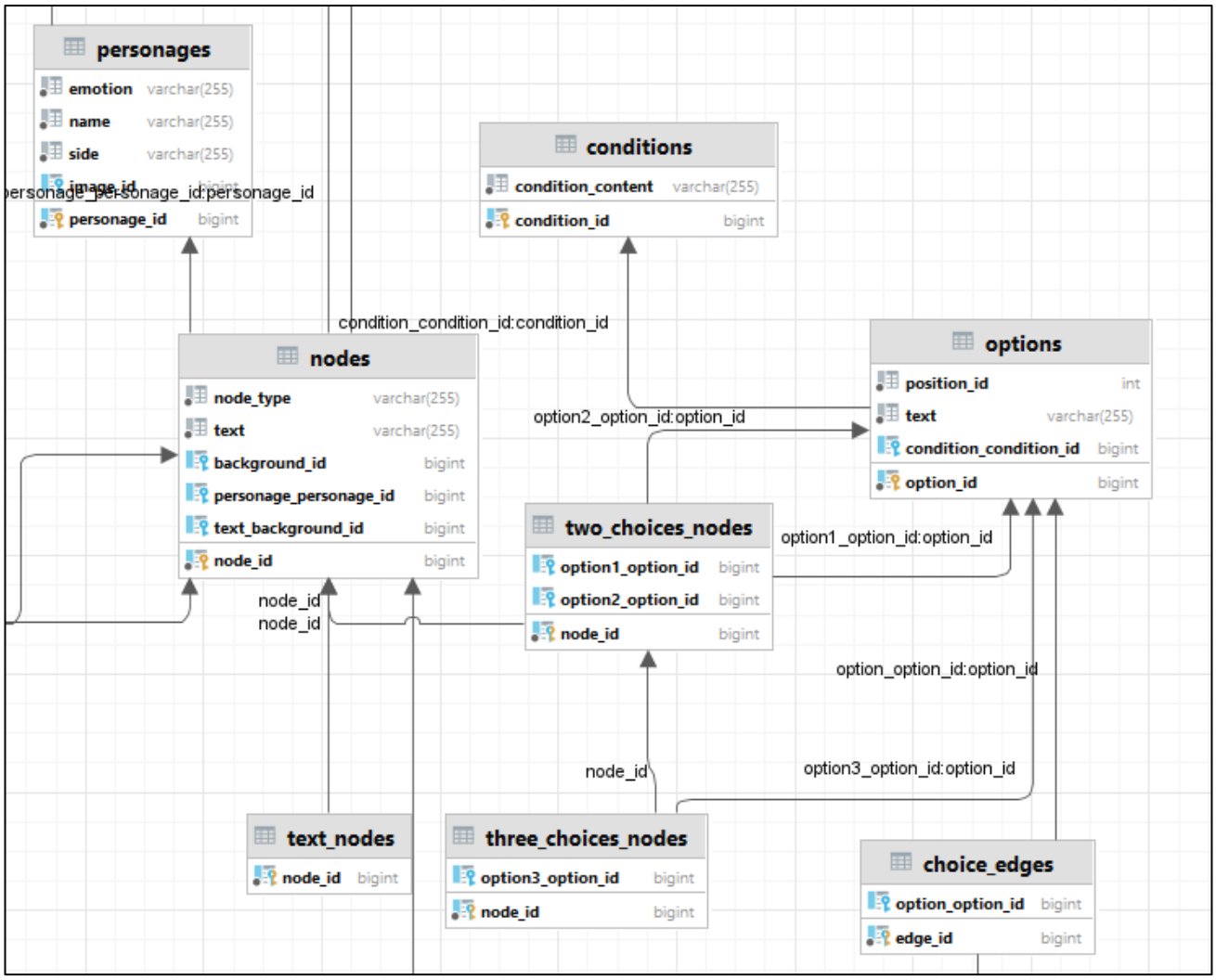


Рисунок 2.7 – Нормалізований сегмент бази даних вмісту сцен Nodes

Сегмент Edges відповідає за порядок появи сцен, визначаючи попередню та наступну сцену. Як і сегмент Nodes, він відіграє ключову роль у правильності відображення сцени. Ненормалізований сегмент Edges такий вигляд, наведений на рисунку 2.8.

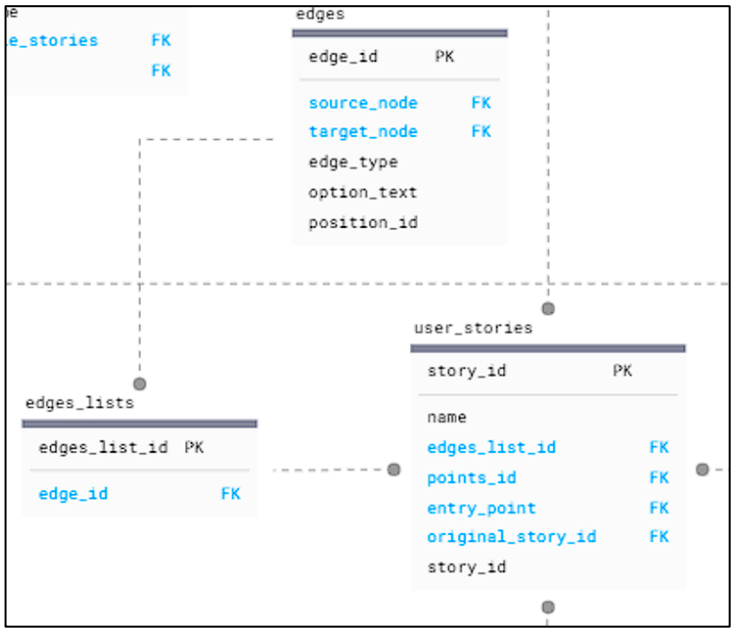


Рисунок 2.8 – сегмент Edges бази даних

Відповідно до вимог засобу поширення візуальних новел його було розділено на відповідні таблиці, налаштовано зв’язки та виконано аналогічні дії, що й до попередніх сегментів для досягнення третьої нормальної форми, внаслідок чого цей сегмент набув нового вигляду на рисунку 2.9.

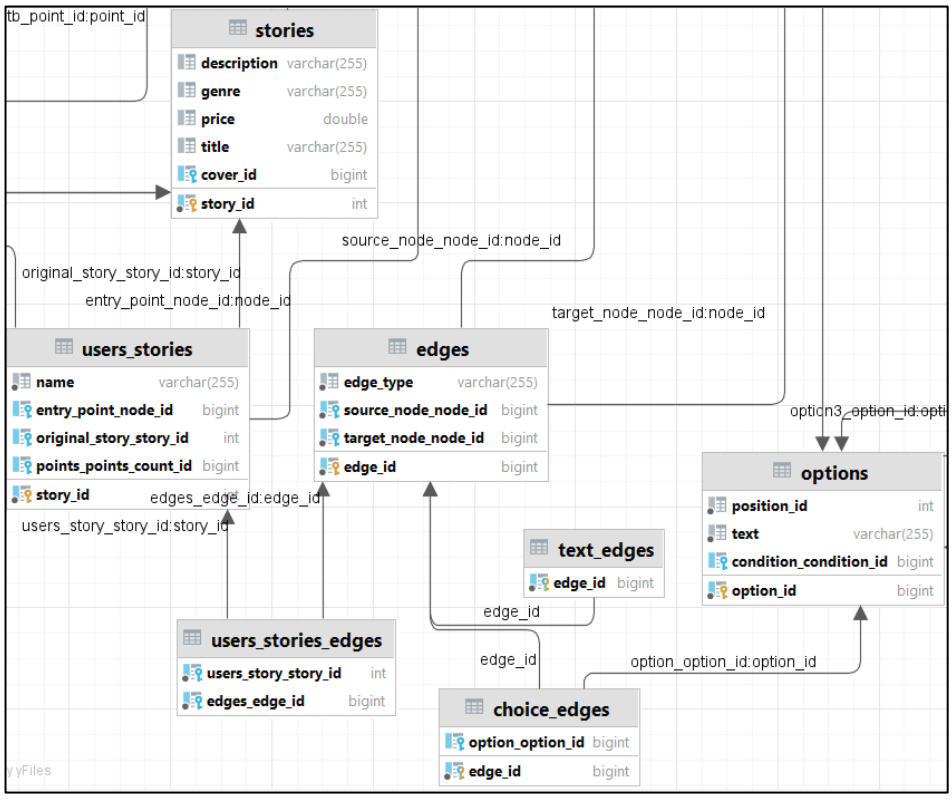


Рисунок 2.9 – Нормалізований сегмент Edges бази даних

Таким чином було проведено нормалізацію бази даних, інші сегменти були аналогічно нормалізовані до третьої нормальної форми. ER-діаграма нормалізованої бази даних продемонстрована в додатку А.

2.5 Висновки з розділу

Під час розробки структури бази даних засобу поширення візуальних новел, була розглянута узагальнена архітектура засобу, що дозволило обґрунтувати доцільність використання клієнт-серверної архітектури для даного застосунку та захищеної бази даних зокрема. Проаналізовано основні елементи, які використовуються в засобі, та побудовано його структурну схему. В результаті виконаного аналізу предметної області було визначено основні дані, які будуть зберігатись на стороні клієнта та сервера.

На основі отриманих даних було ідентифіковано сутності та атрибути, які використовуються у базі даних, що дозволило виконати декомпозицію складності задачі її проєктування. Шляхом встановлення зв’язків між сутностями було розроблено базу даних для сервера та клієнта. Отримана база даних була нормалізована до третьої нормальної форми, що дозволило зменшити ймовірність виникнення аномалій бази даних, а відтак збільшити захист цілісності даних візуальних новел, які поширюються з використанням засобу, який розробляється в межах цієї комплексної бакалаврської дипломної роботи.

3 МЕТОДИ ЗАХИСТУ БАЗИ ДАНИХ

3.1 Узагальнений алгоритм роботи захищеної бази даних

Узагальнений алгоритм роботи захищеної бази даних включає ряд кроків, які допомагають забезпечити високий рівень безпеки та захисту даних. Ці кроки включають перевірку автентифікації, авторизацію, та логування дій які були внесені у базу даних. Перевірки створені на шляхом рольового розмежування, тригерів та процедур які в сукупності реалізують комплексний захист бази даних. Алгоритм роботи сегменту автентифікації захищеної бази даних наведено на рисунку3.1.

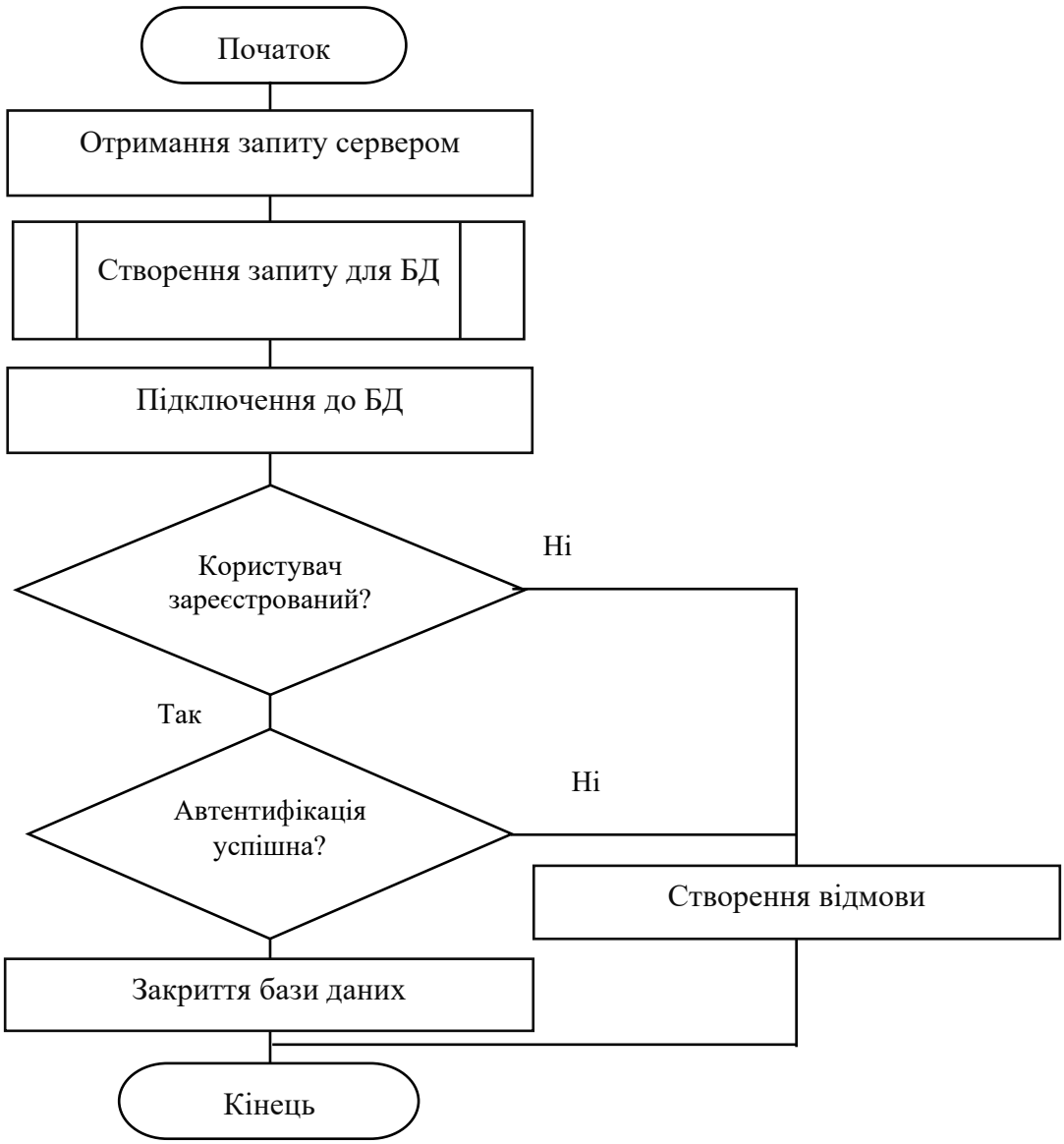


Рисунок 3.1 – Алгоритм роботи сегменту автентифікації захищеної бази даних

Таким чином користувач надсилає на сервер запит, сервер у свою чергу формує запит для бази даних та з’єднується з нею, після чого відбувається перевірка чи існує користувач. У випадку позитивного результату відбувається перевірка автентифікації. Наступним кроком здійснюється перевірка чи може користувач виконувати дії, які містяться у запиті алгоритм роботи сегменту перевірки доступу та логування наведено на рисунку 3.2.

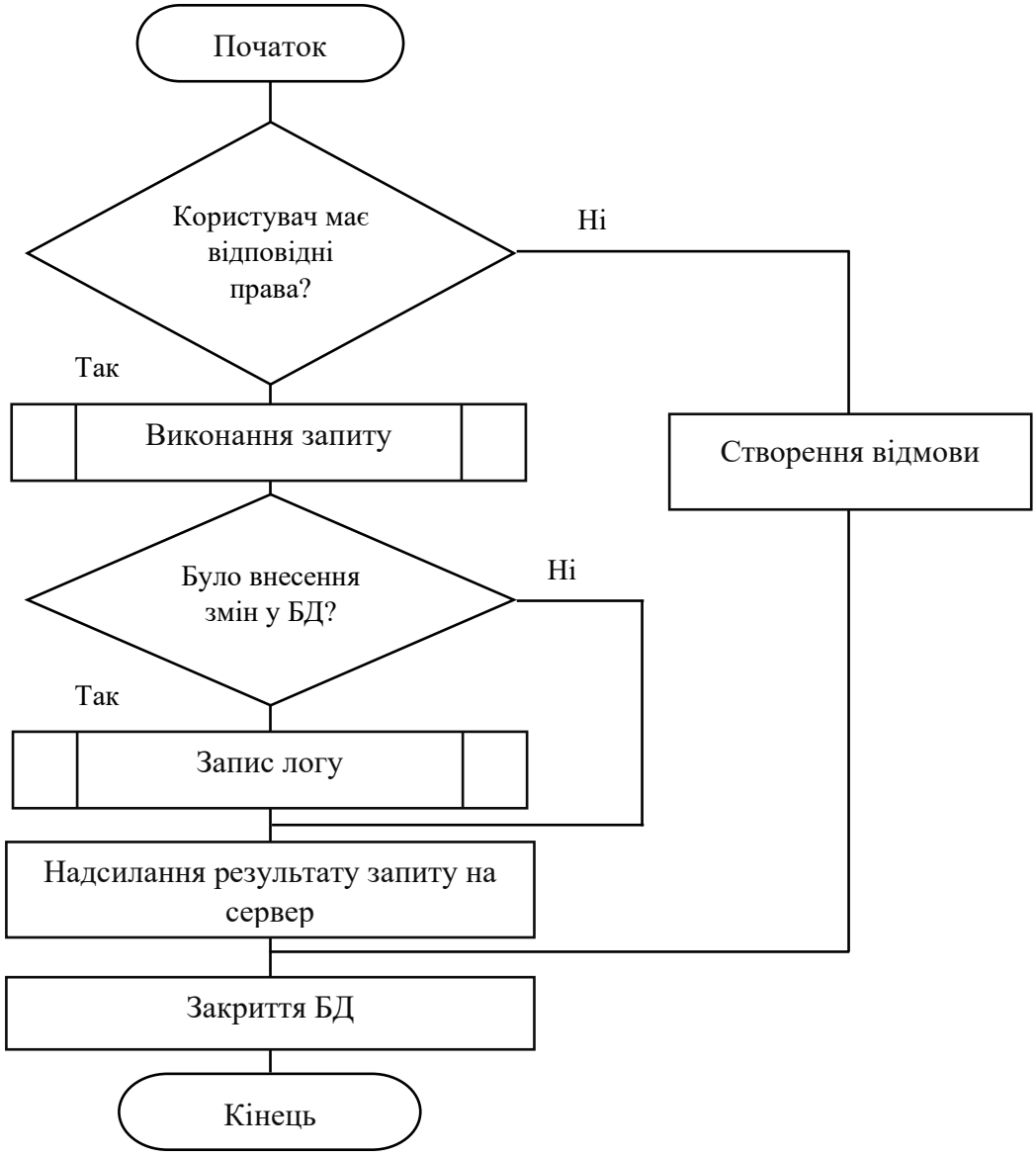


Рисунок 3.2 – Алгоритм роботи сегменту перевірки доступу та логування захищеної бази даних

Якщо на попередніх етапах результат перевірки є позитивним, відбувається перевірка чи може користувач виконати запит відповідно до його ролі у базі даних. Якщо результат також позитивний відбувається виконання запиту, у випадку

негативного результату, як і в попередніх перевірках, буде сформовано відмову та надіслано на сервер. Остання перевірка – перевірка на зміну у базі даних, якщо вона відбулась, відповідний тригер здійснить запис зміни до журналу дій, у випадку якщо зміни не було внесені, відбудеться надсилання результату запиту на сервер.

3.2 Метод розмежування прав доступу до бази даних

Доступ до бази даних відіграє важливу роль в управлінні та організації величезних обсягів даних, пов’язаних із засобом поширення візуальних новел. Для баз даних представленого засобу гарним рішенням є рольове розмежування прав доступу. Система рольового розмежування активно застосовується в ігрових застосунках, де передбачені особливі привілеї та платний контент.

Для користувачів, які мають доступ до бази даних з боку розробників кожна роль матиме певний обсяг дій, які вони можуть виконувати під час користування базою даних, гарантуючи, що лише авторизовані особи зможуть отримати доступ до конфіденційної інформації гравців, вносити зміни в ігрову механіку або здійснювати вплив на віртуальну економіку гри. Визначення ролей для розробників доцільно представити у вигляді таблиці 3.1.

Таблиця 3.1 – Рольове розмежування для розробників

Роль	Опис	Дозволи	Заборони
SceneEditor	Має повний доступ до бази даних.	UPDATE, INSERT, DELETE (nodes,edges,image_data)	N/A
TextWriteEditor	Можна змінювати лише текстові дані.	UPDATE, INSERT (nodes)	N/A
ImageEditor	Можна змінювати лише графічні дані.	UPDATE, INSERT, DELETE (image_data)	N/A
Viewer	Може читати дані в усіх таблицях	SELECT (All tables)	UPDATE, INSERT, DELETE (Всі таблиці)

У засобі для поширення візуальних новел міститься платний контент, зокрема система рангів відповідно до якої, кожен ранг має визначену кількість доступних

історій з категорії платного контенту, таким чином рольове розмежування необхідно впроваджувати не лише з боку розробників, а й для користувачів відповідно до привілеї яку вони придбали. Користувачі різних рангів наслідуватимуть можливості користувача з максимальною кількістю привілеїв, але матимуть персональні обмеження. Визначення ролей для аналогічно представлено у вигляді таблиці 3.2.

Таблиця 3.2 – Розмежування прав доступу для користувачів

Роль	Опис	Дозволи	Заборони
Platinum	Може читати всі безкоштовні та платні історії	SELECT	N/A
V.I.P	Може читати всі безкоштовні історії, а також 6 додаткових платних історій, має отримує персональну підтримку	SELECT	Доступ до даних ролі id 6
Premium+	Можна читати всі безкоштовні історії, а також 2 додаткові платні історії на вибір	SELECT	Доступ до даних ролей id 5,6
Premium	Можна читати всі безкоштовні історії, а також 1 додаткова платна історія	SELECT	Доступ до даних ролей id 4-6
Gamer	Можна читати всі безкоштовні історії	SELECT	Доступ до даних ролей id 3-6
Guest	Має доступ лише до обмеженого вибору безкоштовних історій і функцій і не може залишати коментарі чи оцінки	SELECT	Доступ до даних ролей role id 2-6

Такий детальний контроль над розмежуванням доступу суттєво зменшує ймовірність використання вразливості неавторизованими користувачами та отримання несанкціонованого доступу до обмежених для звичайних користувачів даних. Застосування методу розмежування прав доступу допомагає забезпечити

конфіденційність, цілісність та доступність даних. Адміністратори можуть встановлювати різні рівні доступу для різних груп користувачів, забезпечуючи відповідність до принципів необхідності та обмеження прав доступу.

Доступ до контенту користувачі отримуватимуть за рахунок присвоєння їм ідентифікатора ролі, який визначатиме до якого рангу відноситься користувач. Розмежування ролей буде реалізовано на програмному рівні засобу поширення візуальних новел. Відповідно, якщо користувач має роль «Premium» у таблиці users, якій відповідає ідентифікатор під номером 4 у таблиці subscription_rank, йому буде доступний контент передбачений списком історій який визначається атрибутом max-avaible_stories з таблиці subscription_rank.

Таким чином метод розмежування прав доступу до бази даних засобу буде реалізовуватись за допомогою такої послідовності кроків:

Крок 1. Введення користувачем логіна та пароля.

Крок 2. Автентифікація користувача за допомогою системи керування баз даних. Якщо роль належить до ролей розробників, то завершити розмежування прав доступу на основі розподілу прав, наведених в таблиці 3.1.

Крок 3. На основі прав користувачів, наведених в таблиці 3.2, передати запит до розмежування прав доступу серверній частині застосунку поширення візуальних новел.

Розмежування доступу до контенту таким чином, виключить потребу використання декількох баз даних для кожної історії, що зменшить розміри, обсяг навантаження на сервер, покращить гнучкість розмежування і гранулювання прав доступу користувачів. Останнє є критичним для спрощення адміністрування засобу для поширення візуальних новел, що зменшить вплив людського фактору, а відтак ризики помилок у некоректному розмежуванні прав доступу, що наразі є найбільшою загрозою кібербезпеці відповідно до рейтингу OWASP top 10 [33].

3.3 Алгоритм захисту даних на стороні сервера

Алгоритм захисту даних на стороні сервера включає в себе різні методи та практики, спрямовані на забезпечення безпеки даних, що зберігаються у базі даних на сервері. Основними видами захисту є використання тригерів та процедур, які мають різноманітні функції, такі як журналювання та аудит, шифрування даних та заборона дублювання даних.

Журналювання та аудит дій користувачів здійснюються за допомогою набору тригерів. Цей підхід дозволяє відстежувати та записувати всі дії, виконані над даними, такі як вставка, оновлення або видалення записів. Журнал подій містить інформацію про дії користувачів, що дозволяє виявляти потенційні зловживання або порушення безпеки. Наприклад, при спробі недозволеної зміни даних або доступу до обмежених ресурсів система може засікти цю подію і повідомити відповідним адміністраторам. Загальний алгоритм роботи тригера наведено на рисунку 3.3.

Рисунок 3.3 – Загальний алгоритм роботи тригера

Шифрування даних є ще одним важливим аспектом алгоритму захисту. Процедури шифрування використовуються для захисту конфіденційної інформації перед збереженням у базі даних. Шифрування може бути застосоване до окремих полів або всієї таблиці, залежно від вимог до безпеки. Зашифровані дані неможливо прочитати без відповідного ключа або пароля, що забезпечує конфіденційність інформації навіть в разі несанкціонованого доступу до бази даних.Загальний алгоритм роботи процедури наведено на рисунку 3.4.

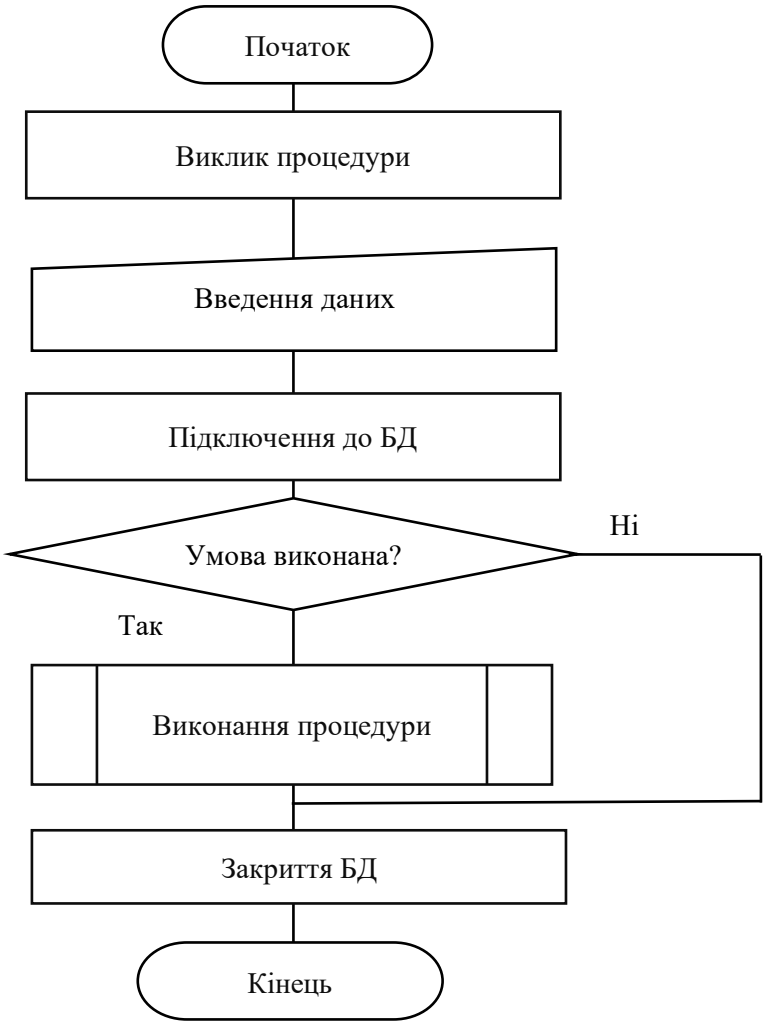


Рисунок 3.4 – Загальний алгоритм роботи процедури

Заборона дублювання даних є іншою важливою функцією алгоритму захисту. Вона реалізовується за допомогою тригера, який перевіряє дані, що вставляються або оновлюються у відповідних таблицях, і запобігає створенню однакових записів. Це забезпечує цілісність даних і унікальність записів, що може бути важливим для забезпечення точності та надійності інформації.

3.4 Алгоритм роботи тригерів та процедур

Для підвищення безпеки бази даних було розроблено низку тригерів та процедур. Процедура RegisterUser відповідає за реєстрацію користувачів та внесенню їх у таблицю users. Дана процедура працює за алгоритмом, наведеним на рисунку 3.5.

Рисунок 3.5 – Алгоритм процедури RegisterUser

Користувач вводить ім’я, пошту та пароль, після чого процедура гешує пароль, з’єднується з базою даних та записує їх у таблицю збільшуючи тим самим швидкість введення даних до таблиці та зменшуючи ризик введення даних у некоректному вигляді або послідовності.

Процедура автентифікації AuthenticateUser здійснює перевірку чи коректно введені дані користувача порівнюючи їх з введеними раніше даними та працює за алгоритмом, наведеним на рисунку 3.6.

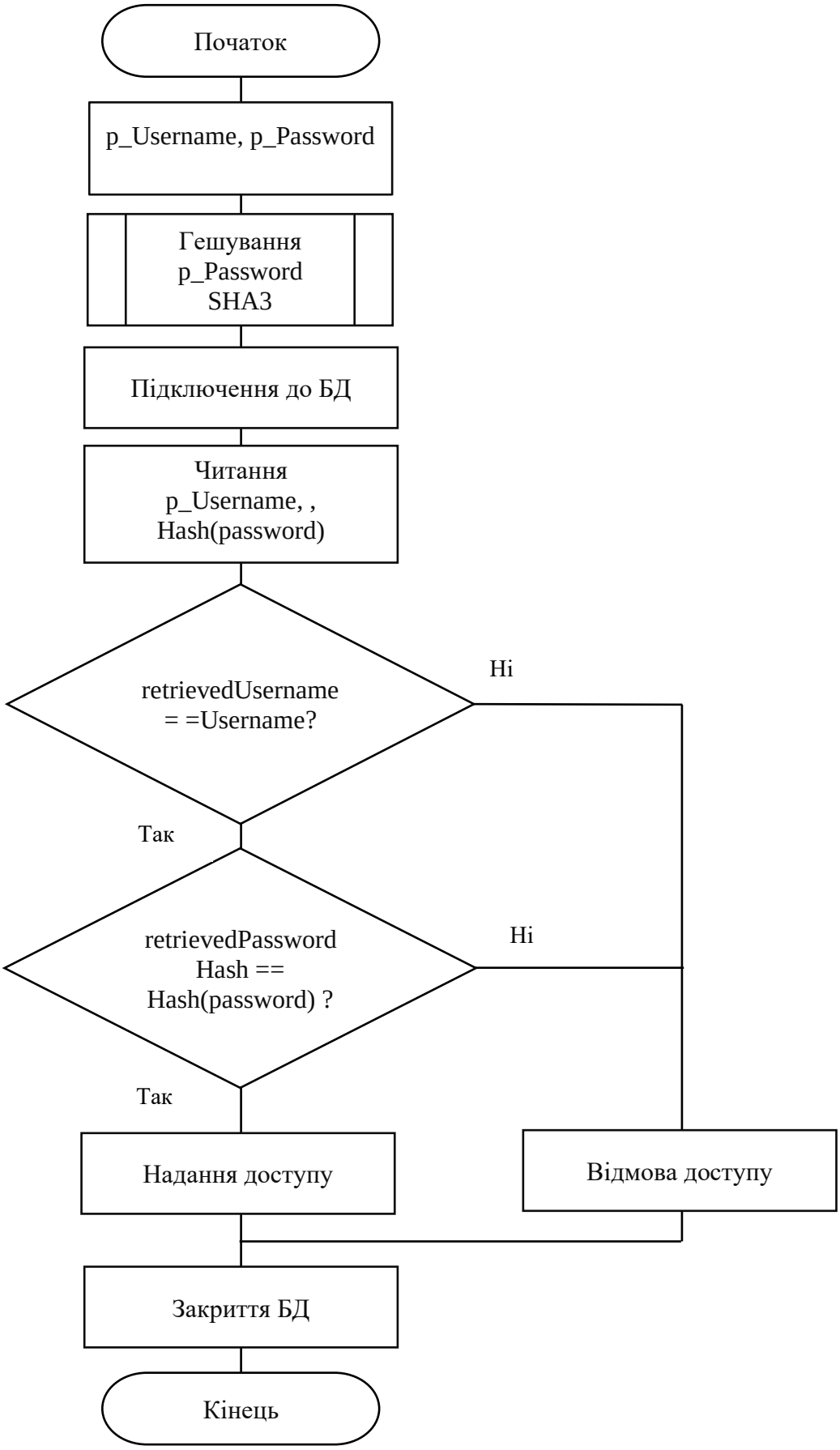


Рисунок 3.6 – Алгоритм процедури AuthenticateUser

Користувач вводить дані для авторизації, процедура гешує введений пароль після чого з’єднується з базою даних та перевіряє чи введене ім’я користувача відповідає зазначеному в таблиці users, у випадку позитивного результату, відбувається перевірка гешів паролю з таблиці, та введеного паролю користувача, якщо на цьому етапі перевірки паролі збігаються, користувачу надається доступ, у випадку неправильного введення паролю або ім’я процедура виведе на екран відповідне повідомлення.

Тригер PreventDuplicateUsers перешкоджатиме введенню користувачів з однаковим іменем та поштою, у разі спроби їх введення він унеможливить введення дублюючих даних у таблицю. Алгоритм роботи тригера має такий вигляд, наведений на рисунку 3.7.

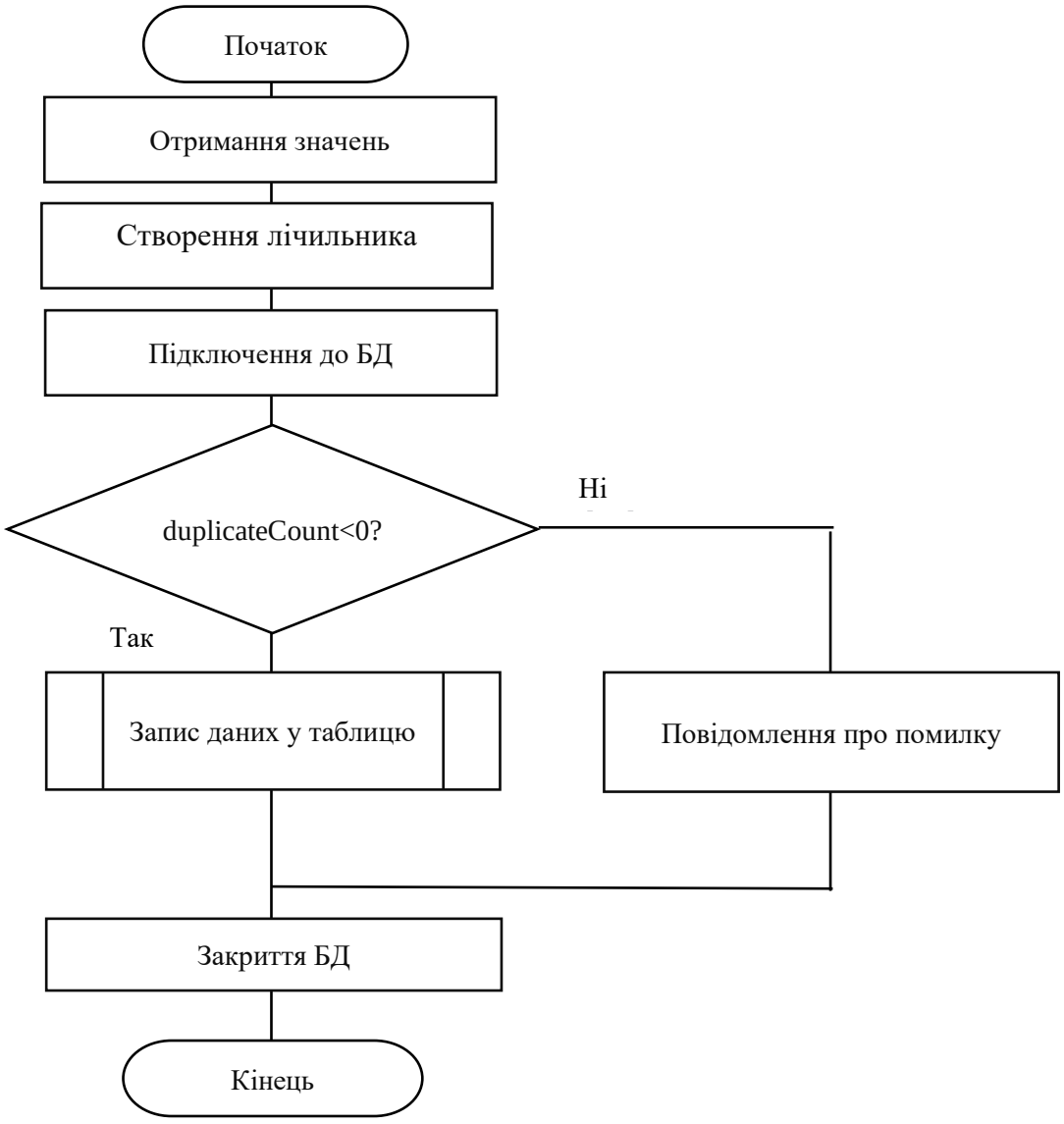


Рисунок 3.7 – Алгоритм роботи процедури PreventDuplicateUsers

Тригер отримує нові значення які вставляються в таблицю users після чого, відбувається оголошення лічильника duplicateCount який зберігатиме кількість повторюваних користувачів, після з’єднання з базою даних підраховується кількість користувачів з однаковим іменем та поштою, якщо при перевірці значення лічильника дорівнює нулю, відбувається введеня даних у таблицю, у випадку якщо значення лічильника дорівнює або більше за одиницю, виводиться повідомлення про дублікат даних, як наслідок введення не відбувається. Даний тригер забезпечує захист від компрометації та появи непотрібних даних.

Серія тригерів audit_trigger створена для аудиту та логування даних які вносяться у базу даних в окрему таблицю audit_log з даними про таблицю у якій були здійснені зміни, дані які були до зміни, дані які стали після зміни, час внесення змін та користувача, який ввів ці зміни. Для кожної таблиці які є критично важливими для збереження цілісності бази даних створений окремий тригер. Алгоритм роботи тригера продемонстровано на прикладі логування дій у таблиці Edge та має такий вигляд, представлений на рисунку 3.8.

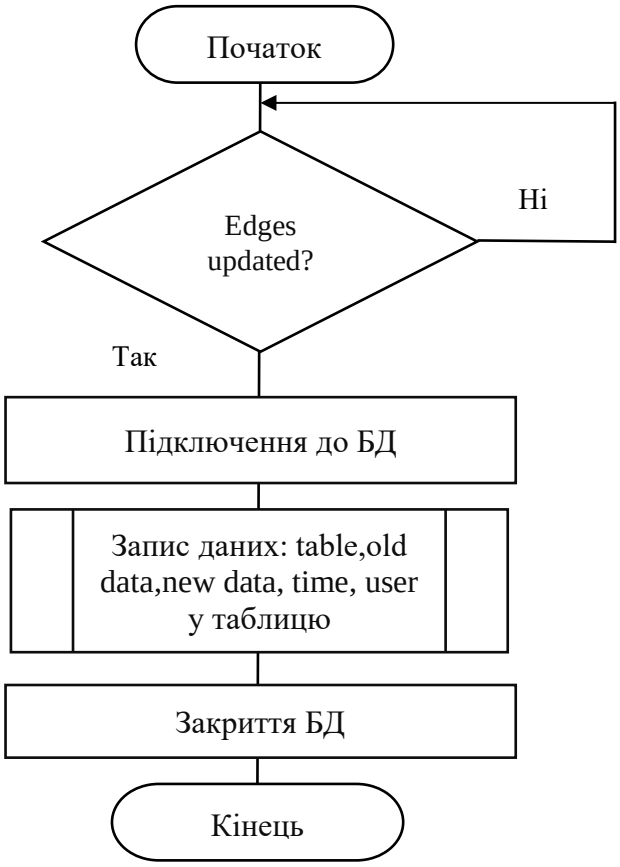


Рисунок 3.8 – Алгоритм роботи процедури edge_audit_trigger

Коли в таблиці виконується оновлення edges, запускається тригер. Тригер перевіряє, чи оновлено будь-який із зазначених стовпців (edge_type, source_node_node_id, target_node_node_id). Якщо будь-який із зазначених стовпців оновлено, тригер переходить до створення запису аудиту. Вставляється новий запис у таблицю audit_log з відповідними даними, включаючи назву таблиці (edges), дію (update), old_data (об’єкт JSON, що містить старі значення), new_data (об’єкт JSON, що містить нові значення), modified_at (об’єкт JSON, що містить нові значення), поточна позначка часу та user_id (ідентифікатор поточного користувача).

Використання тригерів для логування збільшує захищеність та цілісність даних, збільшуючи швидкість знаходження змінених даних та користувача яким вони були внесені. Застосування такого тригеру потребує використання процедури яка полегшить пошук у таблиці audit_log відповідно здійснюючи пошук по таблиці у якій було проведено зміни. Алгоритми роботи тригерів для решти таблиць та процедура яка відповідає за пошук даних по таблицям наведені у додатку Б.

3.5 Висновки з розділу

У даному розділі були розглянуті різні аспекти захисту баз даних. Було представлено узагальнений алгоритм роботи захищеної бази даних, який включає перевірки автентифікації, авторизації, відповідності ролі та логування. Для підвищення рівня безпеки даних, було описано метод розмежування прав доступу до бази даних, що включає в себе розподіл ролей і привілеїв між користувачами та розробниками, які здійснюють налаштування та адміністрування бази даних. Цей метод дозволяє контролювати рівні доступу користувачів до різних об'єктів бази даних в залежності від їхніх ролей та відповідних прав знижуючи загрозу отримання несанкціонованого доступу до бази даних. Особливістю методу є те, що частина задачі розмежування доступу контролюється системою керування базою даних, а частина — програмним засобом. Таке застосування дозволяє подолати обмеження систем керування базами даних щодо гранулювання прав доступу до конкретних записів в межах таблиць та водночас дозволяє забезпечити додатковий

шар захисту для ролей користувачів, зробивши систему керування правами доступу багатошаровою для користувачів, які несуть найбільші ризики, що реалізує один з найкращих підходів до розмежування прав доступу.

Також були представлені алгоритми роботи тригерів та процедур, що допомагають забезпечити відповідну обробку даних на серверному рівні. Вони здійснюють авторизацію, перевірку автентифікації, дублювання, логування критично важливих для бази даних таблиць. Також реалізовано декілька процедур, які пришвидшують введення та пошук даних у базі даних.

Використання цих підходів, алгоритмів та методів допомагає забезпечити ефективний та комплексний захист баз даних. Це включає контроль доступу до інформації, управління привілеями та забезпечення відповідного функціонування бази даних відповідно до встановлених вимог.

4 ТЕСТУВАННЯ ЗАХИЩЕНОЇ БАЗИ ДАНИХ

4.1 Обґрунтування засобів розробки

У сфері систем керування базами даних розробникам і адміністраторам потрібні ефективні інструменти для взаємодії з базами даних і керування ними. На основі виконаного в першому розділі аналізу, випливає, що вибір засобу розробки повинен базуватись на вимогах, які мають задовольняти такі показники якості:

- продуктивність;
- масштабованість;
- широкий спектр функцій;
- зручний інтерфейс.

Під час огляду найпоширеніших інструментів розробки та управління базами даних, для порівняння, було обрано, SQL Server Management Studio, Oracle SQL Developer, SQLite та MySQL Workbench. Аналіз цих інструментів показав, що MySQL Workbench демонструє високу масштабованість, що є важливим аспектом для розвитку баз даних. Він здатний працювати з великим обсягом даних і ефективно опрацьовувати складні запити [34]. Цей інструмент дозволяє розробникам і адміністраторам легко масштабувати базу даних, додавати нові об'єкти, розширювати функціонал та забезпечувати швидкий доступ до інформації. Більше того, він підтримує можливості реплікації та кластеризації, що дозволяє розробникам працювати з розподіленими системами баз даних та забезпечувати цілісність та доступність даних. Також MySQL Workbench надає розширені можливості управління користувачами та правами доступу до бази даних. Розробники можуть легко налаштувати доступ до різних об'єктів бази даних та забезпечити безпеку даних. У SQLite такі рівні контролю доступу можуть бути обмежені або відсутні [35].

У порівнянні з MySQL Workbench, варто зазначити, що Oracle та SQL Server Management Studio є платними інструментами [37]. Використання цих інструментів

може бути недоцільним з економічної точки зору, оскільки вони вимагають плати за ліцензію [38]. У порівнянні з ними, MySQL Workbench є безкоштовним інструментом, що надає широкі можливості розробки та управління базами даних, що задовольняють потреби ефективності, масштабованості та зручного інтерфейсу для розробників та адміністраторів.

Одна з основних сильних сторін полягає в його надійних можливостях надсилання запитів. Завдяки потужному редактору SQL можна писати складні запити, виконувати їх і переглядати результати в різних форматах. MySQL Workbench підтримує підсвічування синтаксису, фрагменти коду, покращуючи читабельність коду та продуктивність

Також даний інструмент має інтуїтивно зрозумілий, зручний, швидкий та інтерактивний інтерфейс дозволяє розробникам ефективно взаємодіяти з базою даних, мінімізуючи затримки та покращуючи загальний час розробки. Наявність розширених інструментів налагодження та профілювання допомагає визначити й оптимізувати продуктивність запитів.

Окрім того MySQL Workbench пропонує можливості з підвищення продуктивності та масштабованості баз даних. Його оптимізований інтерфейс, інструменти для профілювання та налагодження запитів, а також можливості розподілу даних дозволяють розробникам ефективно взаємодіяти з базами даних і забезпечувати швидкий доступ до інформації. В результаті MySQL Workbench є потужним інструментом для розробки та управління базами даних, що задовольняє вимоги щодо продуктивності та масштабованості у сфері баз даних та є ідеальним інструментом для розробки та адміністрування бази даних для засобу поширення візуальних новел.

4.2 Реалізація процедур захисту мовою SQL

Реалізація процедур захисту мовою SQL є невід'ємною частиною безпеки баз даних. Мова SQL є потужним інструментом для управління базами даних і надає вбудовану підтримку для створення ролей, тригерів, процедур та контролю доменної цілісності, що допомагає забезпечити безпеку та захист інформації.

Створення ролей є ефективним методом управління доступом до бази даних. Ролі групують користувачів і надають їм права доступу до об'єктів бази даних. Наприклад, роль "Admin" може мати повний доступ до всіх таблиць та процедур, тоді як роль "Image_editor" може мати обмежений доступ лише до певних об'єктів. Це дозволяє обмежити несанкціонований доступ до конфіденційної інформації та підвищує рівень безпеки бази даних.

Тригери використовуються для автоматичного виконання дій або перевірок при певних подіях в базі даних. Вони можуть викликатись при вставці, оновленні або видаленні даних з таблиць і дозволяють реалізувати додаткову перевірку цілісності даних або виконати специфічні дії. Для захищеної бази даних було розроблено низку тригерів, які відповідають за внесення даних про зміну, введення та видалення даних з таблиць: users, nodes, edges та image_data. У випадку спроби їх зміни тригер зафіксує тип, час, користувача, а також попередні та оновлені дані. Це дозволить у разі виникнення помилок у базі даних швидко відслідкувати таблиці, які піддались змінам та повернути їх дані у попередній стан. Також для попередження введення дублюючих даних був створений тригер, який не дозволить додати два гравця, з однаковою поштою та ім’ям в таблицю users. Це знизить загрозу компрометації даних, а також додавання великої кількості однакових акаунтів які збільшать вміст бази даних.

Процедури є важливими елементами для розподілення пов'язаних дій в базі даних. Вони дозволяють зберігати набір інструкцій і викликати їх з інших частин програми. Процедури можуть використовуватись для перевірки коректності даних, виконання складних обчислень або операцій над даними. Крім того, вони забезпечують зручний інтерфейс для доступу до функціональності бази даних і зменшують необхідність в повторюваному коді. Розроблені в базі даних процедури відповідають за реєстрацію користувача, яка відбувається шляхом автоматичного введення в таблицю users ім’я, пошти та паролю, який був введений користувачем. Пароль який вводить користувач гешується та тільки після цього додається до таблиці. Також реалізована процедура, яка здійснює автентифікацію користувача, звіряючи введене ним ім’я та пароль. Для цього процедура гешує

пароль після чого звіряє його з наявним у таблиці. У випадку, якщо ім’я та пароль відповідають тим, що знаходяться в таблиці користувач проходить автентифікацію.

Контроль доменної цілісності є ще однією важливою функцією мови SQL. Вона дозволяє встановлювати обмеження на значення, які можуть бути збережені в стовпцях таблиць. Це дозволяє забезпечити доменну цілісність даних і уникнути некоректних або неприпустимих значень у базі даних. В кожній таблиці розробленої бази даних встановлене обмеження NOT NULL для одного або декількох стовпців, що означає, що ці поля обов'язково повинні мати значення, і не дозволяються нульові значення у цих стовпцях. Також крім зовнішніх ключів, що вказують на інші таблиці, контроль доменної цілісності також реалізований за допомогою різних типів обмежень.

Наприклад, в таблиці image_data використовується обмеження PRIMARY KEY, яке гарантує унікальність значень у стовпці id. Також, в таблиці conditions використовується обмеження PRIMARY KEY, що забезпечує унікальність значень у стовпці condition_id. В кожній таблиці використовуються зовнішні ключі та обмеження, які забезпечують зв'язність та цілісність даних між відповідними таблицями. Загалом, ці обмеження та зовнішні ключі сприяють дотриманню правильних залежностей та цілісності даних в базі даних, гарантуючи, що лише коректні та валідні значення будуть збережені у таблицях.

Загалом, використання мови SQL для створення ролей, тригерів, процедур та контролю доменної цілісності є важливими аспектами безпеки та захисту баз даних. Вони забезпечують контроль доступу, автоматичну перевірку цілісності даних і спрощують управління базою даних. Відповідне використання цих інструментів сприяє зменшенню ризику несанкціонованого доступу до даних та забезпечує цілісність та конфіденційність інформації.

4.3 Блокове тестування захищеної бази даних

Під час блокового тестування перевіряється правильність виконання окремих функцій та операцій бази даних. Це включає перевірку правильності роботи тригерів та їх реакцію на певні події, перевірку коректності виконання процедур та

функцій, а також перевірку належного функціонування рольового розмежування для забезпечення відповідного контролю доступу до даних.

Першим доцільно провести тестування реєстрації користувача, та гешування паролю, для цього необхідно викликати процедуру RegisterUser та заповнити необхідні поля для введення, зокрема: нікнейм, пошту, пароль ідентифікатор аватара, ідентифікатор ролі та рівня підписки. Код даної процедури наведено в додатку В. Введення даних представлено на рисунку 4.1.

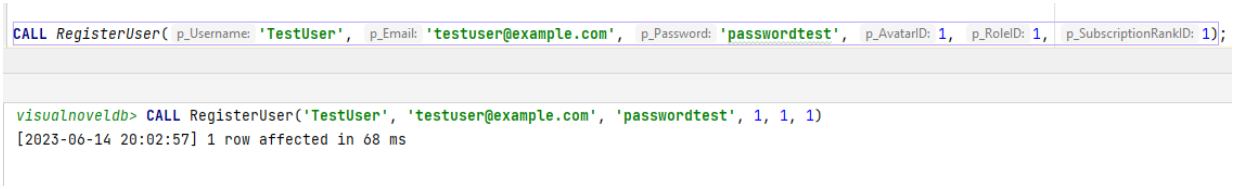


Рисунок 4.1 – Тестування процедури RegisterUser

Наступним кроком необхідно перевірити наявність відповідного запису в базі даних у таблиці users. Результати перевірки наведено на рисунку. 4.2.

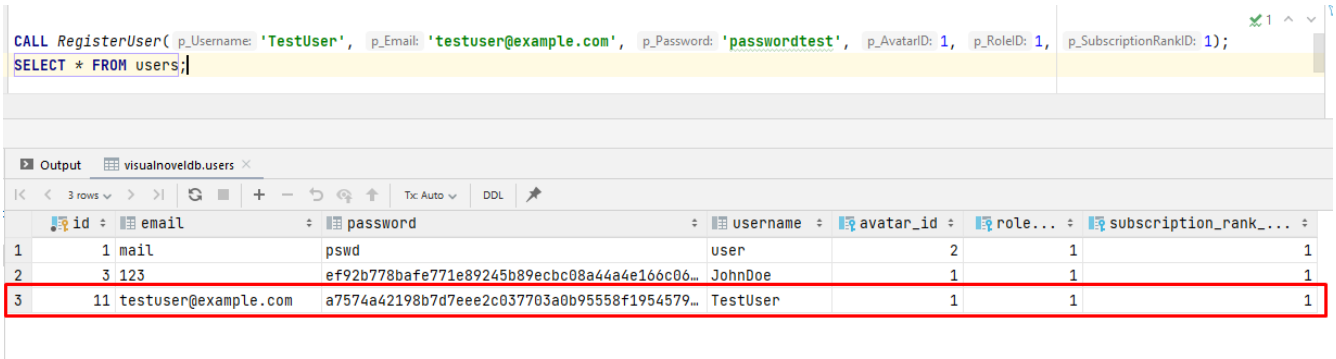


Рисунок 4.2 – Результат тестування процедури RegisterUser

Дані у таблицю були внесені коректно, необхідний сегмент, який підлягає захисту гешується та в наступних перевірках використовується для автентифікації користувача. У цьому сегменті тестування доцільно перевірити роботу тригера PreventDuplicateUsers який не дозволить додати два однакових користувача. У випадку спроби введення двох користувачів, тригер повертає повідомлення про помилку забороняючи введення дублюючих користувачів. Код тригера наведено в додатку В. Робота тригера наведена на рисунку 4.3.

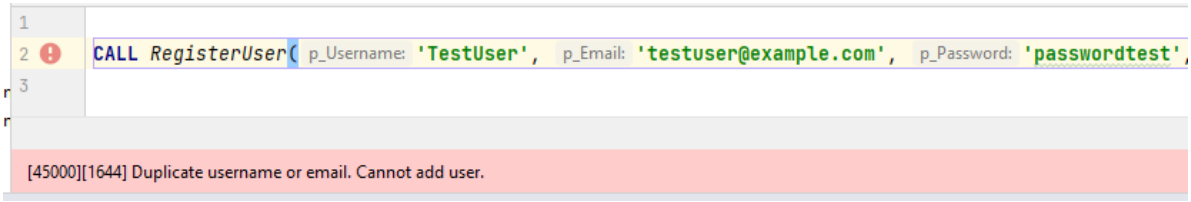


Рисунок 4.3 – Результат тестування тригера PreventDuplicateUsers

Відповідно до створеної процедури протестуємо автентифікацію користувача для цього використаємо процедуру AuthenticateUser яка порівняє введені користувачем дані з наявними в таблиці. Для того, щоб побачити результат перевірки, буде виводитись таблиця з відповідним значенням «1», якщо користувача автентифіковано та «0» якщо спроба невдала Код даної процедури наведено в додатку Б. Результат тестування процедури AuthenticateUser наведено на рисунку 4.4.

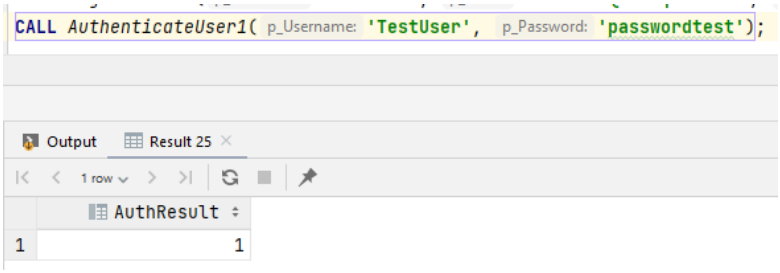


Рисунок 4.4 – Позитивний результат тестування процедури AuthenticateUser

Невдала автентифікація відбудеться у випадку, якщо користувач ввів некоректний пароль або ім’я, для перевірки проведемо аналогічні дії вказавши некоректний пароль passwordtest1 у такому випадку повернене значення має дорівнювати нулю. Результат тестування наведено на рисунку 4.5.

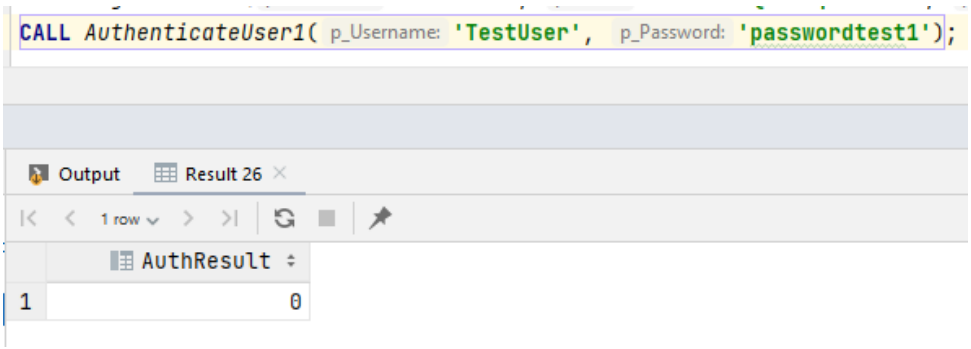


Рисунок 4.5 – Негативний результат тестування процедури AuthenticateUser

Наступним кроком перевірятиметься рольове розмежування користувачів для цього було створено ролі відповідно до визначених у попередніх розділах. Створені ролі наведено на рисунку 4.6.

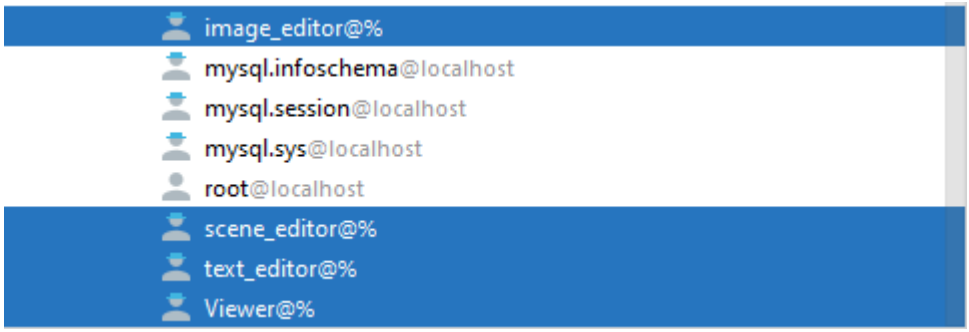


Рисунок 4.6 – Перелік створених ролей

Необхідно створити користувачів та призначити їм відповідні ролі. Таким чином створимо користувача TestSceneEditor який матиме роль scene_editor дана роль має можливість редагувати як графічний так і текстовий вміст, TestImageEditor з ролью image_editor яка дозволить редагувати таблиці з графічним вмустом, TestTextEditor – text_editor для редагування таблиць пов’язаних з репліками та користувача TestVeiwer який виконуватиме роль тестера з дозволом лише на перегляд таблиць. Створення користувачів та надання їм відповідних ролей наведено на рисунку 4.7)

```
-- Створення користувачів
CREATE USER 'TestSceneEditor'@'localhost' IDENTIFIED BY 'passwordtest1';
CREATE USER 'TestImageEditor'@'localhost' IDENTIFIED BY 'passwordtest1';
CREATE USER 'TestTextEditor'@'localhost' IDENTIFIED BY 'passwordtest1';
CREATE USER 'TestViewer'@'localhost' IDENTIFIED BY 'passwordtest1';

-- Надання ролей користувачам
GRANT scene_editor TO 'TestSceneEditor'@'localhost';
GRANT image_editor TO 'TestImageEditor'@'localhost';
GRANT text_editor TO 'TestTextEditor'@'localhost';
GRANT Viewer TO 'TestViewer'@'localhost';
```

```
visualnoveldb> CREATE USER 'TestSceneEditor'@'localhost' IDENTIFIED BY 'passwordtest1'
[2023-06-15 00:10:30] completed in 34 ms
visualnoveldb> CREATE USER 'TestImageEditor'@'localhost' IDENTIFIED BY 'passwordtest1'
[2023-06-15 00:10:30] completed in 35 ms
visualnoveldb> CREATE USER 'TestTextEditor'@'localhost' IDENTIFIED BY 'passwordtest1'
[2023-06-15 00:10:30] completed in 24 ms
visualnoveldb> CREATE USER 'TestViewer'@'localhost' IDENTIFIED BY 'passwordtest1'
[2023-06-15 00:10:30] completed in 38 ms
visualnoveldb> GRANT scene_editor TO 'TestSceneEditor'@'localhost'
[2023-06-15 00:10:30] completed in 19 ms
visualnoveldb> GRANT image_editor TO 'TestImageEditor'@'localhost'
[2023-06-15 00:10:30] completed in 29 ms
visualnoveldb> GRANT text_editor TO 'TestTextEditor'@'localhost'
[2023-06-15 00:10:30] completed in 22 ms
visualnoveldb> GRANT Viewer TO 'TestViewer'@'localhost'
[2023-06-15 00:10:30] completed in 27 ms
```

Рисунок 4.7 – Перелік створених користувачів

Перевірка коректності наданого доступу буде здійснюватись шляхом авторизації під відповідними користувачами. Так, користувачу TestImageEditor можуть бути доступні лише таблиця з графічним вмістом. Всі мультимедійні дані зберігаються в таблиці image_data відповідно, користувач TestImageEditor може мати доступ лише до неї. Перевірка доступу користувача до відповідної таблиці наведена на рисунку. 4.8.

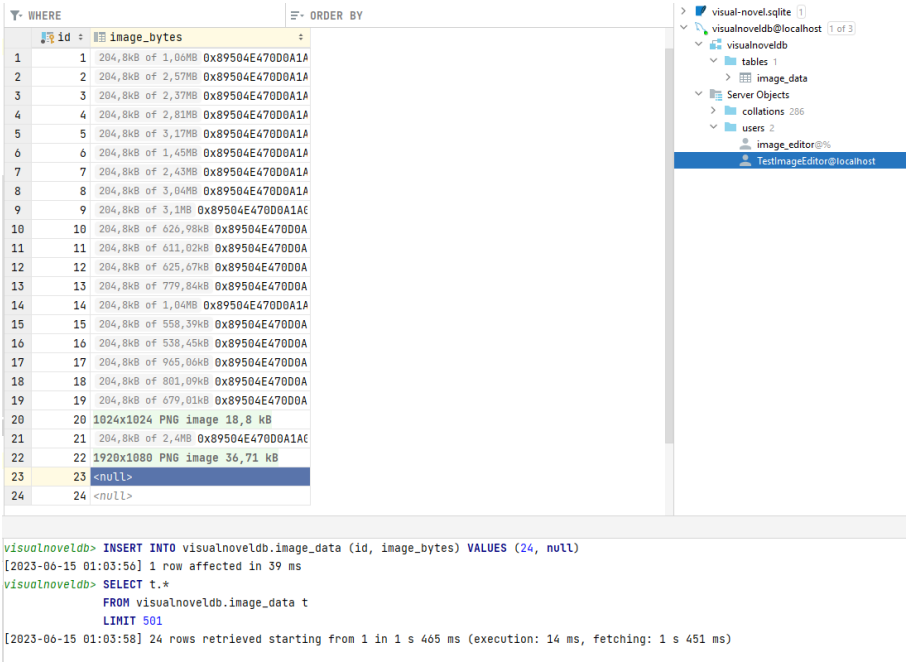


Рисунок 4.8 – Результат перевірки доступу користувача до таблиці image_data

Як можна побачити користувачу TestImageEditor доступна лише одна таблиця яка містить мультимедійні дані. Здійснено перевірку на внесення змін в таблицю, як можна побачити зміни були внесені. Аналогічно до цього має доступи користувач TestTextEditor. Далі доцільно розглянути користувача TestSceneEditor який має ролі image_editor та text_editor а також доступи до таблиць edges та options. Перевірка прав доступу користувача TestSceneEditor наведена на рисунку 4.9.

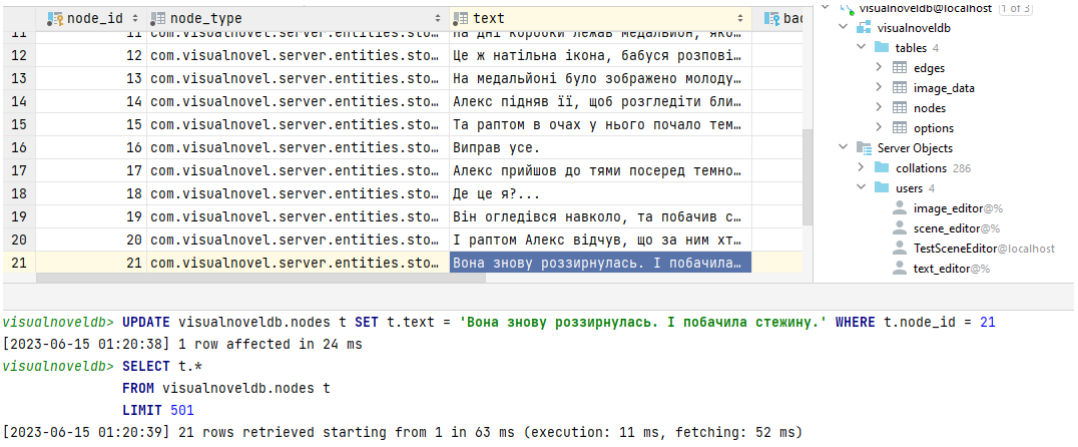
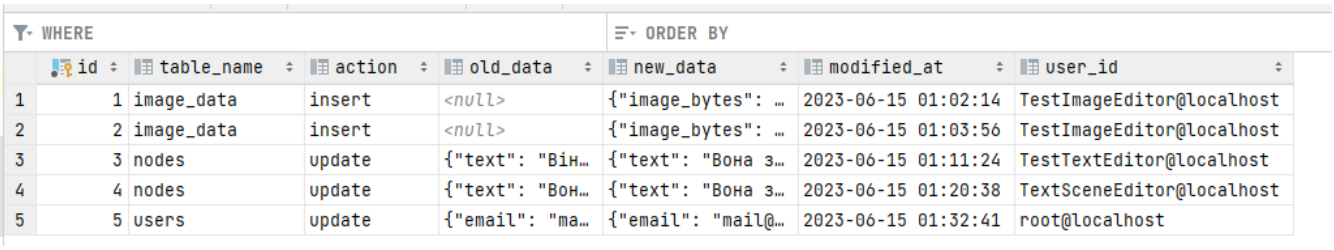


Рисунок 4.9 – Результат перевірки доступу користувача TestSceneEditor до таблиць

Після внесення змін в базу даних доцільно протестувати роботу тригера який відповідає за логування дій в базі даних. Для цього достатньо перевірити таблицю audit_log так, як в попередніх тестуваннях вносились зміни повязанні з введенням та оновленням даних таблиць. Код ьтригерів, які відповідають за аудит подій наведено в додатку В. Результат перевірки таблиці audit_log наведено на рисунку 4.10.



WHERE ORDER BY

	id	table_name	action	old_data	new_data	modified_at	user_id
1	1	image_data	insert	<null>	{"image_bytes": …	2023-06-15 01:02:14	TestImageEditor@localhost
2	2	image_data	insert	<null>	{"image_bytes": …	2023-06-15 01:03:56	TestImageEditor@localhost
3	3	nodes	update	{"text": "Він…	{"text": "Вона з…	2023-06-15 01:11:24	TestTextEditor@localhost
4	4	nodes	update	{"text": "Вон…	{"text": "Вона з…	2023-06-15 01:20:38	TextSceneEditor@localhost
5	5	users	update	{"email": "ma…	{"email": "mail@…	2023-06-15 01:32:41	root@localhost

Рисунок 4.10 – Результат перевірки таблиці audit_log

Як можна побачити тригер коректно виконує задану функцію, реєструючи ім’я таблиці, вид дії, стару версію даних, що дозволяє у разі необхідності замінити дані, як наслідок виправити можливу помилку яку допустив користувач, нові дані, час їх зміни та користувача який безпосередньо виконував дії пов’язані з базою даних.

4.4 Інтеграційне тестування захищеної бази даних та застосунку

Інтеграційне тестування визначається як процес перевірки взаємодії різних модулів у складі програмного застосунку. У розробленому засобі, особлива увага і обов'язкове тестування приділяються встановленню та перевірці зв'язку між застосунком і базою даних. Це означає, що необхідно провести тестування роботи репозиторіїв та сервісів. Для досягнення надійності та якості системи, рекомендується проводити окреме тестування клієнтської та серверної частин застосунку. Це дозволить виявити та виправити можливі проблеми, що виникають під час взаємодії між компонентами програми, забезпечуючи стабільну та безперебійну роботу програмного застосунку. Для тестування перевіримо зв’язок засобу з базою даних зокрема таблиці nodes шляхом введення в базу даних

інформації різного типу. Результати виконання тесту наведено на рисунку. 4.11. Дані матимуть такий вигляд:

– 100L, "100TextNode", personageService.getById(10L), imageDataService.getImage(21L), imageDataService.getImage(21L)

– "TextNode without ID", "100TextNode", personageService.getById(10L), imageDataService.getImage(21L), imageDataService.getImage(21L)

– 101L, "", personageService.getById(10L), imageDataService.getImage(21L), imageDataService.getImage(21L);

– 102L, "102 TextNode", null, imageDataService.getImage(21L), imageDataService.getImage(21L);

– 103L, "103 TextNode", personageService.getById(10L), null, imageDataService.getImage(21L);

– 104L, "104 TextNode", personageService.getById(10L), imageDataService.getImage(21L), null.

На рисунку 4.11 наведено результати автоматичного перебігу інтеграційних тестів.

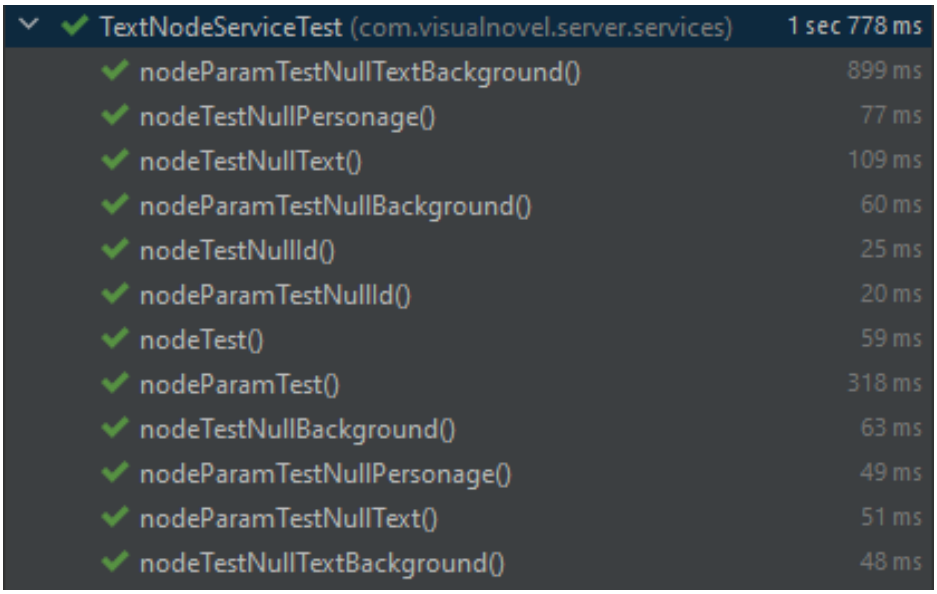


Рисунок 4.11 – Результати виконання тесту nodes

Наступним кроком переглянемо результати введення в таблицю. У випадку якщо умова тесту суперечить можливості введення даних мають підставлятись

стандартні значення так для наведених колонок таблиці nodes стандартні значення такі:

- node_id – auto;
- node_type – com.visualnovel.server.entities.story.node.TextNode;
- text – 100 TexNodel;
- background_id – 22;
- personage_id – 12;
- text_background_id – 2.

Результати тестування наведено на рисунку 4.12.

WHERE ORDER BY

	node_id	node_type	text	b...	pers...	text_background_id
22	100	com.visualnovel.…	100TextNode	21	10	21
23	101	com.visualnovel.…	100TextNode	21	10	21
24	102	com.visualnovel.…	100TextNode	21	10	21
25	103	com.visualnovel.…	102 TextNode	21	12	21
26	104	com.visualnovel.…	103 TextNode	22	10	21
27	105	com.visualnovel.…	104 TextNode	21	10	2

Рисунок 4.12 – Результати заповнення таблиці nodes

Як можна побачити, при спробі введення даних, які не задовольняють умови підставляються визначенні стандартні дані. Аналогічно до тестування nodes виконано тестування для таблиць пов’язаних з Users. Результат тестування наведено на рисунку 4.13.

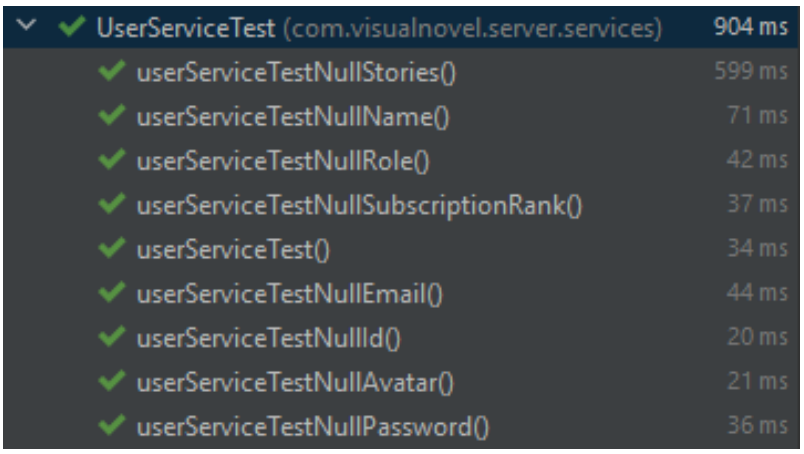


Рисунок 4.13 – Результати виконання тесту Users

Також варто провести тестування для бази даних клієнта, яка зберігає особисті налаштування користувача. Таким чином тест кейси виглядають таким чином:

– new Settings(0.3);
– new Settings(0));
– new Settings(1);
– new Settings(-1).

Результат тестування представлено на рисунку. 4.14.

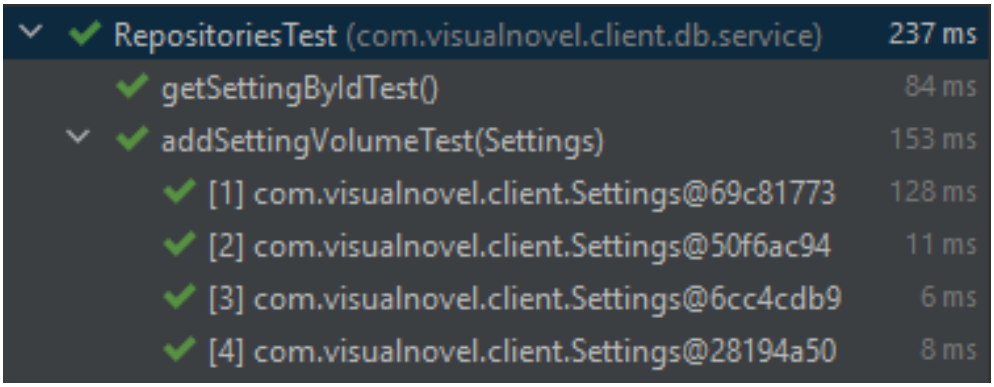


Рисунок 4.14 – Результати виконання тесту Settings

Проводячи тестування можна перевірити реакцію таблиць на спроби введення даних у різній послідовності, різного типу. Працююча інтегрована база даних у випадку спроби введення некоректних даних має підставляти завчасно визначені дані, як наслідок не порушуючи загальну роботу засобу для поширення візуальних новел.

4.5 Мануальне тестування бази даних

На даному етапі доцільно перевірити, як засіб для поширення візуальних новел звертається до таблиць, та як змінюються дані відповідно до перебігу сюжету. Так до прикладу при перебігу діалогових та загальних сцен змінюються дані які використовує засіб, коли звертається до таблиць, приклад початкових даних які використовує засіб наведено на рисунку. 4.15.

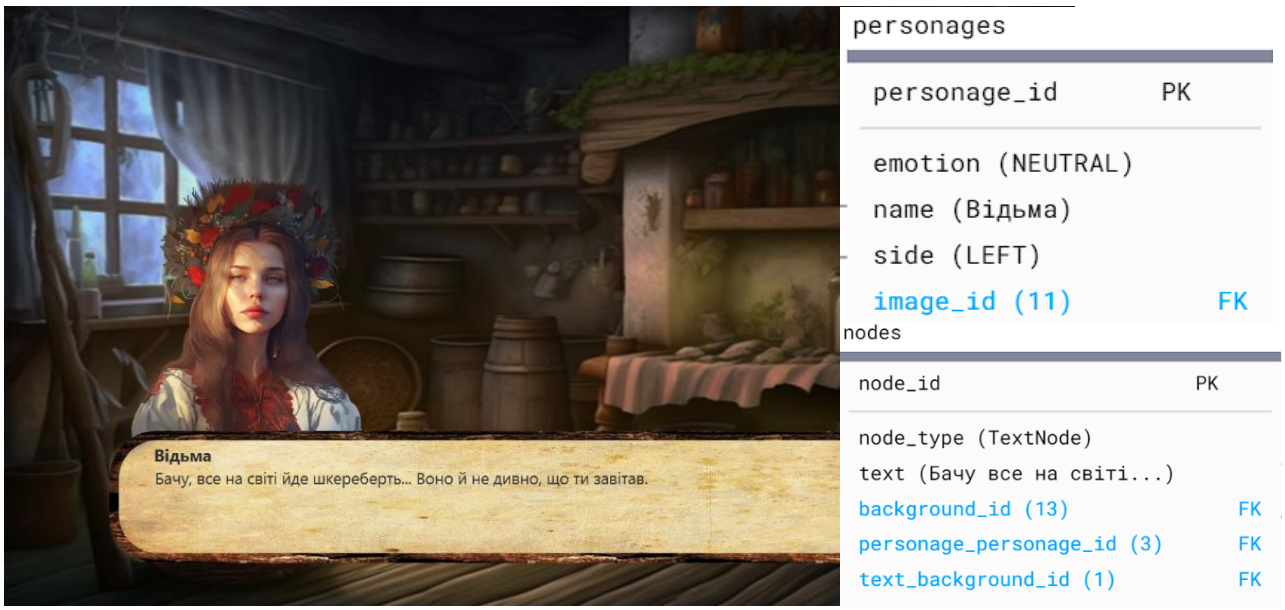


Рисунок 4.15 – Використовувані дані в першій діалоговій сцені

На початку сцени засіб використовує такі початкові дані з таблиці personages: нейтральну емоцію, ідентифікатор, ім’я, та ідентифікатор зображення персонажа. У таблиці nodes визначено, що тип тексту є розмовним, без варіантів відповіді, атрибут text зберігає фрази, які відображаються у текстовому полі, ідентифікатор фонового зображення визначає який задній фон буде використаний в конкретній сцені, ідентифікатор персонажа визначає який персонаж присутній у даний момент на сцені.

Після переходу до наступної діалогової сцени дані, які використовуватиме засіб зміняться, зокрема: тип тексту, емоції, зображення. Результат зміни використовуваних даних наведено на рисунку 4.16.

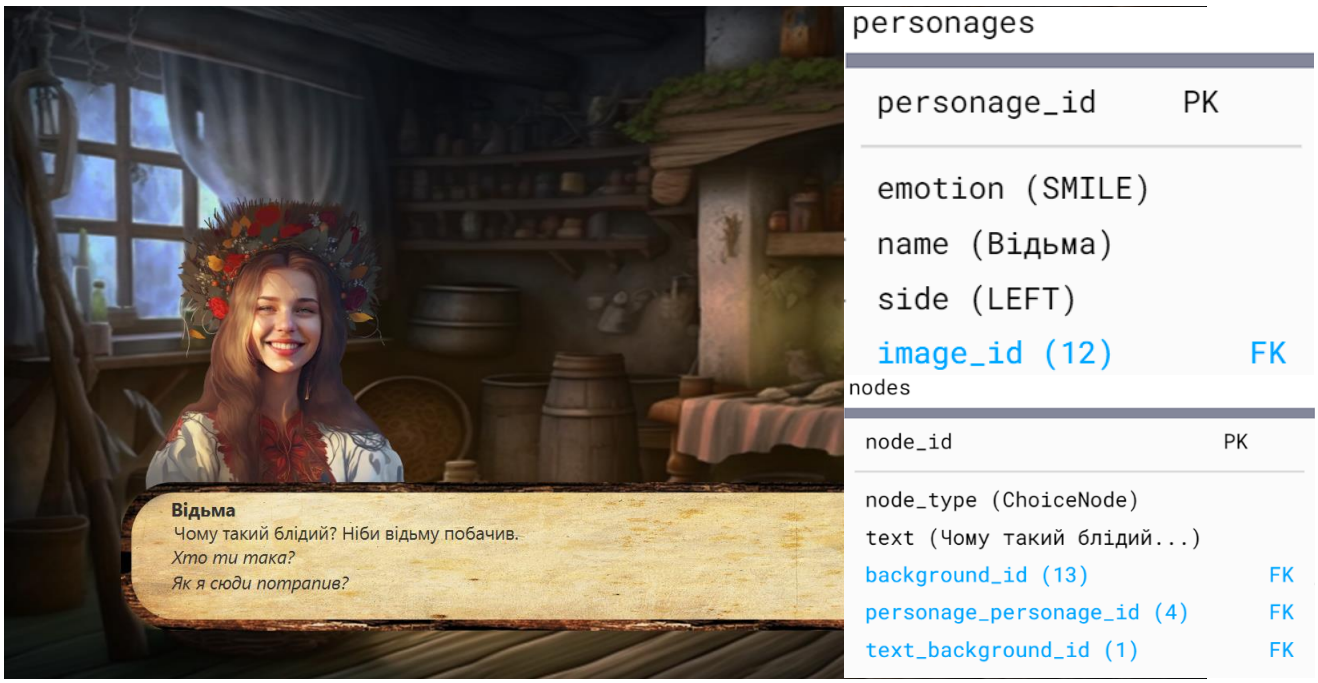


Рисунок 4.16 – Результат зміни даних в другій діалоговій сцені

При переході до наступної діалогової сцени відбулася зміна даних, які використовує засіб. Так змінився основний атрибут node_type з таблиці nodes на ChoiceNode який зазначає, що даний тип тексту передбачає відповідь та атрибут text у якому тепер використовується інша фраза відповідно до діалогу. Дані які використовує засіб зазнали змін. У таблиці personages змінився атрибут який відповідає за емоцію посмішки персонажа, ідентифікатор персонажа вказує на зміну картинки персонажа з нейтральною емоцією на персонажа з посмішкою.

Наступним кроком перевірено зміну загальної сцени, включаючи зміну всіх запрошуваних даних. Таким чином, була обрана остання сцена, після якої відбувається зміна загальної сцени. Дані які використовує засіб в початковій сцені наведено на рисунку 4.17.

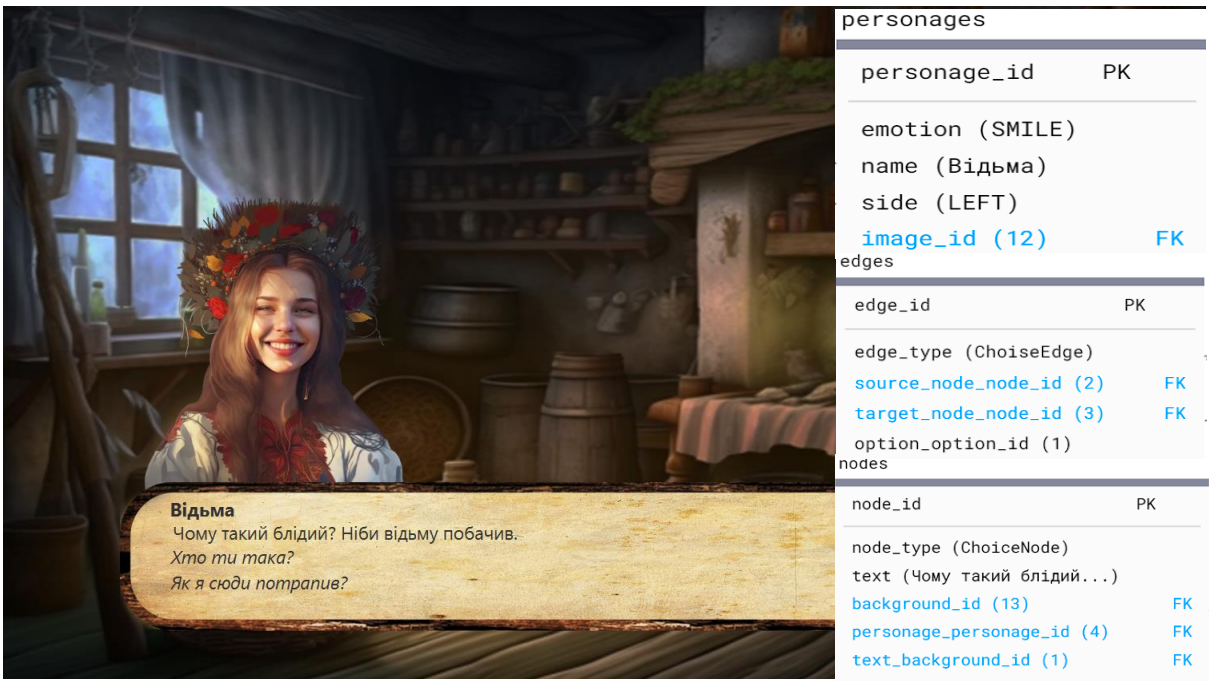


Рисунок 4.17 – Використовувані дані в першій загальній сцені

Після вибору відповіді, відбувається зміна загальної сцени включаючи дані з таблиці personages та nodes а саме: фон, персонажа, його емоцію, позицію відносно діалогового вікна тип тексту та текст. Таблиця edges відповідає за правильний порядок сцен відповідно до визначеного сюжету. Вона містить атрибути source_node_node_id який вказує на діючу сцену, яку бачить користувач у даний момент та target_node_node_id який вказує на ідентифікатор наступної сцени, яку побачить користувач.

При зміні сцени, засіб почне використовувати зовсім інші дані відповідно до кожної таблиці. Дані які використовує засіб в другій загальній сцені наведено на рисунку 4.18.

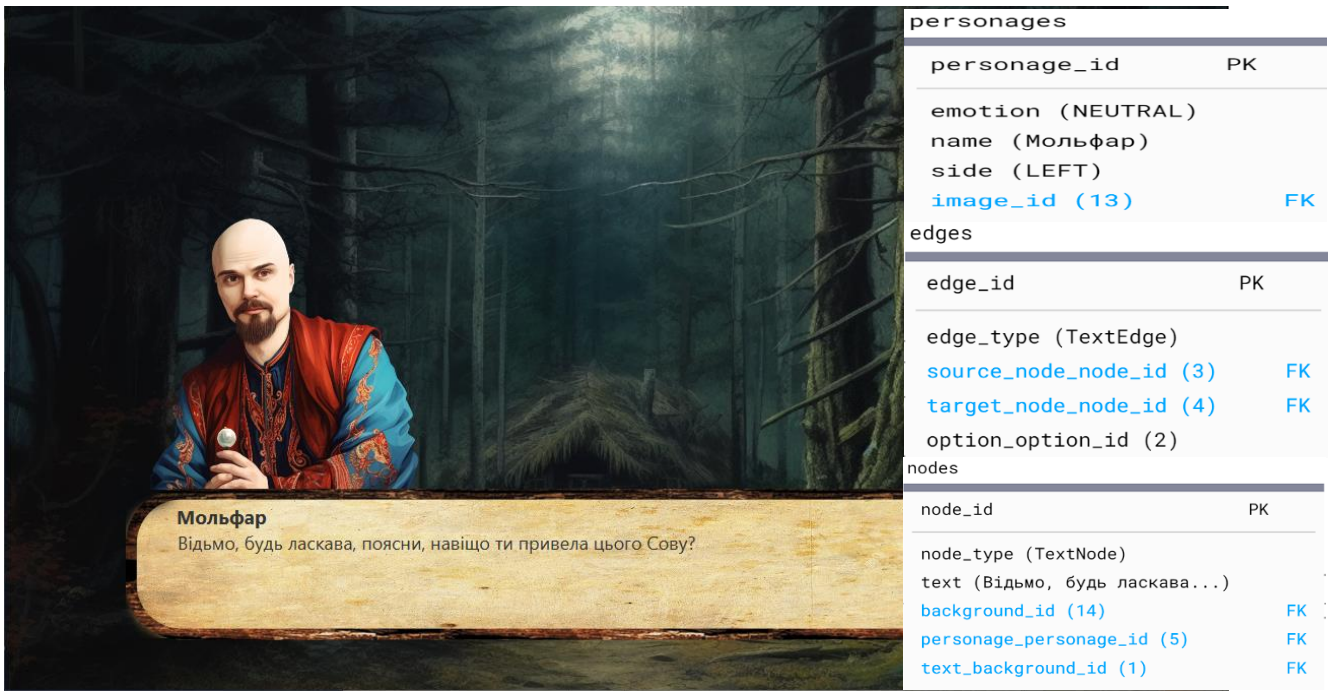


Рисунок 4.18 – Результат Зміни даних в другій загальній сцені

При зміні сцени, засіб змінив усі дані які використовував до цього у попередніх сценах з таблиць personages, nodes та edges.Таким чином відбулася зміна персонажа у таблиці personages, був встановлений новий порядок перебігу сцен у таблиці edges. типу та тексту діалогу у таблиці nodes.

У діалогових сценах, користувач може взаємодіяти з декількома персонажами одночасно. Це означає, що на сцені можуть бути присутні кілька персонажів, і користувач може спостерігати діалоги, що відбуваються між ними. Для здійснення такої функціональності, дані що використовує засіб для поширення візуальних новел з таблиць "personages" та "nodes" можуть змінюватись відповідно до реплік кожного з персонажів, які беруть участь у діалозі.

Це означає, що змінюються вміст і атрибути записів про персонажів у таблиці "personages", а також змінюються у таблиці "nodes", що відображають діалоги та взаємодію персонажів які використовує засіб. Цей підхід дозволяє створювати реалістичні діалогові сцени, в яких персонажі можуть взаємодіяти між собою, а користувач може спостерігати ці взаємодії та відчути більшу глибину реалізму в оповіді. Таким чином діалог між персонажами на одній діалоговій сцені та дані які використовує засіб представлені на рисунку 4.19.

Рисунок 4.19 – Початкові дані у діалоговій сцені

У прикладі розглянуто діалог між трьома персонажами, який відбувається на одній загальній сцені, після зміни діалогової сцени, відбувається зміна персонажа та наступна репліка яка наведена на рисунку 4.20

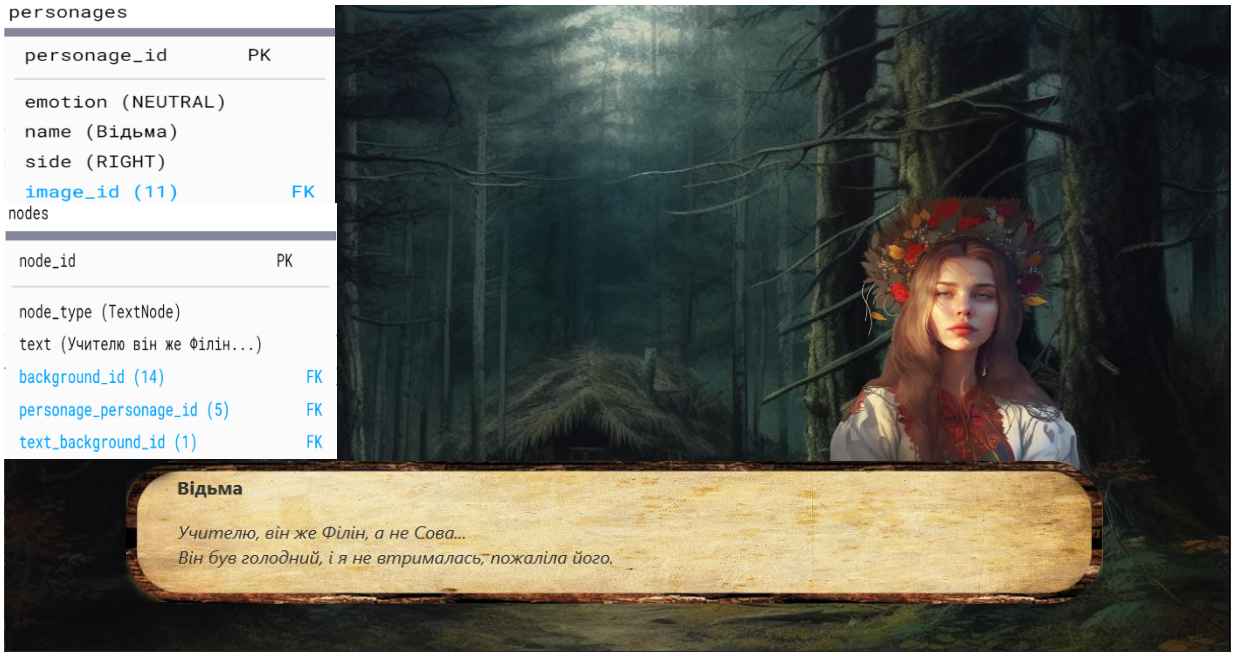


Рисунок 4.20 – Репліка другого персонажу та зміна даних

Так, як діалог відбувається між двома та більше персонажами, наступною діалоговою сценою буде репліка третього персонажа наведена на рисунку 4.21.

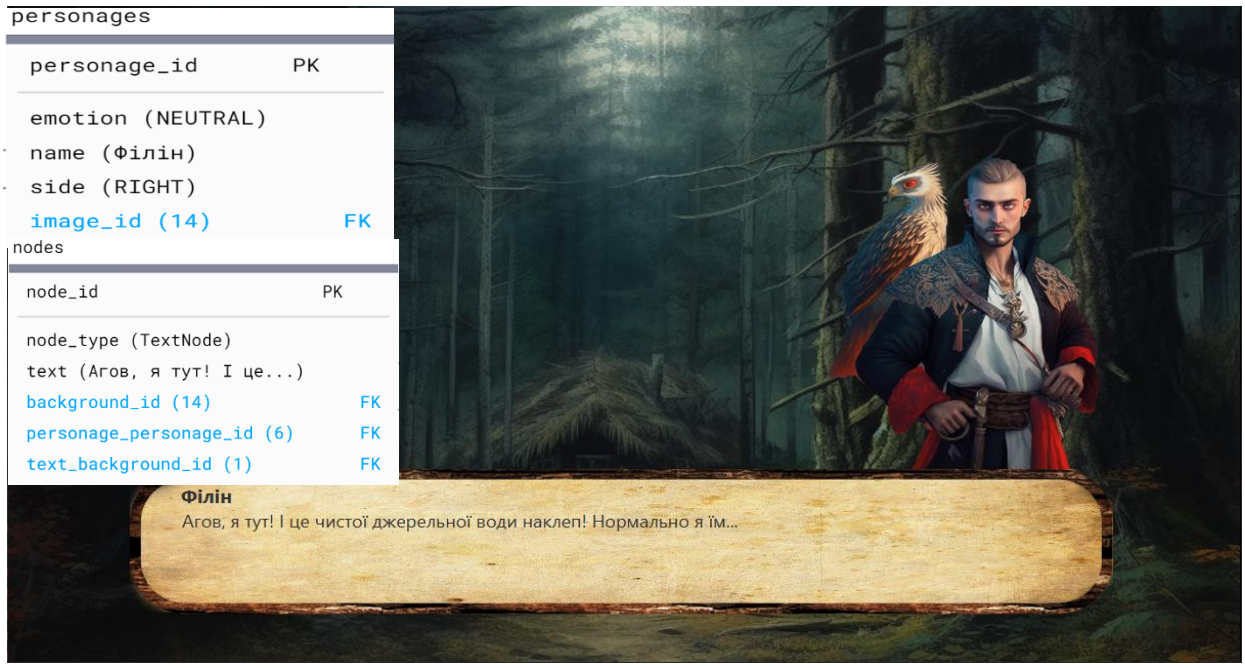


Рисунок 4.21 – Репліка третього персонажу та зміна даних

Відповідно до наведених результатів тестування можна зробити висновок, що програма ефективно використовує дані з бази даних і коректно змінює їх відповідно до вимог сценарію. У цьому контексті, коли змінюється сцена, програма успішно оновлює персонажа, його емоції та відповідний текст, що сприяє покращенню візуального відображення гри.

4.6 Висновки з розділу

На основі аналізу найпоширеніших засобів розробки, було враховано ключові вимоги до продуктивності, масштабованості, широкого спектру функцій та зручного інтерфейсу. В результаті порівняння різних інструментів, був обраний MySQL Workbench як потужний інструмент для розробки та управління базами даних. Його масштабованість, здатність працювати з великим обсягом даних, ефективна обробка складних запитів та підтримка реплікації та кластеризації роблять його ідеальним вибором для розробки захищеної бази даних для засобу поширення візуальних новел.

Розглянуто реалізацію процедур захисту мовою SQL зокрема можливості розмежування прав доступу, впровадження процедур та тригерів для реалізації

комплексного захисту. Під час блокового тестування перевірено правильність функціонування тригерів та процедур. Інтеграційне тестування захищеної бази даних дозволило здійснити перевірку взаємодії різних модулів та компонентів застосунку. Таким чином були перевірені основні модулі, пов'язані зі зламом цілісності даних та некоректною передачею інформації між різними частинами системи. Проведення окремого тестування на клієнтській та серверній сторонах допомогло забезпечити стабільну та безперебійну роботу захищеної бази даних. Застосування автоматизованих тестових сценаріїв забезпечило повноту тестування та ефективну перевірку функціональності. Використання відповідних засобів розробки, реалізація процедур захисту, проведення блокового інтеграційного тестування та мануального сприяють забезпеченню безпеки, надійності та оптимальної роботи захищеної бази даних.

ВИСНОВКИ

У комплексній бакалаврській дипломній роботі з розробки захищеної бази даних для засобу поширення візуальних новел було проведено аналіз основних методів захисту баз даних створених для комп'ютерних ігор. Проаналізовано основні загрози, які постають перед розробниками комп'ютерних ігор. Як наслідок з виконаного аналізу виявлено, що основна частка загроз припадають на дані, які знаходяться в базах даних. Тому в подальшому було розглянуто методи захисту баз даних в сучасних комп’ютерних іграх та визначено, що використання захисту, який включає не тільки інтегрований захист, але й превентивний такий, як моніторинг дозволяє побудувати комплексний захист, який значно зменшує ризики втручання у базу даних та несанкціонованого маніпулювання даними у ній. Таким чином було виявлено, що забезпечення безпеки даних є однією з ключових задач у сфері розробки ігор, а особливо ігор жанру візуальних новел, які в більшій мірі будують комерційний успіх на основі контенту, тобто даних, які зберігаються в базі даних, і в меншій – на “механіці” або правилах гри.

Реалізовано побудову узагальненої архітектури засобу, з якої визначено, що використання клієнт-серверної моделі забезпечить можливість кешування даних на стороні клієнта, що сприяє масштабованості застосунку. Крім того, наявність бази даних на стороні клієнта дозволяє зменшити навантаження на сервер. Шляхом ідентифікації основних сутностей визначено, основні складові бази даних, та її атрибути в наслідок чого розроблено початкову базу даних для сервера та клієнта. Відповідно до вимог базу даних було нормалізовано до третьої нормальної форми. Виконання цих дій дозволило розробити стійку основу для побудови комплексної захищеної бази даних та зменшити ймовірність виникнення аномалій в базі даних, таким чином зменшити ризики для цілісності даних, які зберігаються в ній.

На основі спроєктованої бази даних було запропоновано загальний алгоритм роботи захищеної бази даних, а також метод розмежування прав доступу до бази даних та алгоритм захисту даних на стороні сервера. Було детально вивчено роботу тригерів та процедур у контексті захисту даних та запропоновано їх реалізацію для

захисту бази даних засобу поширення візуальних новел. В результаті проведеного аналізу були розроблені ефективні методи та алгоритми захисту бази даних комп’ютерних ігор зокрема процедури для реєстрації користувачів з інтегрованих гешуванням, перевірки автентифікації та тригери логування подій.

Варіантний аналіз засобів розробки дозволив визначити найкращий інструмент, який відповідає висунутим вимогам, які було отримано внаслідок аналізу ландшафту загроз та відомих засобів протидії ним, а також бізнес-вимог до конкретного засобу поширення контенту візуальних новел, який розробляється в межах цієї комплексної бакалаврської дипломної роботи. Проведення блокового та інтеграційного тестування захищеної бази даних та застосунку, дозволило довести правильність функціонування системи та взаємодію з іншими компонентами засобу для поширення візуальних новел, а відтак довести і коректність технічних рішень, прийнятих під час процесу дипломного проєктування.

Результати роботи можуть бути використані при проектуванні та розробці ігрових систем з метою забезпечення безпеки та захисту конфіденційних даних користувачів. А при внесенні певних змін до структури захищеної бази даних її можна масштабувати для поширення контенту комп'ютерних ігор інших жанрів, близьких до жанру візуальних новел, зокрема, текстових RPG та adventure.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Савчук І. Б. Модель рольового розмежування прав доступу для бази даних графічної новели. м. Вінниця / Наук. керівник Ю. В. Баришев. Вінниця, ВНТУ 2023.
2. How Much Is The Game Industry Worth?. bankmycell. URL: https://www.bankmycell.com/blog/video-game-industry-revenue (accessed: 22.05.2023).
3. XTI S. Increasing Cyberattacks Targeting the Gaming Industry in 2022. SOCRadar® Cyber Intelligence Inc. URL: https://socradar.io/increasing-cyberattacks-targeting-the-gaming-industry-in-2022/ (accessed: 23.05.2023).
4. Cyberpunk 2077 on Steam. Steam. URL: https://store.steampowered.com/app/1091500/Cyberpunk_2077/ (accessed: 27.05.2023).
5. Criddle B. C. Cyberpunk 2077 makers CD Projekt hit by ransomware hack. BBC News. URL: https://www.bbc.com/news/technology-55994787 (accessed: 27.05.2023).
6. What is Cheating?. Computer Hope's Free Computer Help. URL: https://www.computerhope.com/jargon/c/cheat.htm (accessed: 24.05.2023).
7. Counter-Strike: Global Offensive. steam. URL: https://store.steampowered.com/app/730/CounterStrike_Global_Offensive/ (accessed: 27.05.2023).
8. Gaming Respawned. akamai. URL: https://www.akamai.com/resources/state-of-the-internet/soti-security-gaming-respawned (accessed: 24.05.2023).
9. Rainbow Six Siege DDOS Attackers Hit With Ubisoft Lawsuit. gamerant. URL: https://gamerant.com/rainbow-six-siege-ddos-attack-lawsuit/ (accessed: 27.05.2023).
10. Stuart K. Ubisoft hack: users warned to change passwords. the Guardian. URL: https://www.theguardian.com/technology/2013/jul/03/ubisoft-hack-users-warned (accessed: 27.05.2023).

11. Cansever D. Security Games with Insider Threats. Lecture Notes in Computer Science. Cham, 2020. P. 502–505.
12. Smith J. Database Encryption Techniques for Data Protection. International Journal of Information Security. №3, Vol. 12, 2021. Pp. 189-204.
13. Thompson D. Backup and Disaster Recovery Best Practices for Databases. Journal of Database Management. №4, Vol. 29, 2021. Pp. 58-73.
14. Johnson M. Access Control Strategies for Database Security. Journal of Cybersecurity, №2, Vol. 5, 2022. Pp. 120-135.
15. White E. Database Auditing and Logging for Enhanced Security. Journal of Information Systems Security. №1, Vol. 9, 2021. Pp. 45-61.
16. Anderson R. Intrusion Detection and Prevention Systems for Database Security. International Journal of Network Security, №6, Vol. 18, 2022. Pp. 754-769.
17. Roach J. What is DRM is Video Games and How Does it Work? | Digital Trends. Digital Trends. URL: https://www.digitaltrends.com/gaming/what-is-drm-in-video-games/ (accessed: 27.05.2023).
18. Game Obfuscation and Security: Why You Should Definitely Obfuscate Your Game. Pim de Witte. URL: https://pimdewitte.me/2016/11/06/game-obfuscation-and-security-why-you-should-definitely-obfuscate-your-game/ (accessed: 27.05.2023).
19. Part VIIa: Security (TLS/SSL) of 64 Network DO’s and DON’Ts for Multi-Player Game Developers - IT Hare on Soft.ware. IT Hare on Soft.ware. URL: http://ithare.com/64-network-dos-and-donts-for-multi-player-game-developers-part-viia-security-tls-ssl/ (accessed: 27.05.2023).
20. David. What is 'Easy Anti Cheat' that appears in some games, what does it do? Here is the solution to the 'Easy Anti Cheat Not Installed' problem - Esportschimp. Esportschimp. URL: https://esportschimp.com/entertainment/16711.html (accessed: 27.05.2023).
21. BattlEye – The Anti-Cheat Gold Standard » About. BattlEye – The Anti-Cheat Gold Standard. URL: https://www.battleye.com/about/ (accessed: 27.05.2023).

22. VAC Integration (Steamworks Documentation). Steamworks. URL: https://partner.steamgames.com/doc/features/anticheat/vac_integration (accessed: 27.05.2023).
23. Call of Duty®: Warzone™ 2.0. steam. URL: https://store.steampowered.com/app/1962663/Call_of_Duty_Warzone_20/ (accessed: 27.05.2023).
24. Everything you need to know about Valorant. mint. URL: https://www.livemint.com/technology/tech-news/everything-you-need-to-know-about-valorant-11591217861198.html (accessed: 27.05.2023).
25. Tom Clancy's Rainbow Six® Siege. steam. URL: https://store.steampowered.com/app/359550/Tom_Clancys_Rainbow_Six_Siege/?l=english (accessed: 27.05.2023).
26. Apex Legends™. steam. URL: https://store.steampowered.com/app/1172470/Apex_Legends/ (accessed: 27.05.2023).
27. The Witcher 3: Wild Hunt. Official Website. URL: https://www.thewitcher.com/us/en/witcher3#home (accessed: 27.05.2023).
28. The Elder Scrolls | Skyrim. Elder Scrolls. URL: https://elderscrolls.bethesda.net/en/skyrim (accessed: 27.05.2023).
29. Far Cry (series). Far Cry Wiki. URL: https://farcry.fandom.com/wiki/Far_Cry_(series) (accessed: 27.05.2023).
30. MINECRAFT. minecraft. URL: https://www.minecraft.net/en-us/about-minecraft (accessed: 27.05.2023).
31. Red Dead Redemption 2. rockstargames. URL: https://www.rockstargames.com/reddeadredemption2/ (accessed: 27.05.2023).
32. DARK SOULS™ III. steam. URL: https://store.steampowered.com/app/374320/DARK_SOULS_III/ (accessed: 27.05.2023).
33. A01 Broken Access Control - OWASP Top 10:2021. OWASP Foundation, the Open Source Foundation for Application Security | OWASP Foundation. URL:

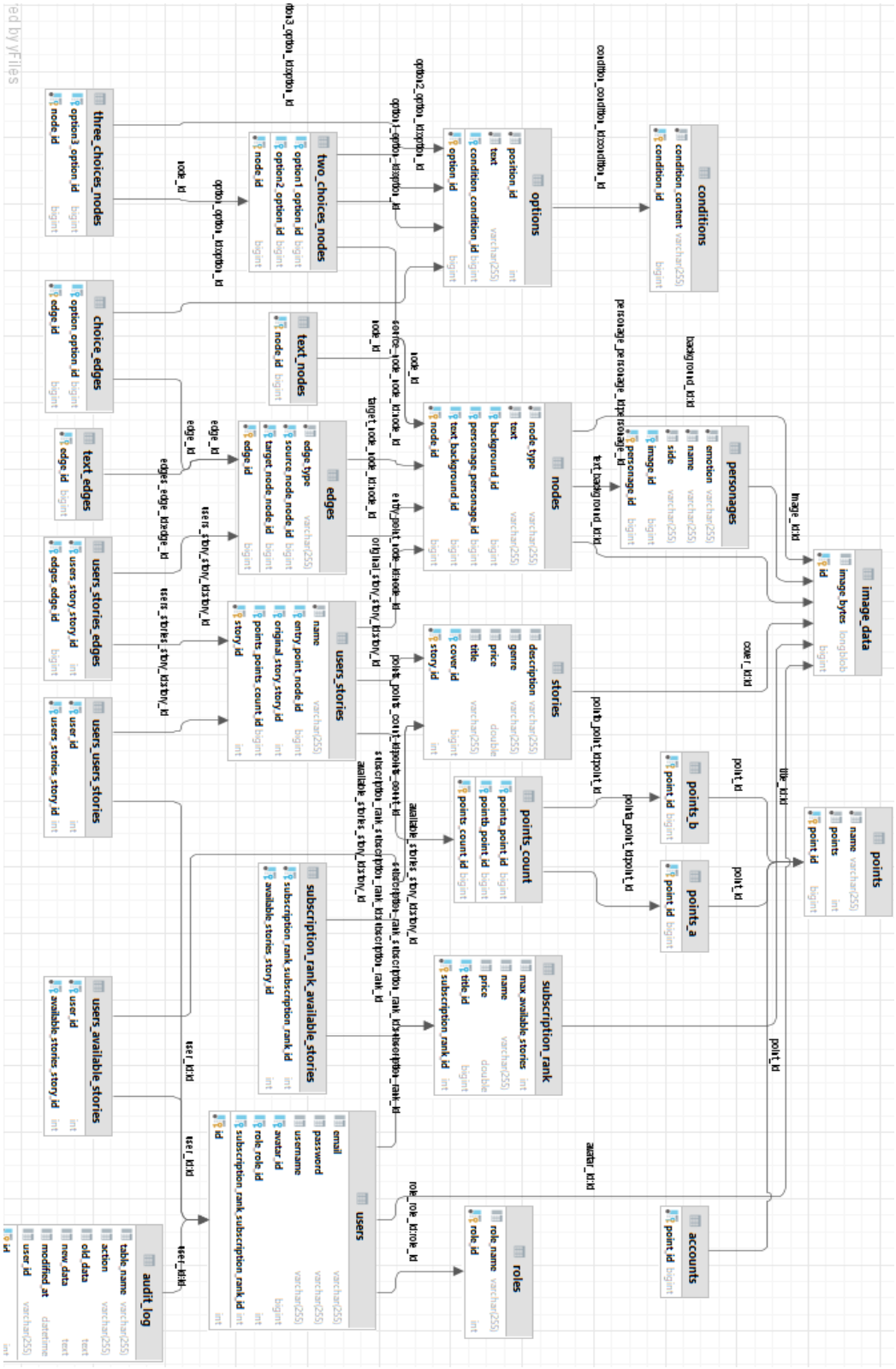
https://owasp.org/Top10/A01_2021-Broken_Access_Control/ (date of access: 27.05.2023).

34. MySQL workbench is. URL: https://www.mysql.com/products/workbench/ (accessed: 01.06.2023).
35. SQLite Home Page. SQLite Home Page. URL: https://www.sqlite.org/index.html (accessed: 01.06.2023).
36. SQL Developer. Oracle | Cloud Applications and Cloud Platform. URL: https://www.oracle.com/cis/database/sqldeveloper/ (accessed: 01.06.2023).
37. Sirkin J., Hughes A. What is Microsoft SQL Server Management Studio (SSMS)? | Definition from TechTarget. Data Management. URL: https://www.techtarget.com/searchdatamanagement/definition/Microsoft-SQL-Server-Management-Studio-SSMS (accessed: 01.06.2023).
38. Brown Michael. Threats and Vulnerabilities in Computer Games. Proceedings of the International Conference on Cybersecurity, 2019, pp. 215-230.
39. Каплун В. А., Дмитришин О. В., Баришев Ю. В. Захист програмного забезпечення. Частина 2 : навчальний посібник. Вінниця : ВНТУ, 2014. 105 с.

ДОДАТКИ

ДОДАТОК А

ER-ДІАГРАМА НОРМАЛІЗОВАНОЇ БАЗИ ДАНИХ



ДОДАТОК Б
ТЕКСТ РОЗРОБЛЕНИХ ЗАСОБІВ АВТОМАТИЗАЦІЇ ЗАХИСТУ БАЗИ ДАНИХ

Код процедури RegisterUser

```
CREATE PROCEDURE RegisterUser
( IN p_Username VARCHAR(255),
 IN p_Email VARCHAR(255),
 IN p_Password VARCHAR(255),
 IN p_AvatarID BIGINT,
 IN p_RoleID INT,
 IN p_SubscriptionRankID INT)
 BEGIN

 DECLARE HashedPassword VARCHAR(255);
 SET HashedPassword = SHA3(p_Password, 256);
 INSERT INTO users (username, email, password, avatar_id, role_role_id, subscription_rank_subscription_rank_id)
 VALUES (p_Username, p_Email, HashedPassword, p_AvatarID, p_RoleID, p_SubscriptionRankID);
END //
```

Код процедури AuthenticateUser

```
DELIMITER //

CREATE PROCEDURE AuthenticateUser(
    IN p_Username VARCHAR(255),
    IN p_Password VARCHAR(255)
)
BEGIN
    -- Encrypt the entered password
    DECLARE HashedPassword VARCHAR(255);
    SET HashedPassword = SHA3(p_Password, 256);

    -- Check if the username and hashed password match
    IF EXISTS (
        SELECT 1
        FROM users


WHERE username = p_Username AND password = HashedPassword
    )
    THEN
        -- Authentication successful
        SELECT 1 AS AuthResult;
    ELSE


-- Authentication failed
        SELECT 0 AS AuthResult;
    END IF;
END //

DELIMITER
```

Код процедури PreventDuplicateUsers

```
CREATE TRIGGER PreventDuplicateUsers
BEFORE INSERT ON users
FOR EACH ROW
BEGIN
   DECLARE userCount INT;

   SELECT COUNT(*) INTO userCount
   FROM users
   WHERE username = NEW.username;

   IF userCount > 0 THEN
      SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Duplicate username. Cannot add user.';
   END IF;
END //
```

Код тригера users_audit_trigger

```
CREATE TRIGGER users_audit_trigger
AFTER UPDATE ON users
FOR EACH ROW
BEGIN
  IF OLD.email != NEW.email OR OLD.username != NEW.username OR OLD.avatar_id != NEW.avatar_id OR
OLD.role_role_id != NEW.role_role_id OR OLD.subscription_rank_subscription_rank_id !=
NEW.subscription_rank_subscription_rank_id THEN
   INSERT INTO audit_log (table_name, action, old_data, new_data, modified_at, user_id)
   VALUES ('users', 'update', JSON_OBJECT('email', OLD.email, 'username', OLD.username, 'avatar_id', OLD.avatar_id,
'role_role_id', OLD.role_role_id, 'subscription_rank_subscription_rank_id', OLD.subscription_rank_subscription_rank_id),
          JSON_OBJECT('email', NEW.email, 'username', NEW.username, 'avatar_id', NEW.avatar_id, 'role_role_id',
NEW.role_role_id, 'subscription_rank_subscription_rank_id', NEW.subscription_rank_subscription_rank_id), NOW(),
CURRENT_USER());
  END IF;
```

Код тригера image_data_audit_trigger

```
CREATE TRIGGER image_data_audit_trigger
AFTER UPDATE ON image_data
FOR EACH ROW
BEGIN
  IF OLD.image_bytes != NEW.image_bytes THEN
    INSERT INTO audit_log (table_name, action, old_data, new_data, modified_at, user_id)
    VALUES ('image_data', 'update', OLD.image_bytes, NEW.image_bytes, NOW(), CURRENT_USER());
  END IF;
```

Код тригера nodes_audit_trigger

```
CREATE TRIGGER nodes_audit_trigger
AFTER UPDATE ON nodes
FOR EACH ROW
BEGIN
  IF OLD.node_type != NEW.node_type OR OLD.text != NEW.text OR OLD.background_id != NEW.background_id OR
OLD.personage_personage_id != NEW.personage_personage_id OR OLD.text_background_id !=
NEW.text_background_id THEN
    INSERT INTO audit_log (table_name, action, old_data, new_data, modified_at, user_id)
    VALUES ('nodes', 'update', JSON_OBJECT('node_type', OLD.node_type, 'text', OLD.text, 'background_id',
OLD.background_id, 'personage_personage_id', OLD.personage_personage_id, 'text_background_id',
OLD.text_background_id),
          JSON_OBJECT('node_type', NEW.node_type, 'text', NEW.text, 'background_id', NEW.background_id,
'personage_personage_id', NEW.personage_personage_id, 'text_background_id', NEW.text_background_id), NOW(),
CURRENT_USER());
  END IF;
```

Код тригера edges_audit_trigger

```
CREATE TRIGGER edges_audit_trigger
AFTER UPDATE ON edges
FOR EACH ROW
BEGIN
  IF OLD.edge_type != NEW.edge_type OR OLD.source_node_node_id != NEW.source_node_node_id OR
OLD.target_node_node_id != NEW.target_node_node_id THEN
    INSERT INTO audit_log (table_name, action, old_data, new_data, modified_at, user_id)
    VALUES ('edges', 'update', JSON_OBJECT('edge_type', OLD.edge_type, 'source_node_node_id',
OLD.source_node_node_id, 'target_node_node_id', OLD.target_node_node_id),
          JSON_OBJECT('edge_type', NEW.edge_type, 'source_node_node_id', NEW.source_node_node_id,
'target_node_node_id', NEW.target_node_node_id), NOW(), CURRENT_USER());
  END IF;

CREATE TRIGGER accounts_audit_trigger
AFTER UPDATE ON accounts
FOR EACH ROW
BEGIN
  IF OLD.point_id != NEW.point_id THEN
    INSERT INTO audit_log (table_name, action, old_data, new_data, modified_at, user_id)
    VALUES ('accounts', 'update', OLD.point_id, NEW.point_id, NOW(), CURRENT_USER());
  END IF;
```

Код тригера users_audit_delete_trigger

```
CREATE TRIGGER users_audit_delete_trigger
AFTER DELETE ON users
FOR EACH ROW
BEGIN
  INSERT INTO audit_log (table_name, action, old_data, new_data, modified_at, user_id)
  VALUES ('users', 'delete', JSON_OBJECT('email', OLD.email, 'username', OLD.username, 'avatar_id', OLD.avatar_id,
'role_role_id', OLD.role_role_id, 'subscription_rank_subscription_rank_id', OLD.subscription_rank_subscription_rank_id),
        NULL, NOW(), CURRENT_USER());
END;
```

Код тригера users_audit_insert_trigger

```
CREATE TRIGGER users_audit_insert_trigger
AFTER INSERT ON users
FOR EACH ROW
BEGIN
  INSERT INTO audit_log (table_name, action, old_data, new_data, modified_at, user_id)
  VALUES ('users', 'insert', NULL,
        JSON_OBJECT('email', NEW.email, 'username', NEW.username, 'avatar_id', NEW.avatar_id, 'role_role_id',
NEW.role_role_id, 'subscription_rank_subscription_rank_id', NEW.subscription_rank_subscription_rank_id), NOW(),
CURRENT_USER());
END;
```

Код тригера image_data_audit_delete_trigger

```
CREATE TRIGGER image_data_audit_delete_trigger
AFTER DELETE ON image_data
FOR EACH ROW
BEGIN
  INSERT INTO audit_log (table_name, action, old_data, new_data, modified_at, user_id)
  VALUES ('image_data', 'delete', JSON_OBJECT('image_bytes', OLD.image_bytes),
        NULL, NOW(), CURRENT_USER());
END;
```

Код тригера image_data_audit_insert_trigger

```
CREATE TRIGGER image_data_audit_insert_trigger
AFTER INSERT ON image_data
FOR EACH ROW
BEGIN
  INSERT INTO audit_log (table_name, action, old_data, new_data, modified_at, user_id)
  VALUES ('image_data', 'insert', NULL,
        JSON_OBJECT('image_bytes', NEW.image_bytes), NOW(), CURRENT_USER());
END;
```

Код тригера nodes_audit_delete_trigger

```
CREATE TRIGGER nodes_audit_delete_trigger
AFTER DELETE ON nodes
FOR EACH ROW
BEGIN
  INSERT INTO audit_log (table_name, action, old_data, new_data, modified_at, user_id)
  VALUES ('nodes', 'delete', JSON_OBJECT('node_type', OLD.node_type, 'text', OLD.text, 'background_id',
OLD.background_id, 'personage_personage_id', OLD.personage_personage_id, 'text_background_id',
OLD.text_background_id),
        NULL, NOW(), CURRENT_USER());
END;
```

Код тригера nodes_audit_delete_trigger

```
CREATE TRIGGER nodes_audit_insert_trigger
AFTER INSERT ON nodes
FOR EACH ROW
BEGIN
  INSERT INTO audit_log (table_name, action, old_data, new_data, modified_at, user_id)
  VALUES ('nodes', 'insert', NULL,
```

```
        JSON_OBJECT('node_type', NEW.node_type, 'text', NEW.text, 'background_id', NEW.background_id,
'personage_personage_id', NEW.personage_personage_id, 'text_background_id', NEW.text_background_id), NOW(),
CURRENT_USER());
END;
```

Код тригера edges_audit_delete_trigger

```
CREATE TRIGGER edges_audit_delete_trigger
AFTER DELETE ON edges
FOR EACH ROW
BEGIN
  INSERT INTO audit_log (table_name, action, old_data, new_data, modified_at, user_id)
  VALUES ('edges', 'delete', JSON_OBJECT('edge_type', OLD.edge_type, 'source_node_node_id',
OLD.source_node_node_id, 'target_node_node_id', OLD.target_node_node_id),
        NULL, NOW(), CURRENT_USER());
END;
```

Код тригера edges_audit_insert_trigger

```
CREATE TRIGGER edges_audit_insert_trigger
AFTER INSERT ON edges
FOR EACH ROW
BEGIN
  INSERT INTO audit_log (table_name, action, old_data, new_data, modified_at, user_id)
  VALUES ('edges', 'insert', NULL,
        JSON_OBJECT('edge_type', NEW.edge_type, 'source_node_node_id', NEW.source_node_node_id,
'target_node_node_id', NEW.target_node_node_id), NOW(), CURRENT_USER());
END;
```

Код тригера accounts_audit_delete_trigger

```
CREATE TRIGGER accounts_audit_delete_trigger
AFTER DELETE ON accounts
FOR EACH ROW
BEGIN
  INSERT INTO audit_log (table_name, action, old_data, new_data, modified_at, user_id)
  VALUES ('accounts', 'delete', OLD.point_id,
        NULL, NOW(), CURRENT_USER());
END;

CREATE TRIGGER accounts_audit_insert_trigger
AFTER INSERT ON accounts
FOR EACH ROW
BEGIN
  INSERT INTO audit_log (table_name, action, old_data, new_data, modified_at, user_id)
  VALUES ('accounts', 'insert', NULL,
        NEW.point_id, NOW(), CURRENT_USER());
END;
```

ДОДАТОК В
КОД СТВОРЕННЯ БАЗИ ДАНИХ

```
use Server_DB;

CREATE TABLE image_data (
  id bigint IDENTITY(1,1) NOT NULL,
  image_bytes VARBINARY(MAX),
  PRIMARY KEY (id)
)
CREATE TABLE conditions (
  condition_id bigint IDENTITY(1,1) NOT NULL,
  condition_content varchar(255) NOT NULL,
  PRIMARY KEY (condition_id)
)


CREATE TABLE points (
  point_id bigint IDENTITY(1,1) NOT NULL,
  name varchar(255) NOT NULL,
  points int NOT NULL,
  PRIMARY KEY (point_id)
)

CREATE TABLE points_a (
  point_id bigint NOT NULL,
  PRIMARY KEY (point_id),
  CONSTRAINT FK_points_a_points FOREIGN KEY (point_id) REFERENCES points (point_id)
)

CREATE TABLE points_b (
  point_id bigint NOT NULL,
  PRIMARY KEY (point_id),
  CONSTRAINT FK_points_b_points FOREIGN KEY (point_id) REFERENCES points (point_id)
```

```
)
CREATE TABLE points_count (
  points_count_id bigint IDENTITY(1,1) NOT NULL,
  pointa_point_id bigint NULL,
  pointb_point_id bigint NULL,
  PRIMARY KEY (points_count_id),
  CONSTRAINT FK_points_count_points_a FOREIGN KEY (pointa_point_id) REFERENCES points_a (point_id),
  CONSTRAINT FK_points_count_points_b FOREIGN KEY (pointb_point_id) REFERENCES points_b (point_id)
)

CREATE TABLE roles (
  role_id int IDENTITY(1,1) NOT NULL,
  role_name varchar(255) NULL,
  PRIMARY KEY (role_id)
)

CREATE TABLE stories (
  story_id int IDENTITY(1,1) NOT NULL,
  cover varchar(255) NULL,
  description varchar(255) NULL,
  genre varchar(255) NULL,
  price decimal(10,2) NULL,
  story varchar(255) NULL,
  title varchar(255) NULL,
  cover_id bigint NULL,
  PRIMARY KEY (story_id),
  CONSTRAINT FK_stories_image_data FOREIGN KEY (cover_id) REFERENCES image_data (id)
)
CREATE TABLE accounts (
  point_id bigint NOT NULL,
  PRIMARY KEY (point_id),
  CONSTRAINT FK_accounts_points FOREIGN KEY (point_id) REFERENCES points (point_id)
)

CREATE TABLE personages (
  personage_id bigint IDENTITY(1,1) NOT NULL,
  emotion varchar(255) NOT NULL,
```

```
  name varchar(255) NOT NULL,
  side varchar(255) NOT NULL,
  image_id bigint NULL,
  PRIMARY KEY (personage_id),
  CONSTRAINT FK_personages_image_data FOREIGN KEY (image_id) REFERENCES image_data (id)
)
CREATE TABLE subscription_rank (
  subscription_rank_id int IDENTITY(1,1) NOT NULL,
  max_available_stories int NULL,
  name varchar(255) NULL,
  price decimal(10,2) NULL,
  title_id bigint NULL,
  PRIMARY KEY (subscription_rank_id),
  CONSTRAINT FK_subscription_rank_image_data FOREIGN KEY (title_id) REFERENCES image_data (id)
)
CREATE TABLE options (
  option_id bigint IDENTITY(1,1) NOT NULL,
  position_id int NOT NULL,
  text varchar(255) NOT NULL,
  condition_condition_id bigint NULL,
  PRIMARY KEY (option_id),
  CONSTRAINT FK_options_conditions FOREIGN KEY (condition_condition_id) REFERENCES conditions
(condition_id)
)
CREATE TABLE nodes (
  node_id bigint IDENTITY(1,1) NOT NULL,
  node_type varchar(255) NOT NULL,
  text varchar(255) NOT NULL,
  background_id bigint NULL,
  personage_personage_id bigint NULL,
  text_background_id bigint NULL,
  PRIMARY KEY (node_id),
  CONSTRAINT FK_nodes_image_data_background FOREIGN KEY (background_id) REFERENCES image_data (id),
  CONSTRAINT FK_nodes_image_data_text_background FOREIGN KEY (text_background_id) REFERENCES
image_data (id),
  CONSTRAINT FK_nodes_personages_personage FOREIGN KEY (personage_personage_id) REFERENCES personages
(personage_id)
)
CREATE TABLE edges (
```

```
  edge_id bigint IDENTITY(1,1) NOT NULL,
  edge_type varchar(255) NOT NULL,
  source_node_node_id bigint NOT NULL,
  target_node_node_id bigint NOT NULL,
  PRIMARY KEY (edge_id),
  CONSTRAINT FK_edges_nodes_source FOREIGN KEY (source_node_node_id) REFERENCES nodes (node_id),
  CONSTRAINT FK_edges_nodes_target FOREIGN KEY (target_node_node_id) REFERENCES nodes (node_id)
)


CREATE TABLE text_nodes (
  node_id bigint NOT NULL,
  PRIMARY KEY (node_id),
  CONSTRAINT FK_text_nodes_nodes FOREIGN KEY (node_id) REFERENCES nodes (node_id)
)

CREATE TABLE text_edges (
  edge_id bigint NOT NULL,
  PRIMARY KEY (edge_id),
  CONSTRAINT FK_text_edges_edges FOREIGN KEY (edge_id) REFERENCES edges (edge_id)
)

CREATE TABLE two_choices_nodes (
  node_id bigint NOT NULL,
  option1_option_id bigint NULL,
  option2_option_id bigint NULL,
  PRIMARY KEY (node_id),
  CONSTRAINT FK_two_choices_nodes_options1 FOREIGN KEY (option1_option_id) REFERENCES options
(option_id),
  CONSTRAINT FK_two_choices_nodes_options2 FOREIGN KEY (option2_option_id) REFERENCES options
(option_id),
  CONSTRAINT FK_two_choices_nodes_nodes FOREIGN KEY (node_id) REFERENCES nodes (node_id)
)

CREATE TABLE three_choices_nodes (
  node_id bigint NOT NULL,
  option3_option_id bigint NULL,
  PRIMARY KEY (node_id),
```

```
  CONSTRAINT FK_three_choices_nodes_options3 FOREIGN KEY (option3_option_id) REFERENCES options (option_id),
  CONSTRAINT FK_three_choices_nodes_two_choices_nodes FOREIGN KEY (node_id) REFERENCES two_choices_nodes (node_id)
)
CREATE TABLE choice_edges (
  edge_id bigint NOT NULL,
  option_option_id bigint NULL,
  PRIMARY KEY (edge_id),
  CONSTRAINT FK_choice_edges_options FOREIGN KEY (option_option_id) REFERENCES options (option_id),
  CONSTRAINT FK_choice_edges_edges FOREIGN KEY (edge_id) REFERENCES edges (edge_id)
)

CREATE TABLE subscription_rank_available_stories (
  subscription_rank_subscription_rank_id int NOT NULL,
  available_stories_story_id int NOT NULL,
  PRIMARY KEY (subscription_rank_subscription_rank_id, available_stories_story_id),
  CONSTRAINT FK_subscription_rank_available_stories_subscription_rank FOREIGN KEY (subscription_rank_subscription_rank_id) REFERENCES subscription_rank (subscription_rank_id),
  CONSTRAINT FK_subscription_rank_available_stories_stories FOREIGN KEY (available_stories_story_id) REFERENCES stories (story_id)
)

CREATE TABLE users (
  id int IDENTITY(1,1) NOT NULL,
  email varchar(255) NULL,
  password varchar(255) NULL,
  username varchar(255) NULL,
  avatar_id bigint NULL,
  role_role_id int NULL,
  subscription_rank_subscription_rank_id int NULL,
  PRIMARY KEY (id),
  CONSTRAINT FK_users_image_data_avatar FOREIGN KEY (avatar_id) REFERENCES image_data (id),
  CONSTRAINT FK_users_roles FOREIGN KEY (role_role_id) REFERENCES roles (role_id),
  CONSTRAINT FK_users_subscription_rank FOREIGN KEY (subscription_rank_subscription_rank_id) REFERENCES subscription_rank (subscription_rank_id)
)
CREATE TABLE users_stories (
  story_id int IDENTITY(1,1) NOT NULL,
```

```
  name varchar(255) NOT NULL,
  entry_point_node_id bigint NULL,
  original_story_story_id int NULL,
  points_points_count_id bigint NULL,
  PRIMARY KEY (story_id),
  CONSTRAINT FK_users_stories_entry_point_node FOREIGN KEY (entry_point_node_id) REFERENCES nodes (node_id),
  CONSTRAINT FK_users_stories_original_story FOREIGN KEY (original_story_story_id) REFERENCES stories (story_id),
  CONSTRAINT FK_users_stories_points_points_count FOREIGN KEY (points_points_count_id) REFERENCES points_count (points_count_id)
)

CREATE TABLE users_available_stories (
  user_id int NOT NULL,
  available_stories_story_id int NOT NULL,
  PRIMARY KEY (user_id, available_stories_story_id),
  CONSTRAINT FK_users_available_stories_user FOREIGN KEY (user_id) REFERENCES users (id),
  CONSTRAINT FK_users_available_stories_stories FOREIGN KEY (available_stories_story_id) REFERENCES stories (story_id)
)

CREATE TABLE users_stories_edges (
  users_story_story_id int NOT NULL,
  edges_edge_id bigint NOT NULL,
  PRIMARY KEY (users_story_story_id, edges_edge_id),
  CONSTRAINT FK_users_stories_edges_users_story FOREIGN KEY (users_story_story_id) REFERENCES users_stories (story_id),
  CONSTRAINT FK_users_stories_edges_edges FOREIGN KEY (edges_edge_id) REFERENCES edges (edge_id)
)

CREATE TABLE users_users_stories (
  user_id int NOT NULL,
  users_stories_story_id int NOT NULL,
  PRIMARY KEY (user_id, users_stories_story_id),
  CONSTRAINT FK_users_users_stories_user FOREIGN KEY (user_id) REFERENCES users (id),
  CONSTRAINT FK_users_users_stories_users_stories FOREIGN KEY (users_stories_story_id) REFERENCES users_stories (story_id)
```

ПРОТОКОЛ ПЕРЕВІРКИ
БАКАЛАВРСЬКОЇ ДИПЛОМНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Засіб поширення візуальної новели. Частина 2. Захищені бази даних

Автор роботи: Савчук Іван Борисович

Тип роботи: бакалаврська дипломна робота .
(БДР, МКР)

Підрозділ кафедра захисту інформації ФІТКІ .
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність – 98,2%. Схожість – 1,8%.

Аналіз звіту подібності (відмітити потрібне):

√ 1. Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

□ 2. Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

□ 3. Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Каплун В. А.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Савчук І. Б.

Керівник роботи _____ Бацманов Ю. В.

ІЛЮСТРАТИВНА ЧАСТИНА

ЗАСІБ ПОШИРЕННЯ ВІЗУАЛЬНОЇ НОВЕЛИ. ЧАСТИНА 2. ЗАХИЩЕНІ БАЗИ ДАНИХ

(Назва комплексної бакалаврської дипломної роботи)

Виконав: студент 4 курсу групи 1БС-19б
Спеціальності 125 Кібербезпека
_____ Іван САВЧУК

Керівник: к. т. н., доцент, доцент каф ЗІ
_____ Юрій БАРИШЕВ
« 16 » червня 2023 р.

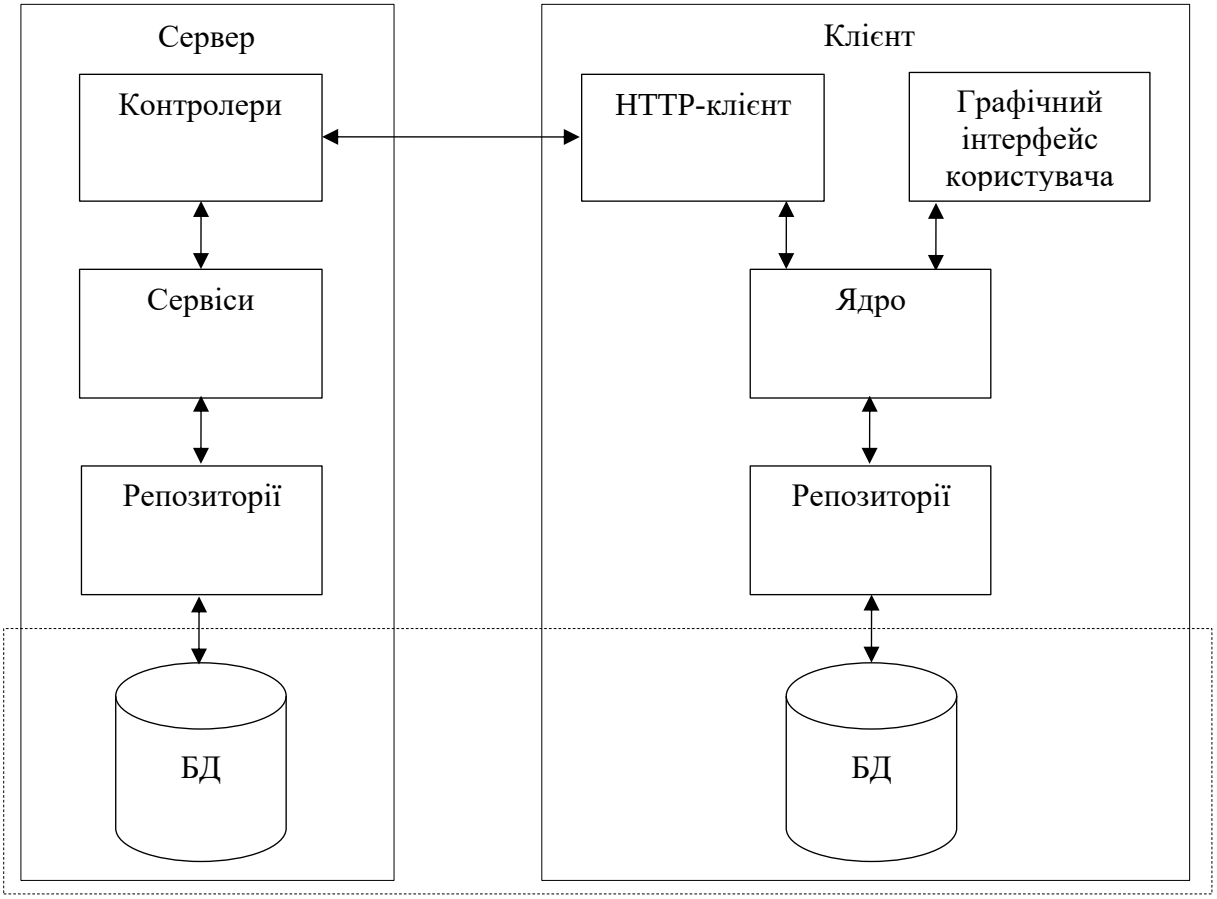
Аналіз загроз комп’ютерним іграм

Тип загрози	Опис	Потенційні наслідки
Вторгнення	Несанкціонований доступ до ігрової системи або мережі	Крадіжка особистих даних гравців, втрата конфіденційності
Шахрайство з даними	Крадіжка акаунтів, валюти або інших важливих ігрових даних	Втрата власності гравців, фінансові втрати
Віруси та шкідливі програми	Використання шкідливого програмного забезпечення в ігровому середовищі	Пошкодження системи, втрата даних, погіршення продуктивності
Чити та шахрайство у грі	Використання нечесних методів для отримання переваг у грі	Нерівні умови гри, знецінення досягнень, втрата задоволення
DDoS-атаки	Спроби перевантаження серверів гри, щоб зупинити доступ гравців	Втрата доступу до гри, збитки від зупинки сервісу
Завдання шкоди гравцям	Цілеспрямовані дії з метою завдання шкоди іншим гравцям	Втрата гри, погіршення геймплею, конфлікти між гравцями

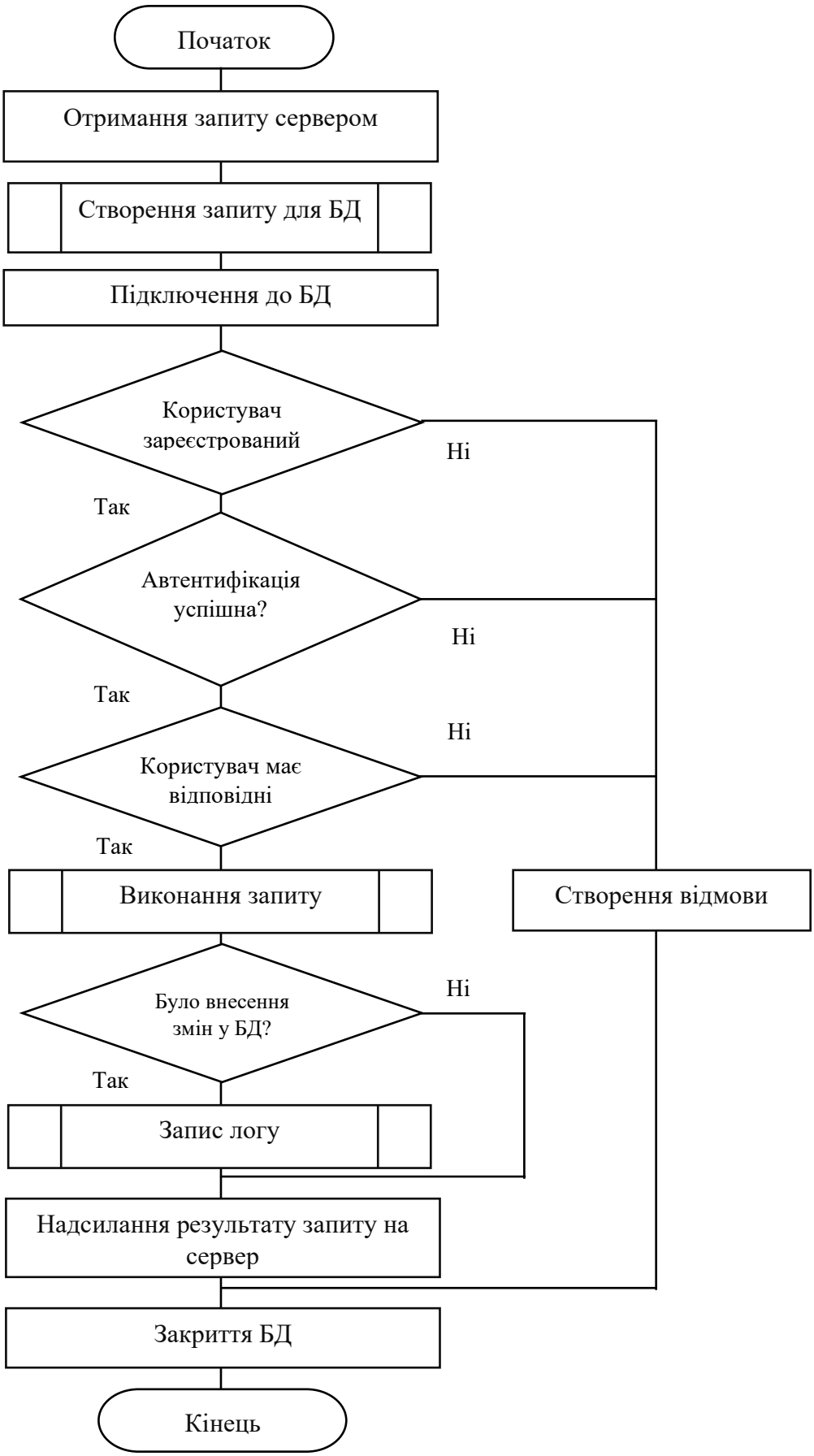
Аналіз методів захисту баз даних

Метод захисту баз даних	Опис	Переваги	Недоліки
Аутентифікація	Встановлення ідентифікації та перевірка користувачів бази даних	Запобігає несанкціонованому доступу до бази даних	Вимагає ефективного управління обліковими записами користувачів
Авторизація	Контроль доступу користувачів до конкретних ресурсів бази даних	Забезпечує обмеження прав доступу до конфіденційних даних	Вимагає точного визначення рівнів доступу і прав користувачів
Шифрування	Застосування алгоритмів шифрування для захисту даних	Забезпечує конфіденційність інформації в базі даних	Може вплинути на продуктивність системи
Резервне копіювання	Створення резервних копій бази даних для відновлення	Забезпечує можливість відновлення даних у випадку втрати	Вимагає достатнього простору для зберігання резервних копій
Моніторинг	Систематичний контроль та аналіз активності в базі даних	Виявляє підозрілу або несанкціоновану діяльність	Вимагає постійного нагляду та ресурсів для аналізу
Аудит	Запис та аналіз активності користувачів у базі даних	Виявляє аномальну або несанкціоновану активність	Вимагає додаткового зберігання і обробки аудиторської інформації

Узагальнена архітектура засобу



Узагальнений алгоритм роботи бази даних



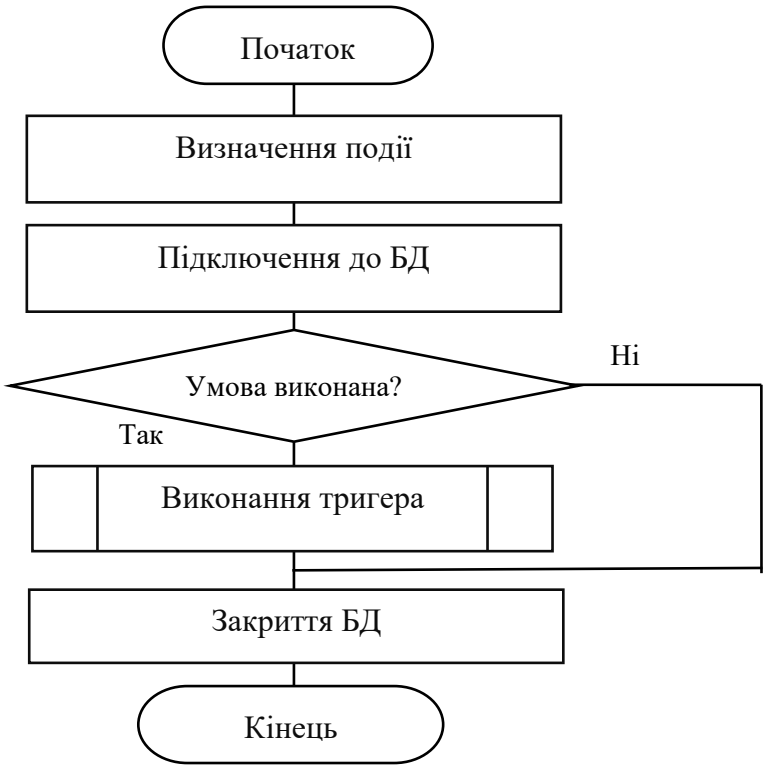
Модель розмежування прав доступу

Роль	Опис	Дозволи	Заборони
SceneEditor	Має повний доступ до бази даних.	UPDATE, INSERT, DELETE (nodes,edges,image_data)	N/A
TextWriteEditor	Можна змінювати лише текстові дані.	UPDATE, INSERT (nodes)	N/A
ImageEditor	Можна змінювати лише графічні дані.	UPDATE, INSERT, DELETE (image_data)	N/A
Viewer	Може читати дані в усіх таблицях	SELECT (All tables)	UPDATE, INSERT, DELETE (Всі таблиці)

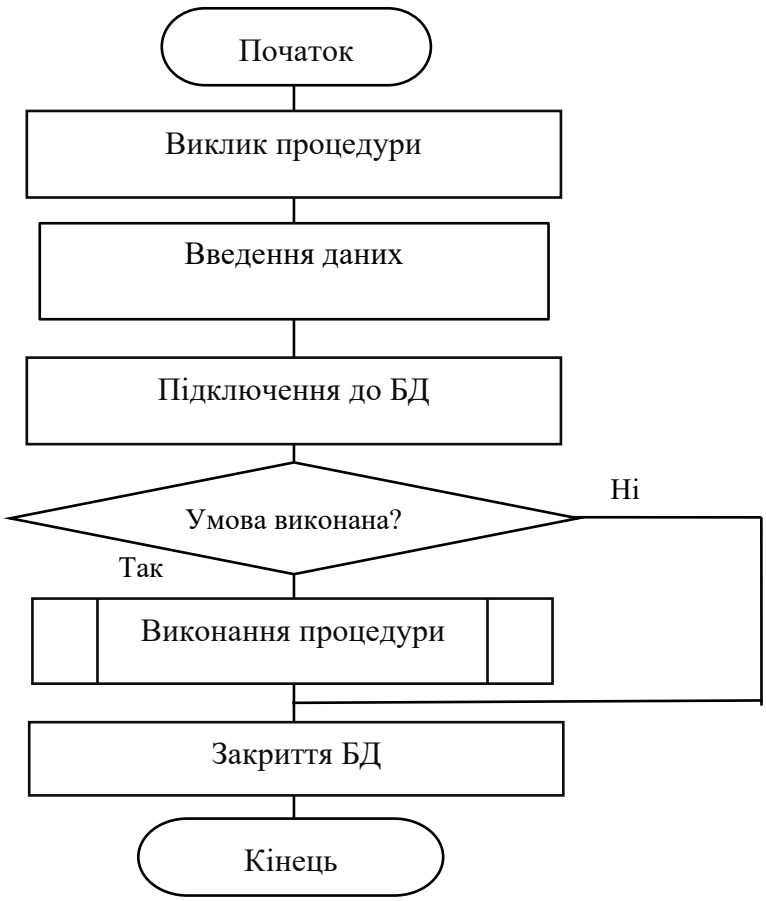
Розмежування прав доступу розробників

Роль	Опис	Дозволи	Заборони
Platinum	Може читати всі безкоштовні та платні історії	SELECT	N/A
V.I.P	Може читати всі безкоштовні історії, а також 6 додаткових платних історій, має отримує персональну підтримку	SELECT	Доступ до даних ролі id 6
Premium+	Можна читати всі безкоштовні історії, а також 2 додаткові платні історії на вибір	SELECT	Доступ до даних ролей id 5,6
Premium	Можна читати всі безкоштовні історії, а також 1 додаткова платна історія	SELECT	Доступ до даних ролей id 4-6
Gamer	Можна читати всі безкоштовні історії	SELECT	Доступ до даних ролей id 3-6
Guest	Має доступ лише до обмеженого вибору безкоштовних історій і функцій і не може залишати коментарі чи оцінки	SELECT	Доступ до даних ролей role id 2-6

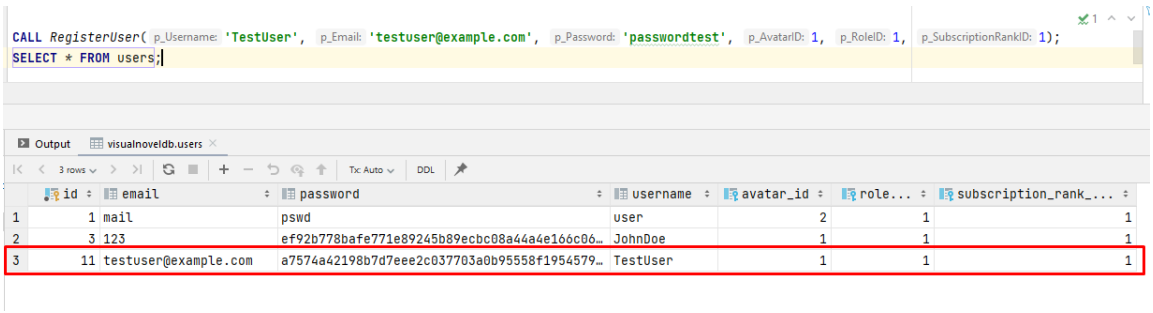
Алгоритм захисту даних на стороні сервера



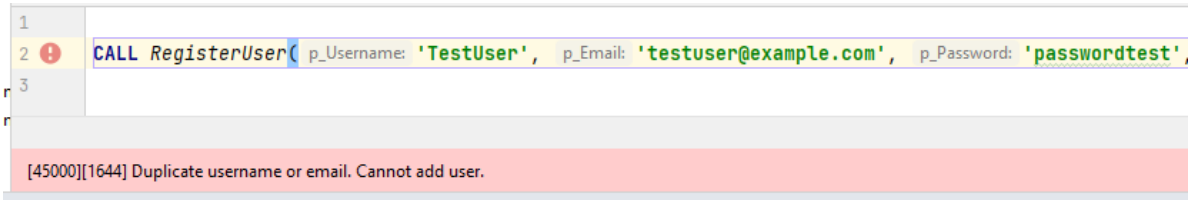
Алгоритм роботи тригера



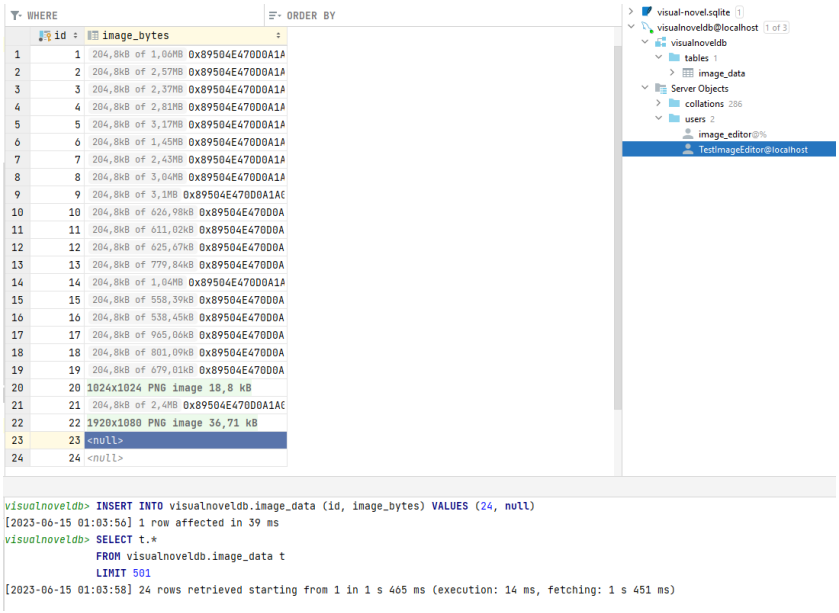
Результат тестування процедури RegisterUser



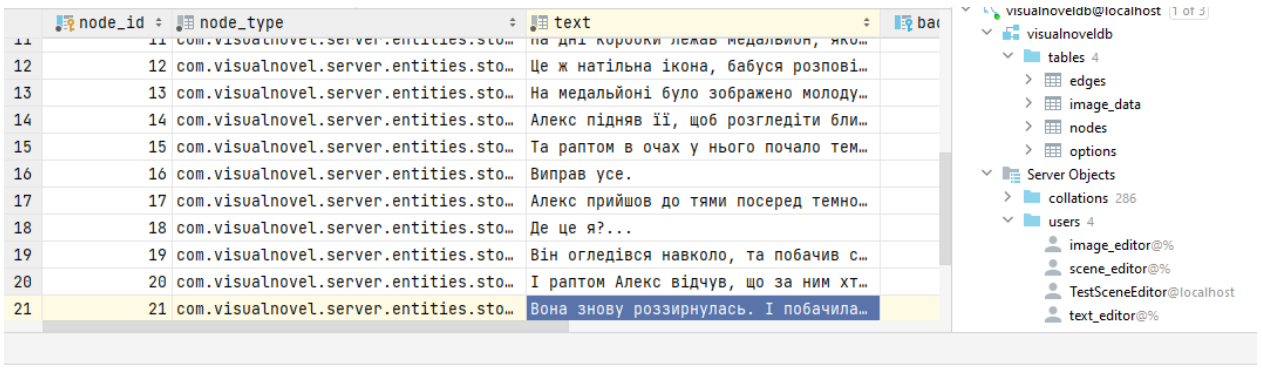
Результат тестування тригера PreventDuplicateUsers



Результат перевірки доступу користувача до таблиці image_data



Результат перевірки доступу користувача TestSceneEditor до таблиць

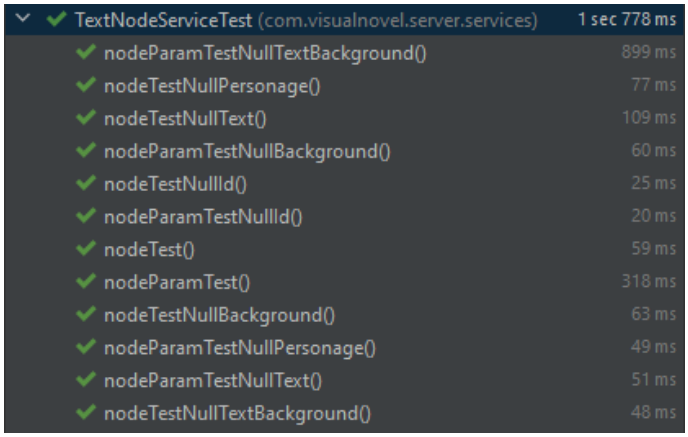


	node_id	node_type	text	ba
11	11	com.visualnovel.server.entities.sto…	На дні коробки лежав медальйон, яко…	
12	12	com.visualnovel.server.entities.sto…	Це ж натільна ікона, бабуся розпові…	
13	13	com.visualnovel.server.entities.sto…	На медальйоні було зображено молоду…	
14	14	com.visualnovel.server.entities.sto…	Алекс підняв її, щоб розгледіти бли…	
15	15	com.visualnovel.server.entities.sto…	Та раптом в очах у нього почало тем…	
16	16	com.visualnovel.server.entities.sto…	Виправ усе.	
17	17	com.visualnovel.server.entities.sto…	Алекс прийшов до тями посеред темно…	
18	18	com.visualnovel.server.entities.sto…	Де це я?...	
19	19	com.visualnovel.server.entities.sto…	Він огледівся навколо, та побачив с…	
20	20	com.visualnovel.server.entities.sto…	І раптом Алекс відчув, що за ним хт…	
21	21	com.visualnovel.server.entities.sto…	Вона знову роззирнулась. І побачила…	

visualnoveldb@localhost 1 of 3
visualnoveldb
tables 4
edges
image_data
nodes
options
Server Objects
collations 286
users 4
image_editor@%
scene_editor@%
TestSceneEditor@localhost
text_editor@%

```
visualnoveldb> UPDATE visualnoveldb.nodes t SET t.text = 'Вона знову роззирнулась. І побачила стежину.' WHERE t.node_id = 21
[2023-06-15 01:20:38] 1 row affected in 24 ms
visualnoveldb> SELECT t.*
               FROM visualnoveldb.nodes t
               LIMIT 501
[2023-06-15 01:20:39] 21 rows retrieved starting from 1 in 63 ms (execution: 11 ms, fetching: 52 ms)
```

Результати виконання тесту nodes

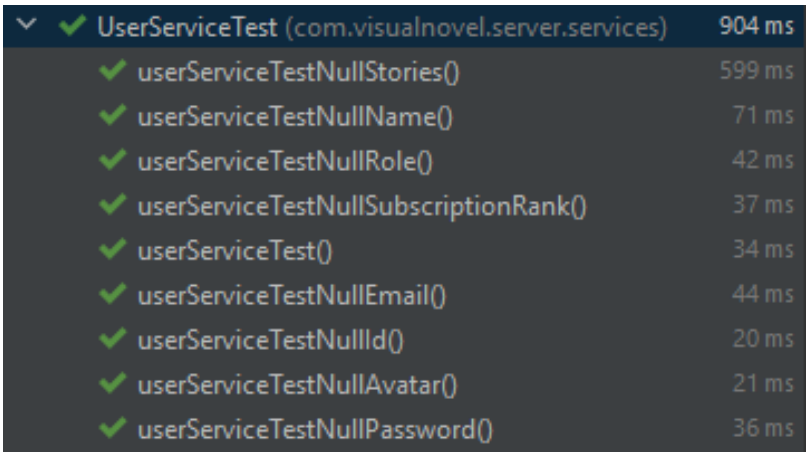


Результати заповнення таблиці nodes

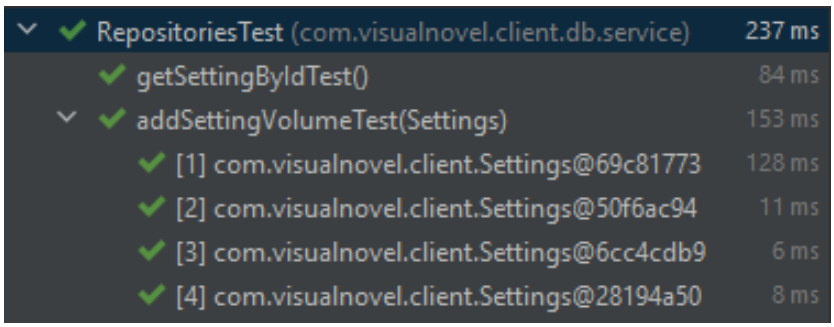
WHERE ORDER BY

	node_id	node_type	text	b...	pers...	text_background_id
22	100	com.visualnovel.…	100TextNode	21	10	21
23	101	com.visualnovel.…	100TextNode	21	10	21
24	102	com.visualnovel.…	100TextNode	21	10	21
25	103	com.visualnovel.…	102 TextNode	21	12	21
26	104	com.visualnovel.…	103 TextNode	22	10	21
27	105	com.visualnovel.…	104 TextNode	21	10	2

Результати виконання тесту Users

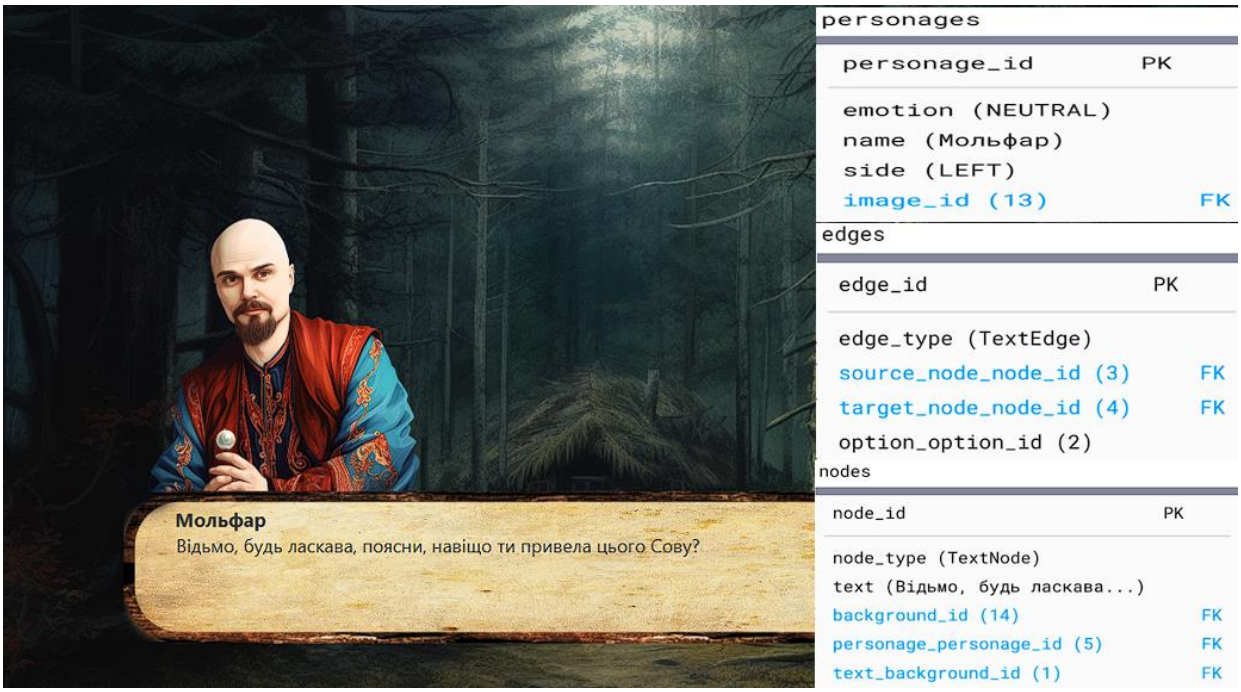


Результати виконання тесту Settings



Результат зміни даних в другій діалоговій сцені

Результат зміни даних в загальній сцені



Результат зміни персонажа в загальній сцені