Хмельницький національний університет
Факультет інформаційних технологій
Кафедра кібербезпеки та комп'ютерних систем та мереж

КВАЛІФІКАЦІЙНА РОБОТА

бакалавр
Освітній рівень

Побудова децентралізованої захищеної бази даних з використанням криптографічного шифрування
Назва теми

КПКБ. 180131.18.01.07 КБ
Шифр

Галузь знань *12 «Інформаційні технології»*
Шифр, назва

Спеціальність *125 «Кібербезпека»*
Шифр, назва

Спеціалізація *Кібербезпека*

Виконав: студент 4 курсу, група КБ-18-1 _____ О.В. Нагребецький
Підпис Ініціали, прізвище

Керівник _____ Шітова В.М.
Підпис, дата Ініціали, прізвище

Нормоконтролер 09.06.22 Косновий С.В.
Підпис, дата Ініціали, прізвище

До захисту допускаю:
Зав. кафедри кібербезпеки,
та комп'ютерних систем
і мереж _____ Кльоц Ю.П.
Підпис, дата Ініціали, прізвище

13 06 2022р.

Хмельницький, 2022

№ рядка	формат	Позначення	Найменування	Кіл. листів	№ екз	Примітка
			Текстові документи			
1	А4	КПКБ. 180131.18.01.07 ПЗ	Пояснювальна записка	63		
			Графічні матеріали			
2	А4	КПКБ. 180131.18.01.07	Схема генерування хешу	1		
3	А4	КПКБ. 180131.18.01.07	Повноцінна схема взаємодії та обміну даних	1		
4	А4	КПКБ. 180131.18.01.07	Модель опрацьовування бази даних-	1		
5	А4	КПКБ. 180131.18.01.07	Модель порушника	1		

КПКБ. 180131.18.01.07 ВП

Зм	Арк	№ докум	Підпис	Дата	Побудова децентралізованої захищеної бази даних з використанням криптографічного шифрування	Літера	Аркуш	Аркушів
						У	1	1
Розробив		Нагребецький О.В..				ХНУ, КБ-18-1		
Перевір.		Тітова В.Ю.						
Н. контр.		Мостовий С.В		09.06.22				
Затв.		Кльоц Ю. П.		13.06.22				

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра КІБЕРБЕЗПЕКИ

Освітній рівень БАКАЛАВР

Галузь знань 12 ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

Спеціальність 125 КІБЕРБЕЗПЕКА

Освітня програма ОСВІТНЬО-ПРОФЕСІЙНА ПРОГРАМА ПІДГОТОВКИ БАКАЛАВРІВ

ЗАТВЕРДЖУЮ

Зав. кафедри Ю.П. Кльоц

"1" 03 2022 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Нагребецькому О.В.
Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Побудова децентралізованої захищеної бази даних з використанням криптографічного шифрування

Керівник проєкту (роботи) Тітова В. Ю.
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

2. Строк подання студентом проекту (роботи) на кафедру 02.06.2022

3. Вихідні дані до проекту (роботи): створення інформаційної захищеної системи із можливістю передачі даних у відкритий простір мережі інтернет, створення алгоритму для збереження інформації.

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити) провести аналіз можливих рішень завдання, опрацювати моделі загроз та порушника, розробити план для подолання загальних загроз, проаналізувати можливі загрози і ризики, провести симуляцію або моделювання загрози при кібератаці, опрацювати вже існуючі програмні рішення, створити систему контролю доступу, провести оцінку доцільності та вартості використання системи, створити додаткове програмне забезпечення для реалізації API – запитів.

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень) «Схема генерування хешу», «Модель порушника», «Модель опрацьовування бази даних», «Повна схема взаємодії та обміну даних»

6. Консультанти розділів курсового проєкту (роботи)

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

Нормоконтроль	Мостовий С.В.	–	
Антиплагіат	Мостовий С.В.	–	

7. Дата видачі завдання «2» лютого 2022р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) курсового проєкту (роботи)	Термін виконання етапів проекту (роботи)	Примі
1	Ознайомлення з предметною областю	6.02	–
2	Пошук теоретичної інформації про предметну область	12.02	–
3	Проведення аналізу даних та пошук підходу до реалізації моделі проектування	26.02	–
4	Початок створення моделі загроз та моделі порушника	4.03	–
5	Створення методів аутинтефікації	12.03	–
6	Створення нової бази даних	15.03	
7	Створення алгоритму шифрування для збереження інформації	21.03	–
8	Створення керуючого інтерфейсу із необхідним функціоналом	4.04	
9	Створення додаткового програмного забезпечення для реалізації API - запитів	8.04	–
10	Додавання додаткового методу захисту опрацювання запитів внутрішньої системи	14.04	–
11	Проведення тестового випробування системи	17.04	–
12	Завершення системи та внесення змін до зовнішнього оформлення	26.04	
13	Оформлення пояснювальної записки згідно вимог	8.05	–
14	Оформлення графічної частини	23.05	–
15	Захист КР	03.06	–

Студент Підпис Нагребецький О.В. Ініціали, прізвище

Керівник проекту (роботи) Підпис Мостовий В.В. Ініціали, прізвище

№ рядка	формат	
1	А4	КПК
2	А4	КПК
3	А4	КП
4	А4	КП
5	А4	КП

Зм Арк № док

Розробив Нагребецький О.В.

Перевір. Тітова

Анотація

Тема кваліфікаційної роботи: "Побудова децентралізованої захищеної бази даних з використанням криптографічного шифрування"

Автор роботи: Нагребецький О.В.

Керівник роботи:

Обсяг – 63 сторінок, 22 рис., 1 таблиця, 36 джерел, 4 додатки.

МОДЕЛІ, МЕТОДИ, БАЗА ДАНИХ, КРИПТОГРАФІЧНЕ ШИФРУВАННЯ, ІНФОРМАЦІЙНА СИСТЕМА, КРИПТОГРАФІЧНИЙ ЗАХИСТ, API – ЗАПИТИ.

Метою роботи є створення захищеної бази даних, із інтерфейсом користувача, можливістю передачі інформації та віддаленого опрацювання даних.

У даному проекті проведено аналіз предметної області та існуючих рішень для вирішення подібних задач, створено програмне забезпечення для користувачів та поставлених задач. Створено користувацький інтерфейс, захищену бази даних із використанням алгоритмів шифрування, методи зовнішньої взаємодії із системою. В якості середовища розробки програмного продукту використано PHP Storm та мову PHP v.7.4. За допомогою цих засобів було розроблено програмне забезпечення, продукт має багатофакторний захист для збереження та використання інформації.

Були проаналізовані можливі загрози та створено окремі моделі. Використаний метод шифрування відповідає до стандарту SHA – 1. Додаткові програмні забезпечення, що були використані під час створення програмного продукту: Postman API, PHP Artisan, Sanctum, Telegram, Laravel Fortify.

9.06.2022

Annotation

Topic of the qualification qualification work: "Building a decentralized secure database with the establishment of cryptographic encryption"

Author of the work: O.V. Nahrebetskyi.

Supervisor of Work:

Scope – 63 pages, 22 images, 1 table, 36 references, 4 adds

MODELS, METHODS, DATA BASE, CRYPTOGRAPHIC ENCRYPTION, INFORMATION SYSTEM, CRYPTOGRAPHIC PROTECTION, ARCI - NOTES.

The aim is to create a secure database, with a user interface, the ability to transmit information and remote processing of data.

This project analyzed the subject area and existing solutions for solving similar problems, and created a software for the user and the set tasks. Created correlation interface, secured database using encryption algorithms, methods of external interaction with the system. As a medium for the development of the software product used PHP Storm and PHP language v.7.4. With the help of these tools was developed software, the product has a big factor protection for the preservation and use of information.

Potential threats were analyzed and certain models were created. Used encryption method meets the standard SHA - 1. Additional software that was used during the creation of the software product: Postman API, PHP Artisan, Sanctum, Telegram, Laravel Fortify.

9.06.2022

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ 4

ВСТУП 6

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 9

1.1 Збір інформації та первинне дослідження вже існуючих систем для збереження інформації 9

2.1 Пошук існуючих методів розподілу рівнів доступу до інформації згідно існуючих стандартів. 16

1.2 Пошук та аналіз інформаційних загроз під час опрацьовування БД 19

1.3 Виконання функціоналу програми 21

1.4 Постановка задачі проектування 23

2 ПРЕДПРОЕКТНИЙ АНАЛІЗ НЕОБХІДНИХ СКЛАДОВИХ ДЛЯ ПРОЕКТУ ТА ЇХ ОБЛАШТУВАННЯ 25

2.1 Аналіз необхідних складових виходячи зі необхідного функціоналу програми 25

2.2 Остаточне налаштування та підключення робочого середовища для проекту 27

3 СИНТЕЗ КОМПЛЕКСНОЇ СИСТЕМИ ЗАХИСТУ ІНФОРМАЦІЇ 36

3.1 Створення захищеної бази даних 36

3.2 Обгрунтування та вибір СКБД 38

3.3 Структурний опис побудованої системи користувачів 43
3.4 Опис діалогу роботи автоматизованої системи 44
3.5 Реалізація API – запитів програми 47
3.6 Висновки про створення програмне забезпечення 52
4 ТЕСТУВАННЯ СТВОРЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 55
4.1 Пояснення резонності проведення тестових атак на створене ПО 55
4.2 SQL ін’єкція в систему 56
4.3 Приблизний вигляд можливих кібератак через користувачів 58
4.4 Проведення пробних атак на створений сервіс за домопогою сторонніх програм 59
ВИСНОВКИ 62
СПИСОК ДЖЕРЕЛ 63

ПЕРЕЛІК СКОРОЧЕНЬ

1. БД – база даних
2. *СУБД* – система управління базами даних.
3. SQL – “structured query language”, мова структурованих запитів, безпосередньо використовується для керування та взаємодії із базою даних.
4. GUI (ГІК) – графічний користувацький інтерфейс, або графічний інтерфейс користування.
5. ООСКБД – об’єктно орієнтовані системи керування базою даних.
6. 4GL – Fourth Generation Language – мова четвертого покоління для опрацьовування баз даних.
7. FTP - File Transfer Protocol – протокол передачі даних по інтернет сітці
8. HTML - стандартизована мова, що створена для розмітки документів та перегляду веб-сторінок у браузері
9. HTTP - HyperText Transfer Protocol – незахищений сертифікацією протокол передачі даних, що використовується в комп'ютерних мережах.
10. HTTPS - HyperText Transfer Protocol Secure – захищений окремо створеним сертифікатом унікальності та підтвердження, протокол передачі даних, що використовується в комп'ютерних мережах
11. IP - *Internet Protocol* — міжмережевий протокол
12. TCP - *Transmission Control Protocol* - протокол керування передачі
13. SMTP - Simple Mail Transfer Protocol – простий протокол передачі пошти
14. CGI - *Common Gateway Interface – загальний шлюзний інтерфейс*
15. ПЗ – програмне забезпечення
16. Ajax - Asynchronous Javascript and XML – мова програмування back – end
17. DHTML - Dynamic HTML — концепція створення вебсайту, що розглядає HTML-документ як об'єктну структуру, використовує поєднання статичної мови розмітки HTML та інших об’єктів створеного веб-застосунку.

18. IOS - це власницька мобільна операційна система від Apple
19. LAN – локально – обчислювальна мережа
20. RiSC - Reduced Instruction Set Computing — обчислення зі скороченим набором команд

ВСТУП

Будь-яке збереження інформації, змушує підняти питання про їх захищеність від сторонніх. Адже, небезпека підстерігає нас всюди та постійно, вже не кажучи про ненадійність надання особистої інформації до сумнівного інтернет ресурсу, оплати покупок карткою, до використання локальної мережі на роботі чи вдома, тощо. Не існує абсолютного захисту, що здатен відбити будь-яку кібератаку. Прогрес – це постій процес та головний противник кібербезпеки, бо завжди знайдуться нові методи взлому чи хтось виявить чергову вразливість програмного коду.

У наш час, електронні – інформаційні пристрої та системи - заполонили кожну шпарину нашого життя. Той самий холодильник нового покоління, що підключений до мережі інтернет, може нести в собі загрозу зовнішнього вторгнення. Звичайно, ніхто вже не може відмовитись від комп'ютера чи смартфону, тому, й питання захисту – є постійним. Якщо відмовитись від сучасного телефону, скільки часу людина витрачатиме для того, щоб поспілкуватись із друзями чи знаходити потрібну інформацію?

Для збереження конфіденційності, цілісності данних, обмеженню доступу від сторонніх вторгнень, були створені безліч програмно-апаратних пристроїв, що виконують такі:

- Функції моніторингу даних;
- Шифрування та недоступніть;
- Ідентифікація, перехоплення та відсторонення небажаних з'єднань;
- Методи кібергігієни;
- тощо.

Кожен, абсолютно кожен, за своє життя потрапляв на гачок злочинців.. Від переходу по невідомому посиланню, передачі власних даних до незахищеного, або невідомого джерела, відповідь на телефонний дзвінок із невідомого номера, тощо... Силове та інтелектуальне змагання між спеціалістами з безпеки та злочинцями подібна до гри у «кішки та миші». З

розвитком технологій все більше методів вторгнення, обману і подібного… І методи захисту оновлюються, ледве, не щодня.

Варто зауважити, що захист інформації – була і залишається з найактуальніших проблем сьогодення, нехтування нею – неприпустимо. Сьогодні, потужність сучасних комп'ютерів та хакерських систем дозволяє за декілька годин вивести із ладу, взламати, задудосити навіть найбільші системи світу, такі як Amazon, Facebook, тощо. Добре, що не усі компанії під таким прицілом, проте, хоча б первинний захист обов'язковий для кожного.

Провайдери, державні органи, як і основні дії для захисту себе, як користувача – не гарантують 100% безпечного користування мережею інтернет. Блокуючи незаконний чи шкідливий трафік, вони - нездатні перебрати кожен байт інформації на території держави - це неможливо … «Безпечний інтернет» - це запорука кожного, проте, люди не можуть встигнути розібратись в усьому, тому звернення до відповідних джерел та спеціалістів кіберзахисту це нормальна практика.

Спеціалісти кібербезпеки – займаються не лише блокуванням шкідливого трафіку чи захисними програмами, їх робота полягає у захисті самої інформації, техніки, що її обраховує чи цифрових носіїв.

На повсякчас, нестримність розвитку технікобудування все сильніше звужує специфіку захисту кожної системи... Для одного функціонування домашнього комп'ютера – звичайного брандмаузера операційної системи не вистачить. Антивіруси, додаткові брандмауери, VPN – піднімуть ваші шанси та якщо ви попадете під приціл хакінгу, їх може виявитись недостатньо. Порядок на власному ПК – це добре, та значно краще – порядок у власній голові.

Звичайно, що рішення у вигляді захищеної бази даних – не нове, адже, кожна велика система з петабайтами конфіденційної інформації створює власну систему для реалізації захисту. Існують програмні рішення, для яких

були навіть написані власні види мовних засобів чи створені унікальні програмно – апаратні пристрої шифрування.

Проте, як показує практика, чим довше використовується одна й та сама система захисту, протягом великого часу, її ефективність захисту починає знижуватись експотенціально, адже, все більше зловмисників нею зацікавляться, все більше буде знайдено вразливостей та й для звичайного брудфорсу буде більше часу... Тому, створення актуальних рішень, що відповідають нормам захисту – є не лише явищем підтримки конкурентно спроможного ринку, а й нагальність у плані безпеки.

Актуальність даного продукту полягає, як програмне забезпечення, що не лише має власні методи шифрування даних інформації та інтерфейс, а й надає змогу зберігати інформацію під багаторівневим рівнем захисту, що підвищує її потенціал, як конкуретного продукту серед інших.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Збір інформації та первинне дослідження вже існуючих систем для збереження інформації

Історичний екскурс – дає нам зрозуміти, що системи збереження інформації були створені ще за 70 років минулого сторіччя...

Першими вважаються покоління СКБД, що мають ієрархічну й мережеві структурні системи. Дані системи отримали широке поширення в 1970-х роках, а перші комерційні системи подільного типу стала система IMS компанії IBM.

1980-ті роки, дані системи були витіснені системами наступного покоління — повсюдно використовуваними і донині - реляційні СКБД. Дані системи використовуються, як непроцедурні мови управління даними (SQL) і передбачають значний ступінь незалежності даних. Реляційні системи вносять значні удосконалення для зручного керування даними, а графічний користувацький інтерфейс (GUI) та клієнт-серверні застосунки, що дозволяють створювати окремі системи для керування розподіленою базою даних – приносять значні вклади в оптимізацію та пришвидшення опрацювання даних, а паралельність пошуку даних та інтелектуальний аналіз даних дозволяють користувачам опрацьовувати ще більше об’єми інформації.

По при це, в кінці 1980-х років існуюча тоді реляційна модель вже не задовольняли розробників в силу низки обмежень. Відповідно до, зростаючої складності програм баз даних стали дві нові ланки напрямки розвитку СКБД і об'єктно-реляційні СКБД та : об'єктно-орієнтовані СКБД.

1991 подарував нам консорціум ODMG (**Object Data Management Group**), основними цілями якого, стали вироблення промислових стандартів об'єктно-орієнтованих баз даних. Поміж 1993 та 2001 років ODMG опублікували п'ять ревізій своїх специфікацій. Остання версії стандарту мали

індекс 3.0, після чого група розпустилася. До кінця дев’яностих, існувало близько десяти компаній, що виробляли комерційні продукти, що позиціонуються на ринку як ООСКБД. Найбільшими та найвідомішими системами даного виду - стали Objectivity, Versant виробництва однойменних компаній, а також СКБД Jasmine, випущена компанією CA.

Проте, відповідні переваги систем, що дозволяли ефективніше вирішувати певний ряд завдань, об'єктно-орієнтовані системи так і не змогли заполонити значущу частку ринку СКБД, залишившись «нішевим» продуктом.[1]

Беручи до уваги традиційні, реляційні СКБД були проведені та адаптовані із значними зусиллями для роботи з об'єднання об'єктно-орієнтованими і реляційними системами. Розробники постаралися розширити мову SQL, для того, щоб закласти їй змогу концепції об'єктно-орієнтованого підходу, водночас, зберігаючи переваги реляційної моделі (об'єктні розширення мови SQL були зафіксовані в стандарті SQL:1999). Основними принципи стали — розвиток можливостей СКБД, без недоліків та збереження попередніх підходів, та зі збереженням з системами попереднього покоління наступності та взаємозв’язків.

Поняття СКБД у третьому поколінні, якими, власне кажучи, і є об'єктно-реляційні СКБД, з'явились після багатьох публікацій групою відомих фахівців в області баз даних «Маніфесту систем баз даних третього покоління». Основними принципами СКБД третього покоління, стали:

1. Традиційні послуги з управління даними, СКБД третього покоління повинні забезпечувати підтримку розвиненіших структур об'єктів і правил. Розвинутішими структурами об'єктів характеризуватимуть засоби, необхідні для зберігання і маніпулювання нетрадиційними елементами даних (тексти, просторові дані, мультимедіа).
2. СКБД третього покоління повинна включити в собі СКБД другого покоління. Системні можливості другого покоління зробили вирішальний

внесок у двох областях — непроцедурності доступу за допомогою мови запитів SQL і незалежності даних. Ці досягнення обов'язково повинні враховуватись в системах третього покоління.

3. СКБД третіх поколінь, повинні включати в собі відкриті для інших підсистем можливості та запити опрацьовування інформації. Це включатиме оснащення різноманітними інструментаріями підтримки прийняття рішень, доступами зі багатьох мов програмування, інтерфейсами до існуючих популярних систем та бізнес-застосунків, можливістю запуску програм з бази даних на іншій машині і розподілені СКБД. Весь набір інструментарію і СКБД матиме ефективно – функціональних характер, на різноманітних апаратних платформах з різними операційними системами. Окрім, СКБД, що розраховує на широку сферу застосування, повинна бути оснащена мовою четвертого покоління (4GL).

Середина 1990 років були для СКБД не зовсім вдалими, були створені лише кілька дослідних прототипів СКБД, які поєднали найкраще серед реляційних і об'єктно-орієнтованих СКБД та не отримали значного впровадження на ринку =(.

Першим продуктом інформаційної комерції, були властиві об'єктно-реляційні риси, став Universal Server від компанії Informix (згодом була поглинена IBM). В даний час більшість цих ідей вже втілено в реальних комерційних рішеннях, в тому числі і в продуктах основних постачальників СКБД (Oracle Database і IBM DB2).

Розвинення індустрії інформаційно – обчислювальних систем та систем керування базами даних базувались на значних фундаментальних наукових дослідженнях. Взяти ті ж самі, дослідження та їх конкретною реалізацією в прикладних рішеннях минають роки, а іноді й десятиліття. Роботи в області управління даними проводились як університетські дослідницькі групи (MIT, Berkeley), так і центри над розробками основних

постачальників СКБД (Oracle, IBM, Microsoft, тощо). Інвестиції в управління даними — це довгострокове, і разом з тим, вигідне вкладення коштів. У той час, дослідники мали у своєму розпорядженні засоби, що дозволяють ефективно реалізувати складні запити, що маніпулюють мегабайтами різних даних. Проте, беручи до уваги сучасні можливості ми бачимо неймовірну різницю... Сучасні системи здатні за лічені хвилини обробляти терабайти та, навіть, петабайти об'ємів даних із найскладнішими видами з'єднань таблиць...

Основними тенденціями, що надали привід для проведення різних масштабних досліджень в області баз даних стали:

1. **Експонентний ріст**. Обсяг даних, у тому числі синтетичних, що генеруються автоматизованими системами, значно зріс. Збільшилося і число прикладних областей, в яких вимагається обробка великих обсягів даних. Такими областями тепер відносяться не тільки традиційні програми корпоративних систем, пошук у веб, але також і наукові дослідження, обробка природних мов, аналізування соціальних мереж тощо.
2. **Ускладнення структур даних**. Прості види даних у вигляді чисел і символьних рядків стали доповняться численною мультимедійною інформацією, просторовими, процедурними даними та великою кількістю інших складних форматів.
3. **Широке поширення дешевих високопродуктивних апаратних засобів**. Ми щороку бачимо все потужніші системи, за «відносно», невеликі можливості. Зараз, комп'ютер звичайного школяра, за добу може обробити більше даних, аніж найкращі комп'ютери, що використовувались для обрахунків для висадки на місяць.
4. **Активний розвиток засобів комунікації.** Стає єдиним інформаційне середовище, що пронизує весь світ і об'єднує величезне число користувачів та електронних пристроїв. Зараз, майже, на будь-яку

помилку в системі можна знайти відповідь витративши 5 хвилин. Адже, свідомий програміст, викладе питання (а можливо і рішення), для окремої проблеми.

5. **Поява нових важливих областей застосування**. Системи баз даних у першу чергу, пов'язані з інтелектуальним аналізом даних, сховищами даних, а останнім часом — з паралельними обчисленнями і хмарними технологіями. Це дозволило збільшити їх актуальність, а в свою чергу кількість спеціалістів, в яких постійно зацікавлені роботодавці.[2]

Ще однією особливістю сучасних баз даних – це зручний користувацький інтерфейс. MS SQL, MySQL, Heidi, тощо, отримали графічну підтримку, що підняло ефективність аналітики та керування базами даних. Ще одним прикладом сучасних баз даних, є бази зі підтримкою обраховування статистики. Подібні рішення – не для програмістів, а для аналітиків. Деякі програмні продукти, такі як Access, studio R та інші, подібного класу, навіть, мають власну мову програмування для реалізації запитів із БД. До прикладу, ось короткий опис програми Access.

В Access все це один зібраний файл, котрий містить дані у вигляді однієї або кількох таблиць. БД, створена ж таким чином, що на локальному комп'ютері, збережені разом з елементами додатка й програмні коди в одному mdb-файлі, що спрощує як створення, так і використання інших додатків БД. Інструментальні засоби СКБД MS Access включають:

- хелпери, що дозволяють створювати об'єкти БД і об'єкти додатків із діалогового режиму, а також виконувати різноманітні функції з перетворення та реорганізації БД;
- засоби графічного конструювання, що дозволяють створювати об'єкти БД і об'єкти додатків, не вносячи програмного коду;
- засоби програмування, до котрих входять мова структурованих запитів SQL, мова макрокоманд і об'єктно-орієнтована мова програмування Visual Basic for Application (VBA).

Також, варто виділити основну особливість, що дозволяє використовувати подібні програми для будь-яких користувачів, власне, обробляти всю інформацію не написавши запиту:

- засоби для створення таблиць і схем даних;
- засоби конструювання запитів на зміну даних бази та отримання даних;
- засоби створення діалогових вікон, що дозволяють обробляти інформацію, вводити та виводити її у окремому формулярному середовищі;
- засоби створення звітів, призначених для перегляду та виведення на друк даних з бази і результатів їх обробки;
- засоби сторінок доступу до даних та їх створення, що забезпечують роботу з БД у середовищах Інтернет і Інтранет;
- засоби конструювання інтерфейсу користувача – меню, панелей керування додатком, що дозволяють об'єднати операції з роботи з БД у єдиний технологічний процес. [3]

Вельми зручним є те, що БД найбільш розповсюджені для обробки даних інтернету, проте, найбільш складними та структурованими є бази мереж Інтранет окремих великих організацій.

Інтране́т — внутрішня – корпоративна мережа, що створена на базі інтернету та відокремлена від нього. Зазвичай, використовує окремі системи комунікації із внутрішніх ресурсів компанії та має виключне спеціалізоване направлення.

Комп'ютерна мережа, що використовує технології інтернету, але водночас є приватною корпоративною мережею. Подібні мережі підтримують сервіси інтернету, наприклад, такі, як FTP – сервери, електронна пошта, вебсайти тощо, але в межах корпорації. Інтранет-мережа, підключається до зовнішніх мереж, у тому числі і до інтернету, як правило, через засоби захисту від несанкціонованого доступу. Інтранет може бути ізольований від зовнішніх

користувачів або функціонувати як автономна мережа, що не має доступу ззовні.

В Intranet використовуються стандартні для Internet служби, вони абсолютно ті самі, тому є відповідна підтримка CGI, SMTP, FTP, HTML, HTTP, HTTPS, TCP/IP, системи доменних імен і Web-браузери, для відображення інформації з розміщених по підприємству Web-серверів.[4]

Повертаючись до питання особливостей висококваліфікаційних БД, варто зауважити, що за необхідності створення нової бази даних користувач може використати численні шаблони різних об'єктів бази, або швидко їх редагувати за допомогою інтерфейсу, що входять до складу Access, або створювати свої об'єкти для конкретних задач.

База даних у середовищі MS Access дозволяє:

- пошук мети створення БД, яка визначатиме перелік таблиць;
- визначення структури таблиць (полів та їх типів);
- реалізація ключів таблиць та визначення зв'язків між таблицями;
- завантаження даних до таблиці;
- створення інших об'єктів бази даних – запитів, форм, звітів, макросів та модулів.

У MS Access, як і в інших СКБД, використовується спеціалізоване лінгвістичне забезпечення. Зазвичай, це спеціалізована мова для взаємодії з структурованою інформацією БД – SQL, чи System R. Сама назва, частково, відображає її суть як зручного інструмента для створення та реалізації запитів до реляційних баз даних. Дана мова SQL являє собою послідовну комбінацію реляційного обчислення кортежів і реляційної алгебри. Водночас, можливості мови SQL для окремих операцій ширші, оскільки у реляційній алгебрі ми значно сильніше залежимо від потужностей самої системи. Базовий набір мовних операторів, уміщує оператори визначення схеми БД, маніпулювання та вибірки даних, авторизації доступу до даних, вбудування мови SQL в інші додаткові мови програмування. Більшість мов, для

виокремленої та зручної (збереження людської логіки) методології, мають схожі методи обраховування. До прикладу, мова допускає три типи синтаксичних конструкцій, які починаються з ключового слова SELECT (вибрати):

- оператора вибірки (select statement) – список полів, що вибираються;
- специфікації курсора (cursor specification) для тимчасового зберігання результатів запиту у курсорі;
- підзапит (subquery) на виконання певних операцій з даними полів.

Або методи математичних обчислень SUM (просумувати) чи COUNT (підрахувати).

Дані запити, що вираховуються, є максимально оптимізованими, що дозволяють обраховувати петабайти даних за короткі проміжки часу.

1.2 Пошук існуючих методів розподілу рівнів доступу до інформації згідно існуючих стандартів.

Беручись до даного пункту, варто переглянути власне можливий функіонал системи, можливості подальших змін у кількості та статусах користувачів, тощо.

Для подібного розподілу варто організувати графічне відображення розподілу рівнів доступу до інформації. Подібне рішення дозволить проаналізувати необхідний функціонал для різних користувачів та збереження інформаційних даних

Зазвичай, інформаційне розгалуження за грифами доступу виглядає так:

Таблиця 1.1.1 – Побудова таблиці рівня конфіденційності

№ п\п	Інформація	Гриф доступу
1	**2**	**3**
1.	Відомості про ідентифікатори та паролі системного адміністратора та інших осіб, що мають доступ до управління	**К**
1	Відомості про ідентифікатори та паролі користувачів робочих станцій	**Т**
2	Відомості про клієнтів	**К**
3	Відомості стосовно діяльності клієнтів	**К**
4	Відомості про постачальників	**К**
5	Відомості про організацію та технічні засоби реалізації основних задач компанії	**К**
6	Методи шифрування персональних даних, кешів та інших інформаційних одиниць збереження інформації, що використовуються внутрішніми сервісами	**ЦТ**

	Продовження таблиці 1.1.1	
1	2	**3**
№ п\п	**Інформація**	**Гриф доступу**
7	Інформація, що стосується особи працівника	**К**
8	Відомості про зміст бази даних захисту (облікові записи, паролі користувачів зі всіма атрибутами захисту)	**Т**
9	Технічні заходи щодо захисту конфіденційної інформації	**ЦТ**
10	Способи утворення сторонніх запитів, що регламентовані системою	**ЦТ**
11	Токени для інформаційного обміну між системою та ботом Телеграм	**ЦТ**
12	Методи опрацювання токенів всередині системи, що були створені сторонніми засобами, такими як Sanctum, Fortify	**Т**
13	Кількість та методи функціонування запитів до контролерів, їх необхідні змінні	**ЦТ**

	Кінець таблиці 1.1.1	
1	2	**3**
№ п\п	**Інформація**	**Гриф доступу**
14	Доповідні записки, довідки, інформаційні листи, методичні рекомендації з питань збереження конфіденційної інформації	**К**
15	Облікова картка користувача	**К**
16	Адреса користувачів	**К**
17	Ознайомлююча інформація про співробітників та компанію	**В**

1.3 Пошук та аналіз інформаційних загроз під час опрацьовування БД

Беручи до уваги, той факт, що перелік інформаційних загроз може бути невичерпним, варто звернутися до методу розподілу за критеріями подібних ознак. Тому, більшість загроз можна розподілити на:

- За аспектом інформаційної безпеки, на який спрямовані загрози:

Загрози конфіденційності (неправомірний доступ до інформації). Загроза порушення конфіденційності полягає в тому, що інформація стає відомою тому, хто не володіє повноваженнями доступу до неї. Вона має місце, коли отримано доступ до деякої інформації обмеженого доступу, що зберігається в комп'ютерній системі або передається від однієї системи до іншої. У зв'язку з загрозою порушення конфіденційності, використовується термін «витік». Подібні загрози можуть виникати внаслідок «людського фактора» (наприклад, випадкове делегування тому або іншому користувачеві привілеїв іншого користувача), збоїв роботи програмних та апаратних

засобів. До інформації обмеженого доступу належить державна таємниця (комерційна таємниця, персональні дані,

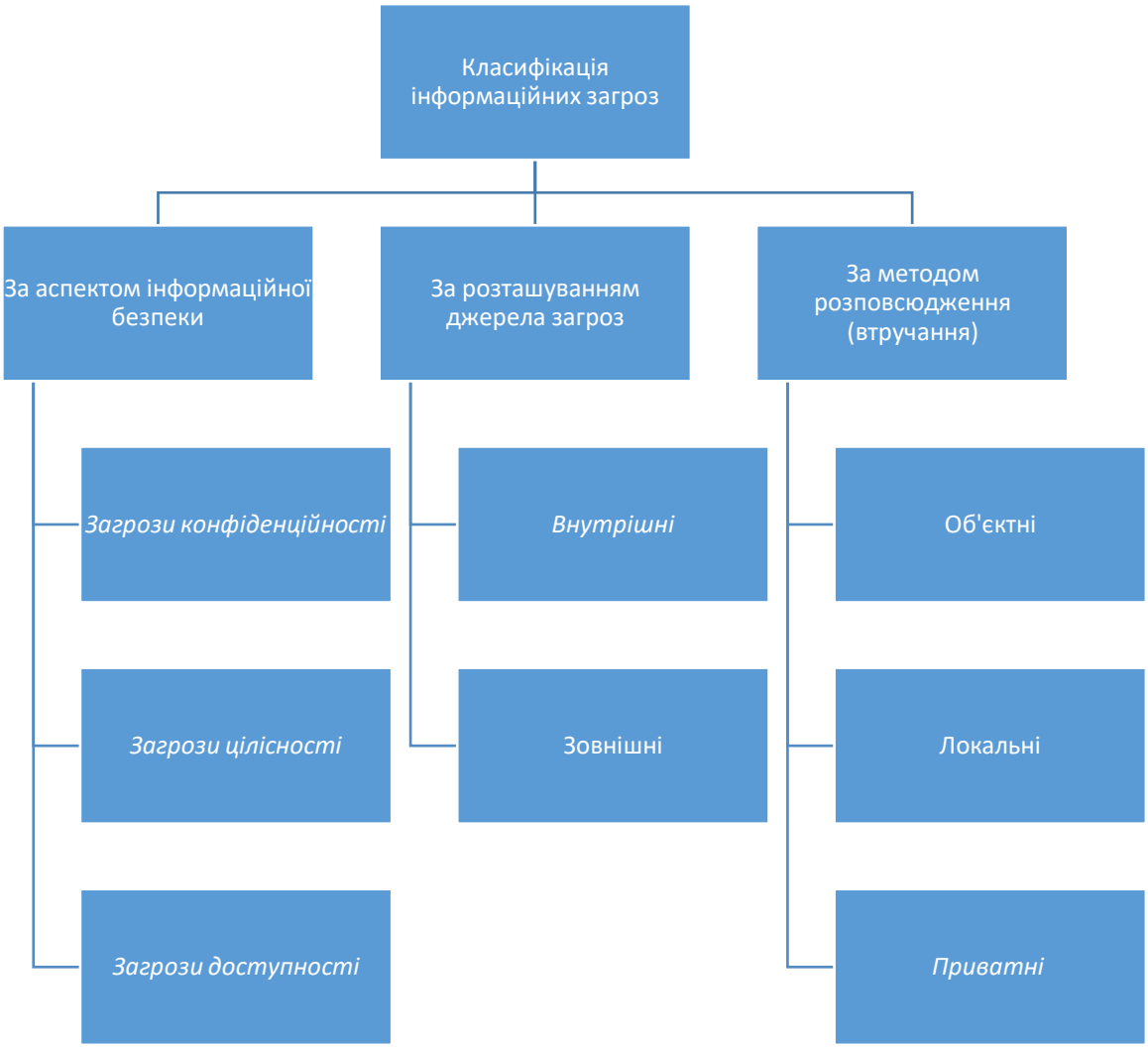


Рисунок 1.2.1 – Схема класифікацій загроз на інформаційних ресурсах

- Професійні види таємниці: лікарська, адвокатська, банківська, службова, нотаріальна таємниця страхування, слідства й судочинства, листування, телефонних переговорів, поштових відправлень, телеграфних або інших повідомлень (таємниця), відомості про сутність винаходу, корисної моделі або промислового зразка до офіційної публікації (ноу-хау) та ін.).

- *Загрози цілісності* (неправомірна зміна даних). Загрози порушення цілісності — це загрози, пов'язані з імовірністю модифікації тієї чи іншої інформації, що зберігається в інформаційній системі. Порушення цілісності може бути викликано різними чинниками — від умисних дій персоналу до виходу з ладу обладнання.
- *Загрози доступності* (здійснення дій, які унеможливлюють чи ускладнюють доступ до ресурсів інформаційної системи). Порушення доступності являє собою створення таких умов, при яких доступ до послуги або інформації або заблокований, або можливий за час, який не забезпечить виконання тих чи інших бізнес-цілей.

За розташуванням джерела загроз:

- *Внутрішні* (джерела загроз розташовуються всередині системи);
- *Зовнішні* (джерела загроз знаходяться поза системою).

За методом втручання:

- *Об'єктні* (заподіяння шкоди об'єкту в цілому, як держава, установа, тощо);
- *Локальні* (заподіяння шкоди окремими частинами об'єкта безпеки);
- *Приватні* (заподіяння шкоди окремим властивостям елементів об'єкта безпеки).[5]

1.4 Виконання функціоналу програми

Беручи до уваги гарний принцип у вигляді: “Чим менше знають користувачі, як воно працює, тим для них й краще”. Для цього, велика кількість систем використовують AJAX – методи.

Ajax – це окрема мова, що була створена для побудови користувацьких інтерфейсів та вебзастосунків, у яких вебсторінка, не перезавантажується, а у фоновому режимі надсилає запити на сервер і сама звідти отримує потрібні дані для генерації сторінки користувачу. AJAX — один з компонентів концепції DHTML.

Про AJAX заговорили після появи в лютому 2005 року статті Джесі Джеймса Гарретта (Jesse James Garrett) «Новий підхід до вебзастосунків». AJAX — не самостійна технологія. Це ідея, що дозволяє реалізувати нові підходи до системи інтерфейс – сервер.

AJAX — це не самостійна технологія, а скоріше, окреме рішення, що дозволяє використання декількох суміжних технологій. Розробка, що використовує AJAX-підхід, для користувачів інтерфейсів, комбінує кілька основних методів і прийомів:

- Використання DHTML для динамічної зміни змісту сторінки.
- Використання XMLHttpRequest для звернення до сервера одразу, не перезавантажуючи всю сторінку повністю
- альтернативний метод — динамічне підвантаження коду JavaScript в тег <SCRIPT>
- динамічне створення дочірніх фреймів та інших систем попереднього завантаження.

Використання даного рішення дозволяє створювати набагато зручніші вебінтерфейси користувача на тих сторінках сайтів, де є необхідність у активній взаємодії з користувачем. AJAX — являє собою асинхронне рішення, тому користувач може переглядати далі контент сайту, поки сервер все ще обробляє запит. Браузер - не перезавантажує web-сторінку, а самі ж дані посилаються на сервер без візуального підтвердження (крім випадків, коли ми самі захочемо). Використання AJAX стало популярним після того, як компанія Google почала активно використовувати його при створенні своїх сайтів, таких як Gmail, Google Maps і Google Suggest. Створення цих сайтів підтвердило ефективність використання даного підходу. [6]

Рис. 1.3.1 – Багатоетапний процес обрахунку запитів системою від користувача із використанням Ajax методів

1.5 Постановка задачі проектування

Необхідно розробити програмний продукт, що буде відповідати таким пунктам, як:

- Оптимізоване середовище опрацьовування даних;
- Зручний інтерфейс користувача;
- Шифрування конфіденцій даних користувачів та спеціалізовані дані роботи системи;
- Інтегрування до системи можливості API – запитів
- Токенізація та можливість збереження куккі
- Відносна підтримка для збереження різних типів даних
- Створення інтерфейсу, процеси якого будуть невідомі для самого користувача
- Розмежовування доступу до інформації

- Децентралізація даних
- Багатоетапний план для реєстрації користувачів, або створення нових
- Метод верифікації нових користувачів за допомогою сторонніх систем
- Створення методів валідації системи для відсторонення можливих інкапсуляцій, тощо

2 ПРЕДПРОЕКТНИЙ АНАЛІЗ НЕОБХІДНИХ СКЛАДОВИХ ДЛЯ ПРОЕКТУ ТА ЇХ ОБЛАШТУВАННЯ

2.1 Аналіз необхідних складових виходячи зі необхідного функціоналу програми

Для даної системи буде введено декілька типів розповсюдження даних

1) це локальне збереження те передача даних

2) система передачі-контролю-отримання зовнішніх посилань до системи

Оскільки, мова PHP – це гіпертекстовий постпроцесор, тобто, мова без якихось початкових налаштувань, почнемо зі конфігурації проекту та завантаження необхідних допоміжних бібліотек.

2.1.1 Встановлення Composer

Composer — це спеціалізований менеджер пакетів прикладного рівня для мови програмування PHP . Він забезпечує стандартизацію форматування для управління залежностями у програмному забезпеченні, а також необхідними бібліотеками. Був розроблений Нілом Адерманом і Хорді Боггіано, які і досі супроводжують проект. Розпочалась розробка в квітні 2011 року і вперше випустили його 1 березня 2012 року. Composer працює з командного рядка і встановлює залежності (наприклад, бібліотек) для застосунку. Також, надає змогу користувачам встановлювати PHP пакети, доступні на «Packagist», який є джерелом його даних для плагінів, тощо. Зазвичай, він займається реалізацією автозавантажування класів, для встановлених бібліотек і це полегшує використання коду від сторонніх розробників.

Composer використовується як складова частина декількох популярних PHP проектів з відкритим вихідним кодом, наприклад: Laravel, Symfony. У випадку даної системи ми будемо налаштовувати його на модель Laravel [7]

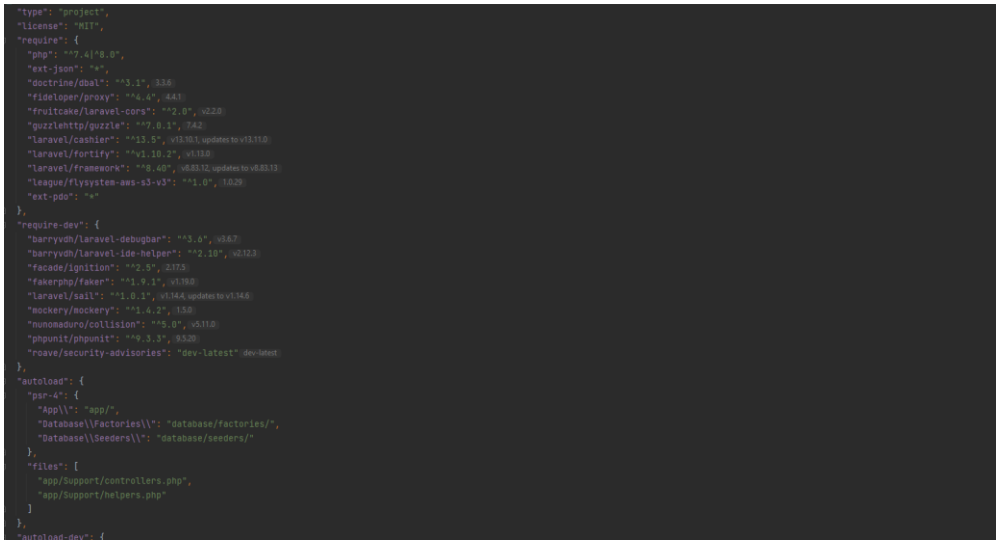


Рис. 2.1.1 – Зовнішній вигляд конфігурації composer.json

2.1.2 Встановлення PHP Artisan/Laravel

Laravel та **Artisan** — це безкоштовні фрейморки, що мають відкритий PHP код, їх задача полягає у менеджменті системного устаткування даних, що дозволяє створювати та викликати консольні команди, контролери, репозиторії та міграції. Значно полегшує та прискорює розробку , а також його тестування. Усі бібліотеки – безкоштовні, мають подібну структуру MVC. Серед особливостей Laravel можна назвати: модульну систему упакування з виділеним менеджером залежностей Composer, різні способи для доступу до реляційних баз даних, утиліти, які допомагають в розгортанні додатків і технічного обслуговування.

2.1.3 Встановлення окремих додаткових бібліотек

До даного проекту були додані такі утиліти: Telegram, Fortify, API та Hash.
Усі бібліотеки – налаштовані та запрограмовані на відповідні завдання.
Telegram – дозволить відправляти до окремо створеного бота повідомлення для верифікації користувачів
API – дозволить використовувати програму без локального ресурсу та запровадити злиття із іншими програмами використовуючи наш функціонал.

Hash – в даній утиліті використовується хешування, для створення засекречених паролів користувачів, окремих генерованих токенів.

2.2 Остаточне налаштування та підключення робочого середовища для проекту

Оскільки, не всі бібліотеки надають змогу, одразу, використовувати їх у нашим планах та цілях, доводиться вносити відповідні корективи.

2.2.1 Створення Telegram Bot – для сповіщень.

Дана система генерує хеш – код, коли наші користувачі будуть створювати нові облікові записи, змінювати паролі, запрошувати нових користувачів, тощо. Найкращим рішенням для подібного – надсилання повідомлень за допомогою різних захищених сервісів, що надають такі можливості. Проте, деякі з них мають не зовсім «дружелюбну» політику на рахунок цього. Тому, вибір впав саме на Телеграм. Він зручний, захищений та відправляє повідомлення за лічені секунди (при доброму інтернет зв’язку). Вже встановлена бібліотека, що інтегрує опрацьовування телеграм запитів у цьому допоможе.

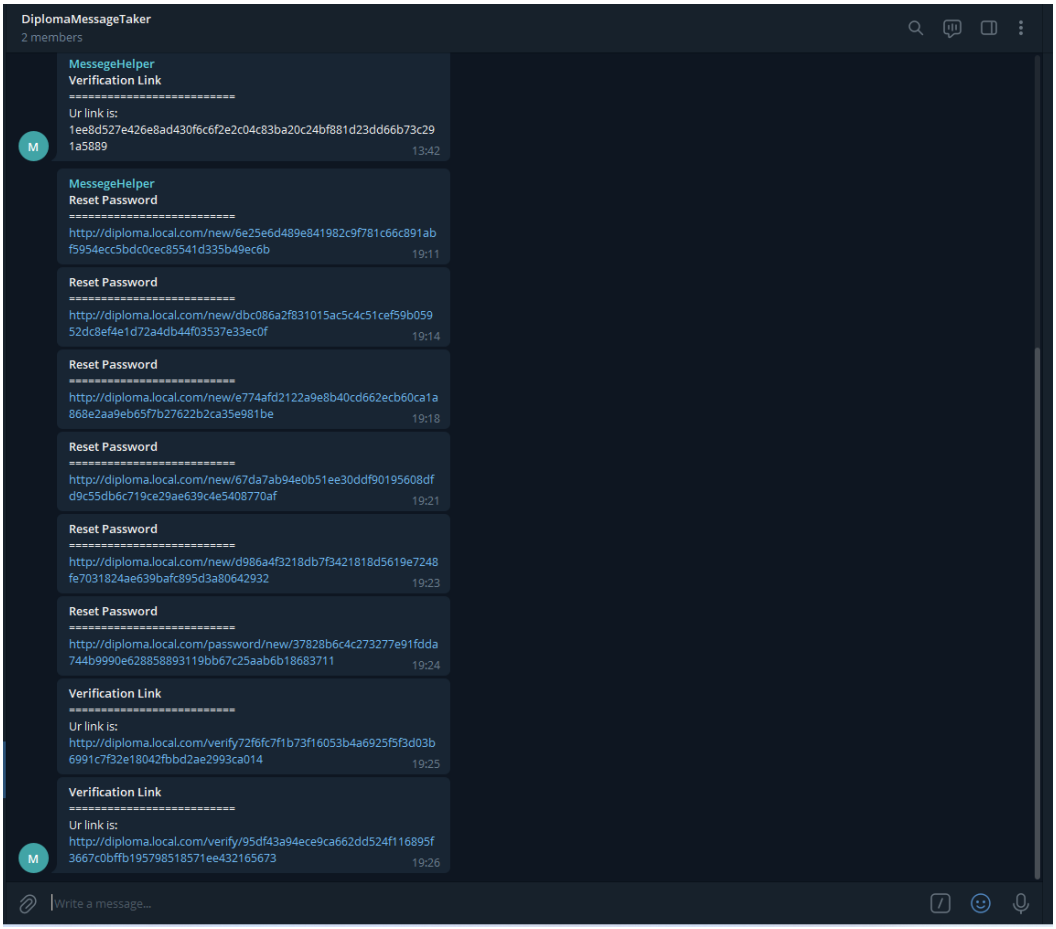


Рис. 2.2.1 – Зовнішній вигляд телеграм – обліку реєстрацій та змін паролів

Беручи до уваги те, що система може брати різні напрямки, а для зручності необхідно приєднати якомога більше систем сповіщення, я створив окрему структуру повідомлень, для полегшення подальших інтеграцій.

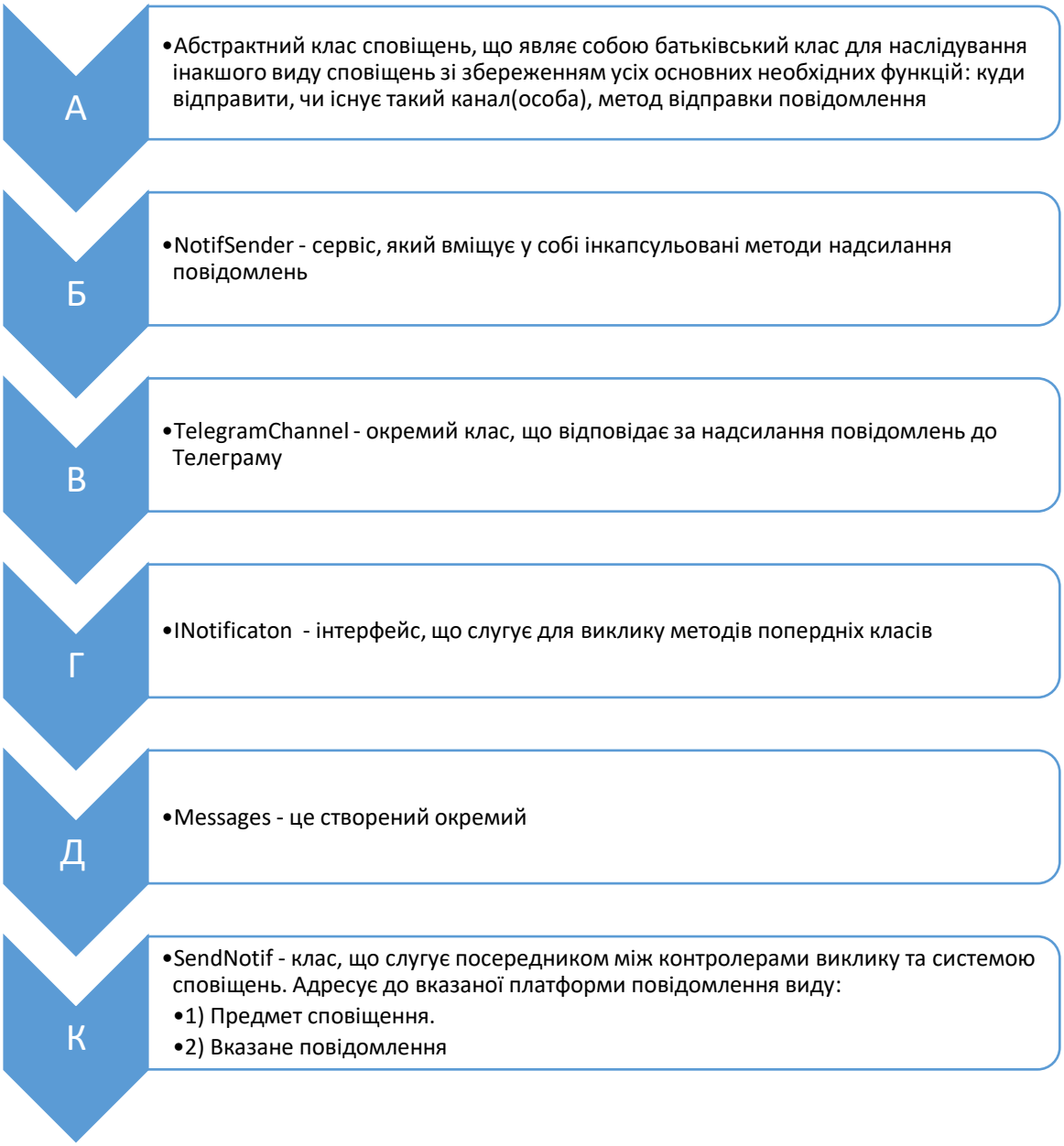


Рис. 2.2.2 – Схематичне зображення системи сповіщень програми

Телеграм бот – має власну специфіку опрацьовування запитів та повертає відповіді у форматі json, котрі зручно та швидко опрацьовувати.

Сама система Телеграм – черезвичайно універсальна, що у майбутньому надасть нам змогу збільшити функціонал програми, відповідно до потрібних нам умов.

Подібна конструкція самої системи може видозмінитись, до прикладу, сповіщення будут відправлятись напряму, до приватних повідомлень користувачів, або створення окремого середовища для збереження даних, що буде використовувати сервіси Телеграму.

Окрім цього, телеграм – не єдина система для сповіщень, до найбільш поширених відноситься система електронної пошти (Google, MailChimp, тощо), або SMS сповіщення. Дані системи не розглядались для написання даної роботи, через їх високу ціну для реалізації та часозатратність, а також через офіційність, тобто використання даних сервісів змушує підпорядковуватись їх правилам організації сповіщень, що додає проблем зі запуском функціоналу.

Єдиним, можливим, недоліком даної системи є те, що необхідно зберігати окремо виданий токен для користування даним ботом від самого сервісу Телеграм, що не є дуже зручно да безпечно.

2.2.2 Переробка системи двухетапної аутентифікації Fortify

Fortify – система обліку та аутинтефікації користувачів, що рекомендується самим сервісом Laravel. Її особливість полягає не лише в повноцінній системі аутентифікації, що автоматично інтегрується до проекту, а й шифруванням та моніторингом входжень до системи.

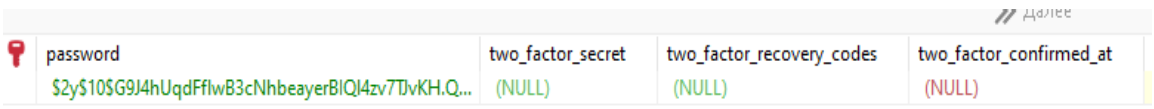


Рис. 2.2.3 – Зовнішній вигляд створених полів бібліотекою Fortify до моделі User

Оскільки, система Artisan, здатна власноруч генерувати міграції за замовчуванням (дозволяючи їх редагувати), до основної моделі користувача, додаються декілька полів для двухфакторного входу в обліковий запис.

Недоліком цієї системи, є її недостатня підготовка в системі користування, оскільки підтвердження того, що користувач – дійсно, зберіг згенерований шифр – немає. Тому, внесені деякі зміни до основних класів користування було отримано рішення даного виду:

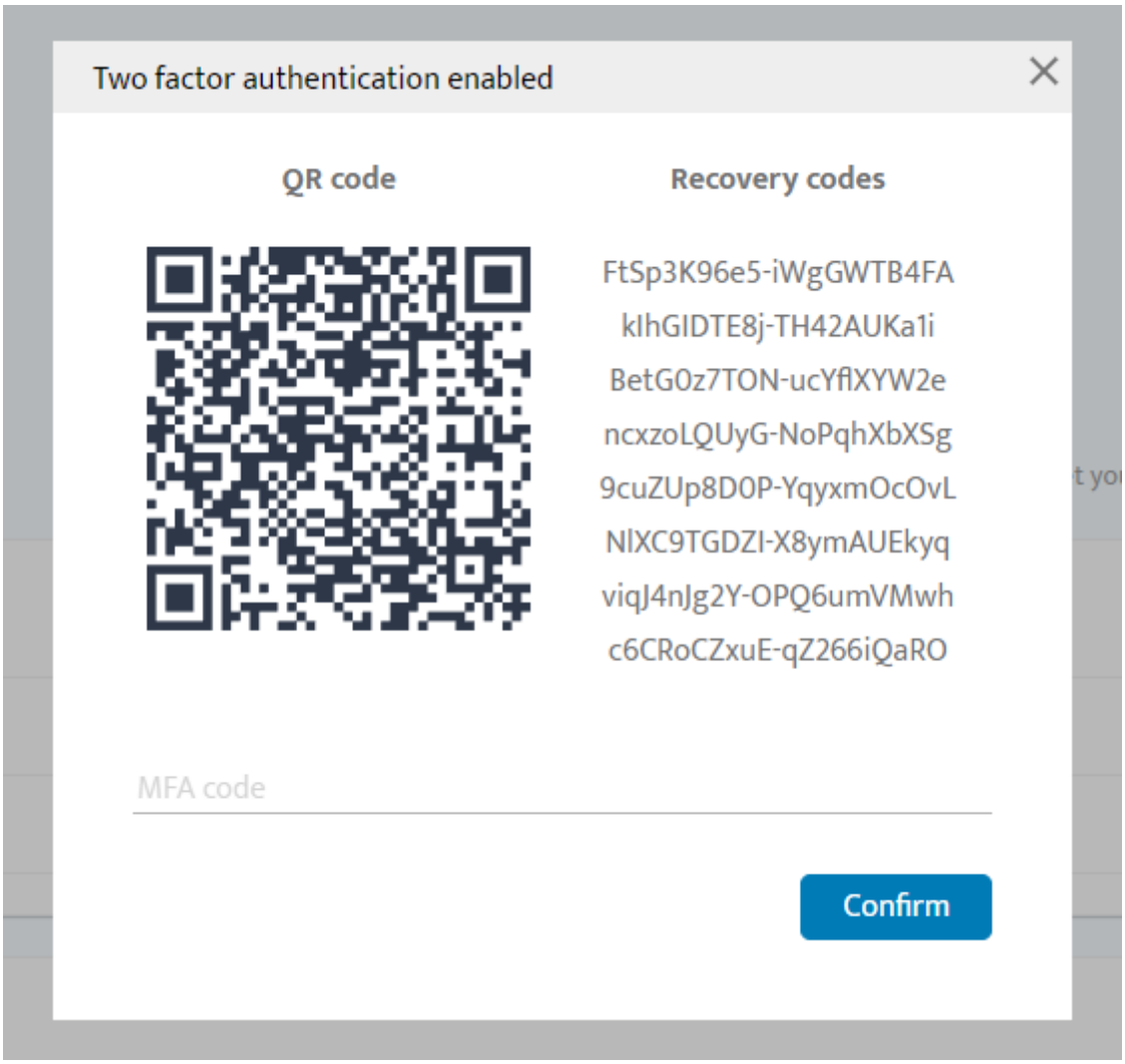


Рис. 2.2.4 – Зовнішній вигляд створеного підтвердження двухфакторної аутентифікації

Дана система використовує Hash подібної до системи SHA-1, створюючи 160 – бітний алгоритм генерації випадкового числа. Після сканування QR – коду, ми отримаємо 6 – значне випадкове число, кожні 30 секунд.

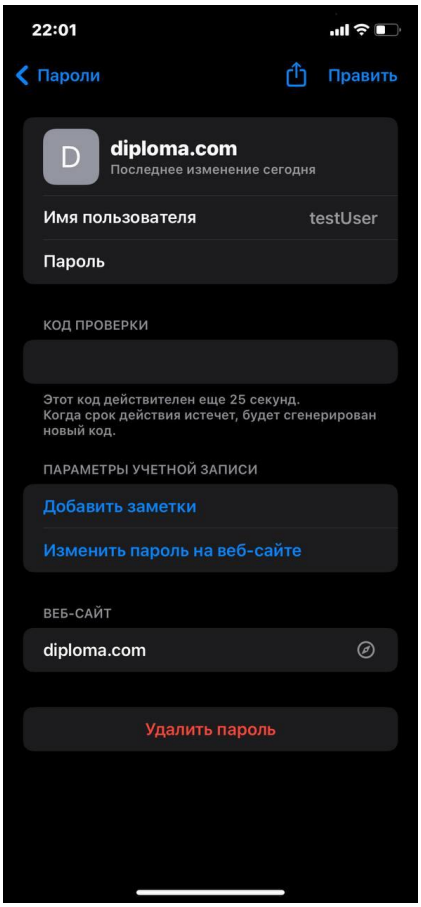


Рис. 2.2.5 – Зовнішній вигляд інтерфейсу пристрою на базі IOS зі згенерованими числами (неможливо показати код на зображенні, через політику безпеки системи)

Подібна організація 2FA – підтвердження далеко не нова, проте, не існує кращої альтернативи, що для самих користувачів, так і розробників програмного забезпечення.

Генерація шестизначного унікального коду відбувається на системі користувача (телефоні, комп’ютері, тощо) та автоматично відображається у налаштуваннях, або додатках користувача.

Нескладна операція відбувається – не постійно, проте, ведеться збереження даних алгоритму та його циклічного періоду, що трохи забирає можливості оперативної та постійної пам’яті.

Одночасно із тим, у випадку «випадкового» видалення алгоритму, є можливість відновлення доступу за допомогою окремих ключів, кожен із них – одноразовий, при використанні усіх акаунт блокується.

2.2.3 Оформлення додаткових сервісів сторонніх зовнішніх запитів до системи

Беручи до уваги тему даної роботи, звернемось до питання децентралізації. Основна думка полягає в тому, щоб надати можливість користувачам використовувати базу даних за рамками програмного продукту, для цього була реалізована система API – запитів сервісом Laravel Sanctum, в подальшому вона буде перевірена стороннім програмним середовищем PostMan.

На основі теоретичного аналізу, проведеного в попередньому розділі, ми визначились з порядком дій, необхідних для успішного виконання індивідуального завдання проектно-технологічної практики.

Окрім логічного порядку дій, також слід визначитись з програмними та програмно-апаратними інструментами, що дозволять нам виконати необхідні операції.

Говорячи про практичну роботу над даним індивідуальним завданням, в першу чергу, слід обрати мову програмування. Мій вибір випав на PHP.

Даний вибір – не спроста, оскільки дана платформа має зручний та швидкий для вивчення метод перевірки API – запитів, що у майбутньому полегшить нам роботу із перевіркою ПЗ.

Найкращий та при цьому доволі простий варіант реалізації буде аналіз уже існуючих рішень, тобто, відкриті у використанні методи збереження інформації, що рекомендуються самою платформою.

Провівши аналіз різних методів реалізації захисту, я прийшов до висновку, що самостійно створене рішення – не може бути виконане у відкладений час. Оскільки, нам необхідний продукт, що гарантовано надасть можливості у безпечному його користуванні. Також, відкриті, «ноу-нейм», бібліотеки, що є у відкритому доступі – не підходять, оскільки, вони не тестовані повноцінно та мають високу вірогідність колізійних виключень коду.

Знайдені рішення та теоретичні данні про них:

1. Postman - це платформа для створення і використання API. Postman спрощує кожен етап життєвого циклу API і оптимізує спільну роботу, щоб ви могли швидше створювати кращі API. Платформа Postman включає повний набір інструментів, які допомагають прискорити життєвий цикл API, - від розробки, тестування, документування і мокинга до обміну і відкриття ваших API. Платформа Postman надає вам розширену аналітичну інформацію про усі ваші операції з API, використовуючи сповіщення і попередження про безпеку, пошук, звітність і багато що інше. [8]

2. Laravel Sanctum - існує для вирішення двох окремих проблем. По-перше, Sanctum - це простий пакет, який використовує для видачі API- токени вашим користувачам без ускладнення OAuth. Ця функція натхненна GitHub і іншими застосуваннями, які видають "персональні маркери доступу". Наприклад, уявіть, що в "налаштуваннях акаунта" вашого застосування є екран, де користувач може згенерувати API-токен для свого акаунта. Ви можете використати Sanctum для генерації і управління цими маркерами. Ці токени зазвичай мають дуже тривалий термін дії (роки), але можуть бути вручну відкликані користувачем у будь-кого.

Laravel Sanctum пропонує цю можливість, зберігаючи призначені для користувача API- токени в одній таблиці бази даних і перевіряючи

достовірність HTTP- запитів, що входять, за допомогою заголовка Authorization, який повинен містити дійсний API- токен.

По-друге, Sanctum існує для того, щоб запропонувати простий спосіб аутентифікації односторінкових застосувань (SPA), яким необхідно взаємодіяти з API, працюючими на Laravel. Ці SPA можуть існувати в тому ж репозиторії, що і ваш додаток Laravel, або можуть бути абсолютно окремим репозиторієм, наприклад SPA, створеним за допомогою Vue CLI або додатка Next.js.

Для цієї функції Sanctum не використовує які-небудь токени. Замість цього Sanctum використовує вбудовані в Laravel сервіси аутентифікації сесії на основі cookie. Зазвичай Sanctum використовує для цього захист веб-аутентифікації Laravel. Це забезпечує захист від CSRF, сеансову аутентифікацію, а також захищає від витоку аутентифікаційних даних через XSS. Sanctum намагатиметься аутентифікуватися за допомогою cookies тільки у тому випадку, якщо запит, що входить, виходить від вашого власного фронтенда SPA. Коли Sanctum розглядає HTTP- запит, що входить, він спочатку перевіряє наявність аутентифікаційного cookie, і якщо такого немає, Sanctum перевіряє заголовок Authorization на наявність дійсного API- токена. [9]

3. Hash 256 – багатоетапний вид шифрування даних за методом Меркла – Демгарда, оскільки, ми будемо зберігати багато даних користувачів, необхідно створити унікальні коди для кожної з дій та шифрування конференційної інформації (втручання до бази даних – не принесе зловмиснику інформації, її ніхто не зможе зрозуміти, окрім, самої програми.) [10]

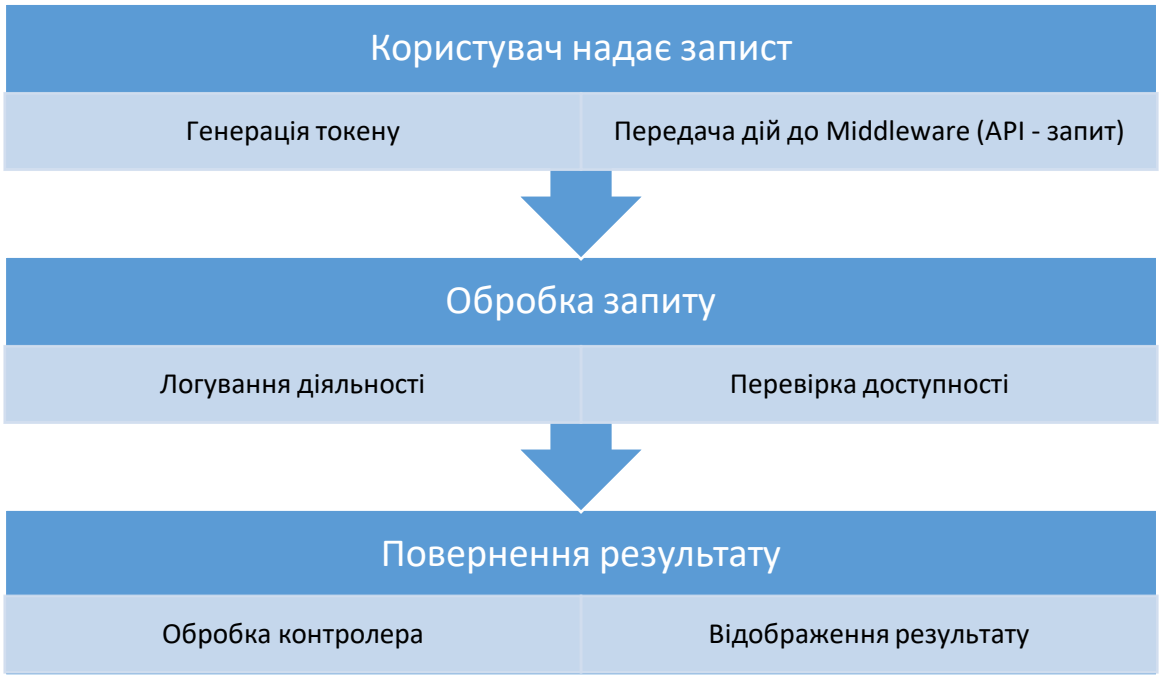


Рисунок 2.2.6 – Схема почергової обробки запиту

Дана система дозволяє нам забезпечити захищеність сесії користувача та його діяльність. Децентралізація можливостей для користувача дозволятиме користуватись сервісом бази даних, навіть, при відключення користувацького інтерфейсу. Проте, необхідним буде інше програмне забезпечення, що дозволятиме обробляти запити та використовувати створене API забезпечення.

Система API запитів надзвичайно поширена, не лише у програмних сервісах, а й на апаратних рівнях. Адже, подібний метод можливо порівняти із системою магнітних ліній кредитних карток, електронних паспортів та навіть радіостанцій із виокремленим діапазоном радіохвиль.

Звичайно, перехопити подібні запити – нескладно, оскільки вони подорожують, зазвичай, мережею, що використовується багатьма особами чи сервісами. Тому, для них організовується система захищеної передачі даних, від згенерованих токенів, до окремих криптографічних систем шифрування.

Як показує досвід, API запит – надзвичайно корисна річ та її неможливо відтворити у бездоганному вигляді... Проблеми виникають у необхідній

інформації до користувачів, навантаженні на власний сервер запитами інших пристроїв та передачею даних, створенні універсального інструмента, що буде підходити якомога більшій кількості осіб, та інше. Подібний список величезний, бо використання таких систем дозволяє інтегрувати сервіс у інші системи та веденню систематизації.

3 СИНТЕЗ КОМПЛЕКСНОЇ СИСТЕМИ ЗАХИСТУ ІНФОРМАЦІЇ

3.1 Створення захищеної бази даних

Основною метою проектувань баз даних - є скорочення надмірності збережених даних. Грамотне планування баз даних забезпечує оптимальне використання оперативної і дискової пам'яті, дає доступ до зручних механізмів змін даних і забезпечує високу їхню цілісність. У невдалому плануванні може мати місце дублювання даних. Адміністрування та системи моніторингу не повинні викликати проблем, із аналізом та їх використанням, надмірне дублювання може викликати подальші помилки зі виводом даних та їх систематизацією, варто пам’ятати, що необхідно стежити за всіма копіями даних, тобто при зміні однієї копії необхідно виправити й інші копії.

Нормалізація – це виокремлений процес зведення структур даних у стан, що здатен надати найкращі умови вибірки, включення, зміни і видалення даних. Подібне можливо досягти за допомогою розбивання великих обсягів даних на менші об’єкти збереження інформації. Кінцевою метою нормалізації є отримання доцільних проектів осередків інформації, що швидко та коректно опрацьовуватимуть прямі запити та не викликатимуть проблем із їх побудовою. Такого проекту бази даних, у якому кожен факт здатен з’явитись лише в одному місці, мається на увазі, виключена надмірність інформації. Подібні трактації є нормованими не стільки з метою економії пам'яті, скільки для нівелювання можливої суперечливості збережених даних. Теоретично, нормалізації відношень описує формальний апарат обмежень на формування відношень (таблиць), що надає змогу усувати дублювання, забезпечити несуперечність даних і зменшити трудозатрати на ведення бази даних. [11]

Перед початком опису самих методів нормалізації, що використовувались, потрібно сказати про універсальне відношення. Таблиці,

в яку включені всі атрибути, які представляють інтерес для даної організації називається універсальним відношенням. При використанні подібних (універсальних) відношень інформаційний ресурс буде складатися з єдиної таблиці, у якій стане розташовуватися вся інформація. Кількість атрибутів у такому відношенні може бути дуже великим.

Надмірність – це інформація перебільшення, накопичення певних полів в багатьох стовпцях багаторазово повторюється. Звичайно, подібні колонки, не будуть викликати проблем при малих кількостях записів, та якщо вони досягатимуть мільйонів, повторне збереження ключових слів може призводити до dump пам’яті – перебільшення використаних об’ємів даних, що можливі для обрахування даною системою. Чим більше даних буде зберігатися в базі даних, тим більше інформації дублюється і тим вище непродуктивні витрати. [12]

Потенційна суперечливість (аномалії відновлення). Унаслідок наявності безлічі копій тих самих даних можлива ситуація, коли одна частина надлишкових копій даних буде змінена, а інша - ні.

Аномалії включення. В базі даних не можна включити суб’єкт, якщо він вже існує. Однак часто буває, що необхідно врахувати всі дані, що можуть потенційно використовуватися.

Аномалії видалення. Необхідно зберігати данні при видаланні батьківського елемента, інакше суб’єкти баз данних можуть бути втрачені. Однак ці дані можуть знадобитися в майбутньому, через що буде потрібно їхнє повторне введення.

Велика частина проблем зникне, якщо дані з універсального відношення рознести в кілька дрібних таблиць. Саме цю задачу і вирішує нормалізація. При першому знайомстві нормалізація здається досить простим заняттям, однак у ході цього процесу можуть виникнути визначені труднощі, наприклад, часто приходиться змінювати склад стовпців у таблицях, щоб мати можливість зв'язувати їх. Процес нормалізації

розбивається на кілька етапів. На кожнім з етапів структура даних повинна задовольняти визначеним вимогам. Таблиця вважається нормалізованою на визначеному рівні, якщо вона задовольняє вимогам, висунутим відповідною формою нормалізації (нормальної форми).[13]

3.2 Обгрунтування та вибір СКБД

Для даного програмного продукту MySQL Server.

MySQL побудована на основі архітектури клієнт-сервер, що дозволяє розбивати процес обробки інформації на два компоненти - попередню обробку даних - клієнтський компонент, і остаточну обробку - серверний компонент. Server предстає у вигляді сервісу бази даних, що забезпечує їх остаточну обробку та дозволяє проводити взаємодії і перетвореннями з декількома різними клієнтськими компонентами, розташованими, як правило в одній мережі (LAN). Він має вбудовану підтримку реплікації даних, могутні інструментальні засоби і відкриту архітектуру, що забезпечує йому репутацію надійного й ефективного інформаційного рішення для організацій усіх розмірів. SQL Server являє собою закінчену інтегровану систему керування базами даних, що задовольняє всім сучасним вимогам побудови масштабованих розподілених інформаційних систем.

MySQL Server має у собі такі функції:

- Relational database management system (RDBMS). Структурно, особливість побудови даних не відрізняється від реляційних моделей бази даних і дозволяє проводити з даними операції відповідно до правил реляційної алгебри, уперше сформульованими Е.Ф.Коддом у 1970р.
- SQL-based. Кожна база має власні особоливості побудови запитів. Проте, як і більшість систем, MySQL має чітко сформоване формування запитів, що логічно побудовані за «людською логікою» та не викликають

складнощів під час їх формування. Адміністратори, користувачі і прикладні програмісти застосовують Structured Query Language (SQL).

- Масштабованість. Окрім попередній пунктів, в нас є ще одна можливість – створення окремих унікальних ідентифікаторів, окрім, звичайних полей ID. Це зроблено для підключення інших баз даних або їх перенесенням до наявної. Таким чином, допускається зниження колізійних моментів під час міграції даних для їх використання. Варто зауважити, для того щоб уникати колізії – потрібно максимально зменшити вірогідність співпадінь із згенерованого хешу, тому генеровано поле, зазвичай, являє собою 128-156 бітний хеш.
- Висока продуктивність. MySQL система, також, має децентралізоване ядро пам’яті, що дозволяє уподібнити його до віртуальної машини. Зручна структура архівації та індексації даних. Окрім цього, вона вдосконалена оптимізацією та підтримкою Laravel Builder, що дозволяє будувати окремо вивірені запити у репозиторіях без передачі Query запитів у стрічковому форматі, що дозволяє ще менше затрачувати пам’яті на опрацьовування даних.[14]

MySQL вважається гарним рішенням для малих і середніх застосувань. Сирцеві коди сервера компілюються на багатьох платформах. Найповніше можливості сервера виявляються в UNIX-системах, де є підтримка багатопоточності, що підвищує продуктивність системи в цілому.

Можливості сервера MySQL:

- Звичність та легкість у встановленні та використанні;
- Підтримка необмеженої кількісті користувачів, що одночасно працюють із БД;
- Величезна кількість рядків для таблиць (50 мільйонів);
- висока швидкість виконання команд;
- наявність простої і ефективної системи безпеки. [15]

У середовищі баз даних клієнт-сервер, сервер автоматично забезпечує цілісність даних. Використовується декілька механізмів підтримки цілісності. В свою ж чергу, забезпечення декларативної посилальної (співвідношення таблиць) цілісністності (DRI), що дозволяє користувачам встановлювати обмеження на дані і співвідношення між таблицями для узгодження ключових слів таблиць дозволяє в рази скорочувати необхідні обрахування. Це є необхідним, також, для узгодження цілісності правил збереження даних або перехресних посилань таблиць.

Для зміни інформації бази даних були погоджені окремі методи, що ідентифікують поля за допомогою ключових сегментів стрічки запиту. Щоб забезпечити сутнісну цілісність записів у таблиці, MySQL підтримує унікальні індекси, що гарантують, що значення ключа в стовпці унікально для всіх записів таблиці. Надзвичайно зручним є також те, що використання полів менеджменту дати створення, редагування, та видалення – є автоматичними. Тобто, система самостійно розуміє, що відбувається з даними. Використання параметрів за замовчуванням і правила, яким повинні задовольняти дані, що зберігаються в таблиці для забезпечення доменної цілісності даних у таблиці, що гарантує, що значення даних стовпця законно.

Координатор розподілених транзакцій. Використання даної функції, дозволяє реалізовувати транзакційні запити то MySQL та опрацьовувати кортежі згрупованих даних із декількох таблиць. Подібне формування дозволяє створювати «абстракційне» відношення таблиць, що можна вважати окремою віртуальною таблицею.

Реплікація. Існує необхідність, коли для даних користувачу необхідно поширювати копії транзакційних даних від одного сервера підприємства на один чи кілька віддалених серверів. Хоча MySQL Server НЕ підтримує можливості багатопроцесорного обраховування даних за замовчуванням, існують різні програмно-апаратні рішення, що допускають подібне, а також включають у собі додаткові функції такі як:

- Розширений контроль цілісності бази даних
- Попереднє сканування та індексація даних (асинхронне попереднє читання) послідовних сторінок операцій
- Можливість додавання ключів і резервних слів
- Системно - збережені процедури [15]

Недоліком системи – є відсутність зручного інтерфейсу опрацьовування бази даних напряму. Для цього була встановлена додаткова система графічного інтерфейсу для контролю бази даних HeidiSQL. Дана система є безкоштовною та має можливість генерування різних баз даних і користувачів.

3.3 Структурний опис побудованої системи користувачів

На даний момент, програмне рішення представляє собою декілька рівнів наслідування між таблицями. У базі існують такі взаємозв'язки: account – user – owner.

1) Акаунт – представляє собою місце організації чи установи, на якій працює користувач. Це зроблено з метою розподілу збереження інформації на окремі доступні поділки для користувачів.
2) Користувач – цей статус отримує будь-який створений запис через інтерфейс реєстрації
3) Власник – це особливий статус користувача, що отримує значно більше прав, аніж звичайний користувач.

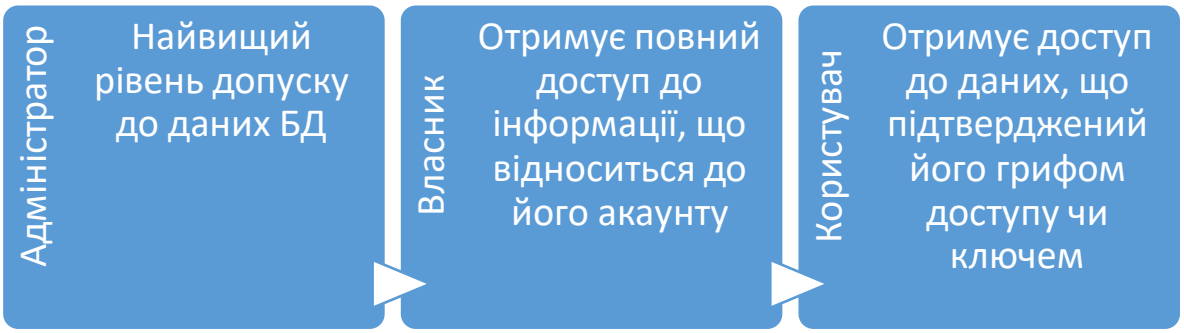


Рисунок 3.3.1 – Схема рівнів доступу до інформації

Дана система підтримує будь-які типи даних, що придатні до збереження.

У випадках стрічкових масивів (текстового формату даних) збереження відбувається у таблиці Files. А якщо файл уособлює собою масив даних чи об’єкт класу для його збереження сформована окрема таблиця, що наслідує ключ у попередній таблиці. Подібне формування створено з метою оптимізації звернень до бази даних. Під час звернення до системи ми опрацьовуємо основну таблиці Files, а якщо нам буде потрібним окремий файл, ми зможемо звернутись до окремої таблиці лише за індексом, що надасть нам швидке завантаження лише 1 запису із таблиці.

3.4 Опис діалогу роботи автоматизованої системи

Представимось новим користувачем системи для пояснення реалізації системи. Усі механізмі автоматизовані та створені з метою поліпшити зручність користування та підтримати захищеність системи. Після стартової сторінки (Рисунок 3.3.3) ми отримуємо посилання аутентифікації в телеграмі.

Як це працює:

Рисунок 3.3.2 – Схема створення нового користувача

Кожен етап підтриманий внутрішнім мідлварами та валідацією, що не дає змоги проводити SQL інкапсуляцію, або будь-які сторонні дії всередині сервісу.

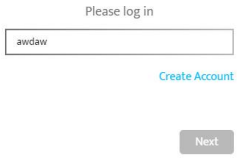


Рисунок 3.3.3 – Стартова сторінка

Після створення запиту для нового користувача, ми отримаємо повідомлення такого виду:

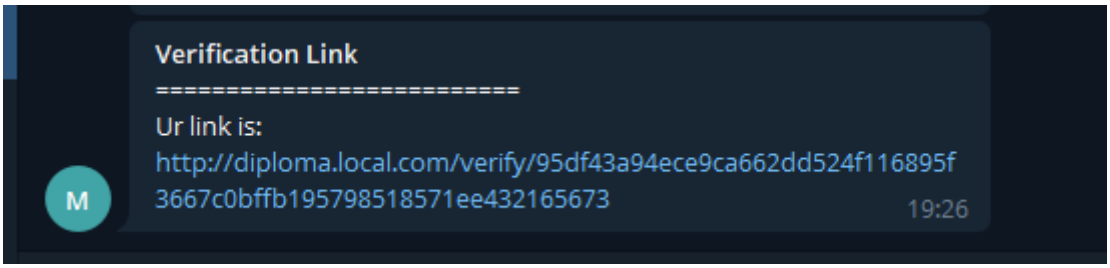


Рисунок 3.3.4 – Вигляд отриманого посилання

Дане посилання є одноразовим та валідним протягом доби. Якщо користувач – не встигне його використати, йому доведеться знову надавати запис для створення облікового запису.

Після переходу за посиланням ми потрапляємо на сторінку підтвердження створення користувача

Рисунок 3.3.5 – Підтвердження входу до облікового запису

Єдине попередження в системі – це не розповсюджувати посилання на відкритих джерелам або неперевіреним особам до його активації, це зумовлено тим, що заволодівши даним посиланням вони здатні створити обліковий запис від імені справжнього користувача, а видалити такий обліковий запис здатні лише адміністратори або власники акаунту в системі.

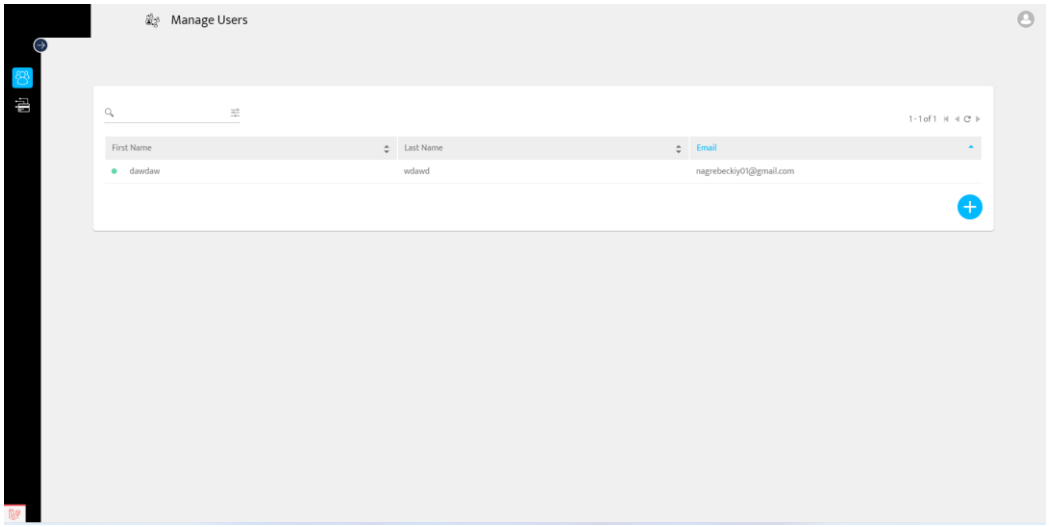


Рисунок 3.3.6 – Підтвердження входу до облікового запису

На даному слайді видно частину інтерфейсу, що створено для керування користувачами системи. Будь-хто здатен надіслати окремий запит для створення нового користувача.

Окрім цього, було створене повноцінне оформлення профілю користувача, яке він здатен редагувати на власний розсуд.

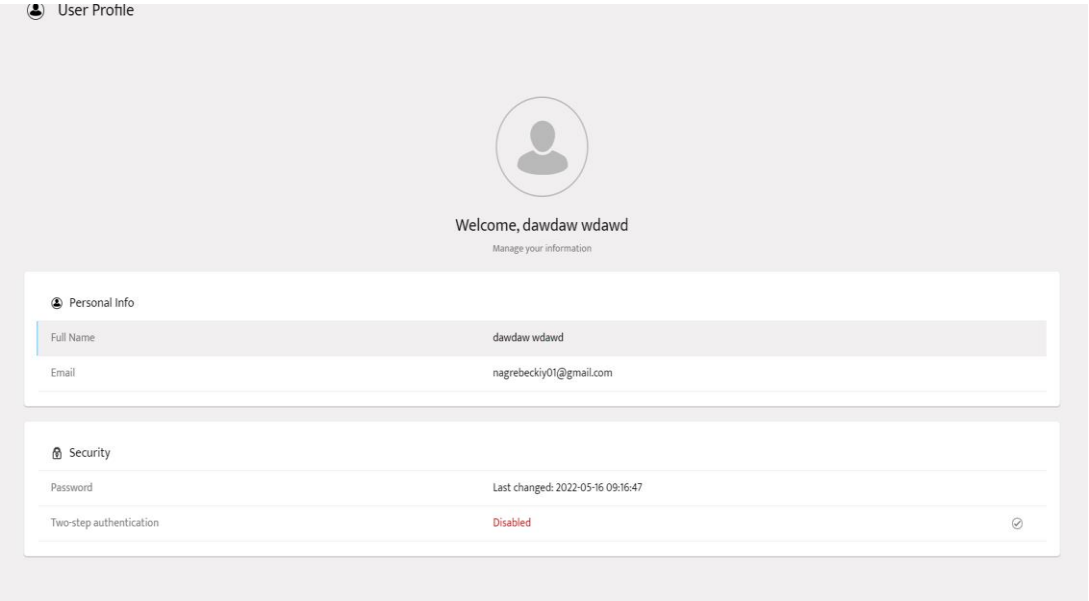


Рисунок 3.3.7 – Зовнішній вигляд сторінки користувача

Для користувачів створені такі можливості

1) Зміна ідентифікуючих полів
2) Увімкнення двухетапної аутинтефікації
3) Зміна конфіденційної інформації (паролю та електронної пошти)

Зміна паролю, ввімкнення і вимикання двухфакторного підтвердження та електронної пошти супроводжуються підтвердженням паролю. А для зміни пошти чи самого паролю, ще необхідно використати підтвердження із посиланням верифікації.

3.5 Реалізація API – запитів програми

Sanctum існує для того, щоб запропонувати простий спосіб аутентифікації односторінкових застосувань (SPA), яким необхідно взаємодіяти з API, працюючими на Laravel. Ці SPA можуть існувати в тому ж репозиторії, що і ваш додаток Laravel, або можуть бути абсолютно окремим репозиторієм, наприклад SPA, створеним за допомогою Vue CLI або додатка Next.js.

Для цієї функції Sanctum не використовує які-небудь токени. Замість цього Sanctum використовує вбудовані в Laravel сервіси аутентифікації сесії на основі cookie. Зазвичай Sanctum використовує для цього захист веб-аутентифікації Laravel. Це забезпечує захист від CSRF, сеансову аутентифікацію, а також захищає від витоку аутентифікаційних даних через XSS. Sanctum намагатиметься аутентифікуватися за допомогою cookies тільки у тому випадку, якщо запит, що входить, виходить від вашого власного фронтенда SPA. Коли Sanctum розглядає HTTP- запит, що входить, він спочатку перевіряє наявність аутентифікаційного cookie, і якщо такого немає, Sanctum перевіряє заголовок Authorization на наявність дійсного API- токена. [7]

Hash 256 – багатоетапний вид шифрування даних за методом Меркла – Демгарда, оскільки, ми будемо зберігати багато даних користувачів, необхідно створити унікальні коди для кожної з дій та шифрування конфіденційної інформації (втручання до бази даних – не принесе зловмиснику інформації, її ніхто не зможе зрозуміти, окрім, самої програми.) [9]

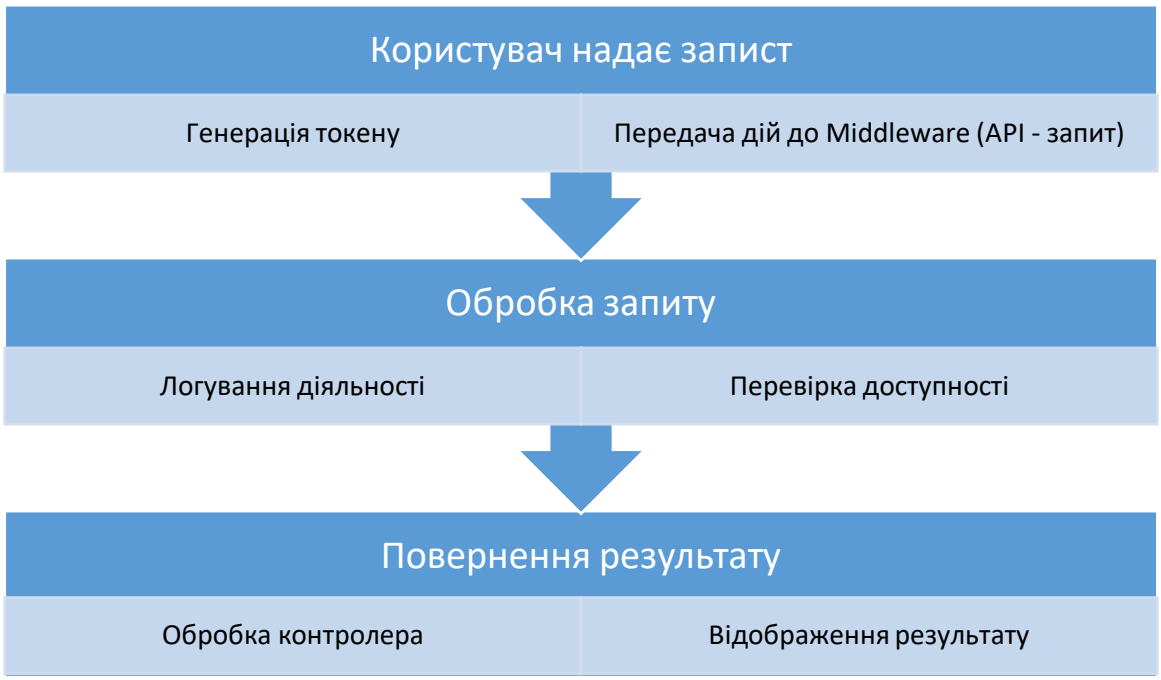


Рисунок 3.3.8 – Схема почергової обробки запиту

Перейдемо до результатів виконання програми.

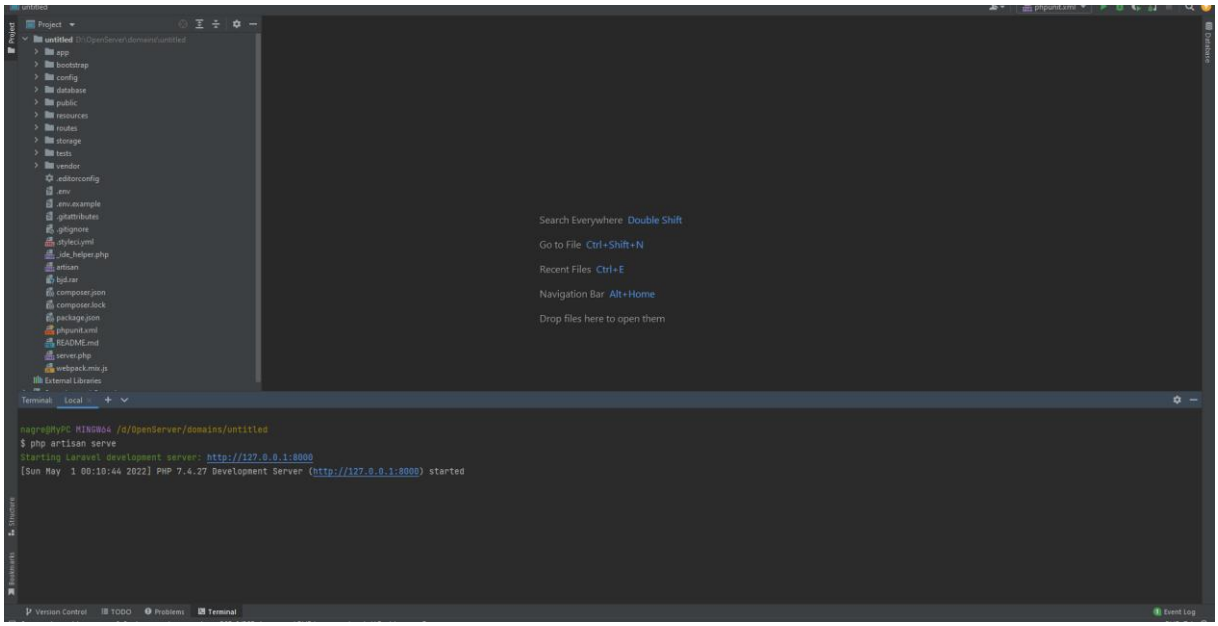


Рисунок 3.3.9 – Генерація тимчасового серверу

На даному слайді вказані можливості PHP Artisan, дана бібліотека дозволяє створити окремий тимчасовий сервіс із нашим фукнціоналом та підтримувати його роботоздатність стільки, скільки нам необхідно.

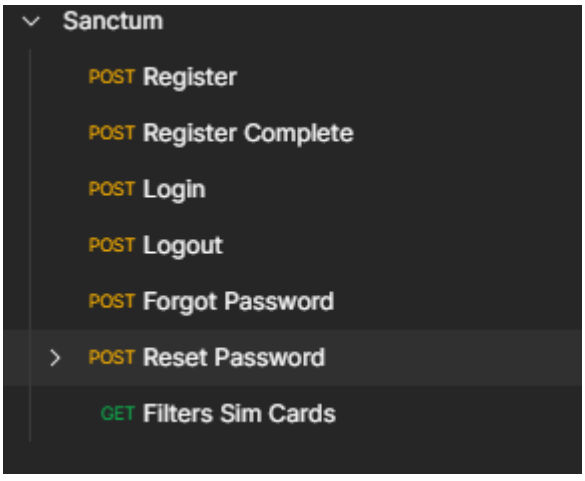


Рисунок 3.3.10 – Створені запити

```php
<?php
Route::group(['middleware' => ['api_auth']], static function() {

});
Route::group(['middleware' => ['auth:sanctum']], function() {
    //Private Routes
    Route::get( uri: '/testRequest', action: 'Api\SanctumApiController@testRequest');
    Route::post( uri: '/logout', action: 'Api\SanctumApiController@logout');
});
//Public Routes
Route::post( uri: '/register', action: 'Api\SanctumApiController@register');
Route::post( uri: '/registerComplete', action: 'Api\SanctumApiController@registerComplete');
Route::post( uri: '/forgotPassword', action: 'Api\SanctumApiController@forgotPassword');
Route::post( uri: '/resetPassword', action: 'Api\SanctumApiController@resetPassword');
Route::post( uri: '/login', action: 'Api\SanctumApiController@login');
```

Рисунок 3.3.11 – Розподіл запитів на публічні та приватні

Завдяки Sanctum, вдалося розподілити функціонал на публічний та приватний. Публічні «рути» - дозволяють користувачам реєструватись (у два етапи), заходити до облікових записів, відновлювати пароль (у два етапи). Внутрішні ж можливості працюють із згенерованими ключами в системі, що не дозволять створити запитів, які ми не передбачали, або ж, додаткову кількість запитів, які порушують умови Посередника.

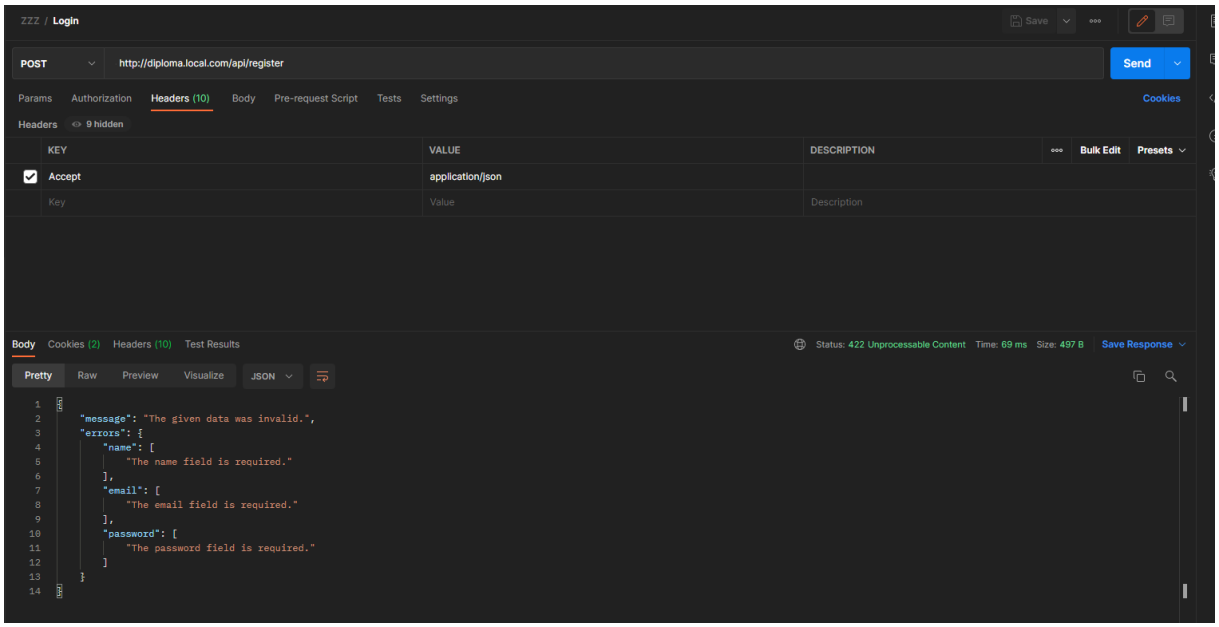
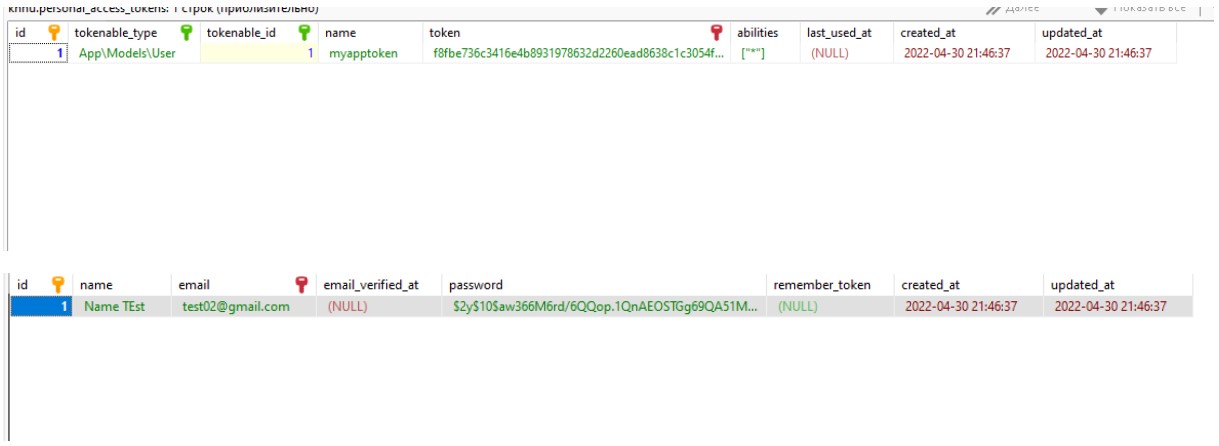


Рисунок 3.3.11 – Демонстрація

Як видно із запиту при реєстрації, в нас утворився повноцінний, що не дозволяє користувачам виходити за передбачені рамки програми. По-перше, існує валідація для кожного нашої дії. По-друге, ми побачимо логування дій користувачів



id	tokenable_type	tokenable_id	name	token	abilities	last_used_at	created_at	updated_at
1	App\Models\User	1	myapptoken	f8fbe736c3416e4b8931978632d2260ead8638c1c3054f...	["*"]	(NULL)	2022-04-30 21:46:37	2022-04-30 21:46:37

id	name	email	email_verified_at	password	remember_token	created_at	updated_at
1	Name TEst	test02@gmail.com	(NULL)	$2y$10$aw366M6rd/6QQop.1QnAEOSTGg69QA51M...	(NULL)	2022-04-30 21:46:37	2022-04-30 21:46:37

Рисунок 3.3.12 – Згенеровані ключи та користувач від попередньої дії

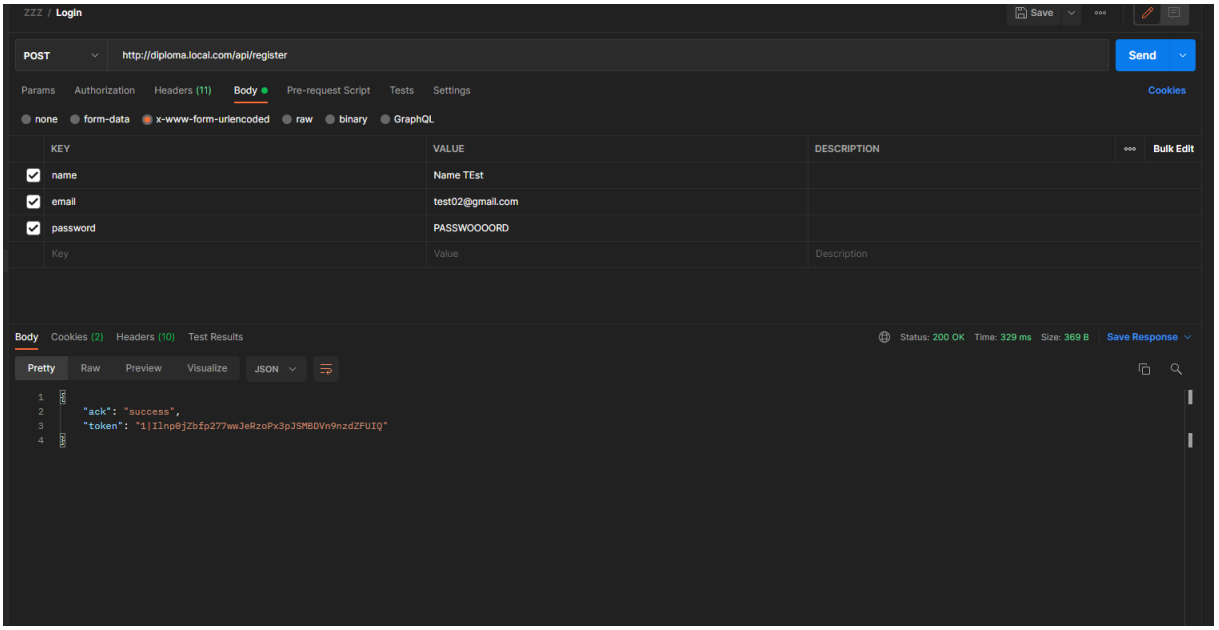


Рисунок 3.3.13 – Згенерований токен у відповідь на реєстрацію

Даний виведений токен – це показовий вивід запиту при реєстрації. Був створений окремий SanctumApiController, що приймає на себе валідацію,

перевірку ключів та основні методи посередника, що дозволить благополучно інтегрувати його до будь-якого ПЗ.

Створені токени – існують виключно у самій програмі, навіть, якщо порушники отримають доступ із бази, копіювання токену – не дасть результатів.

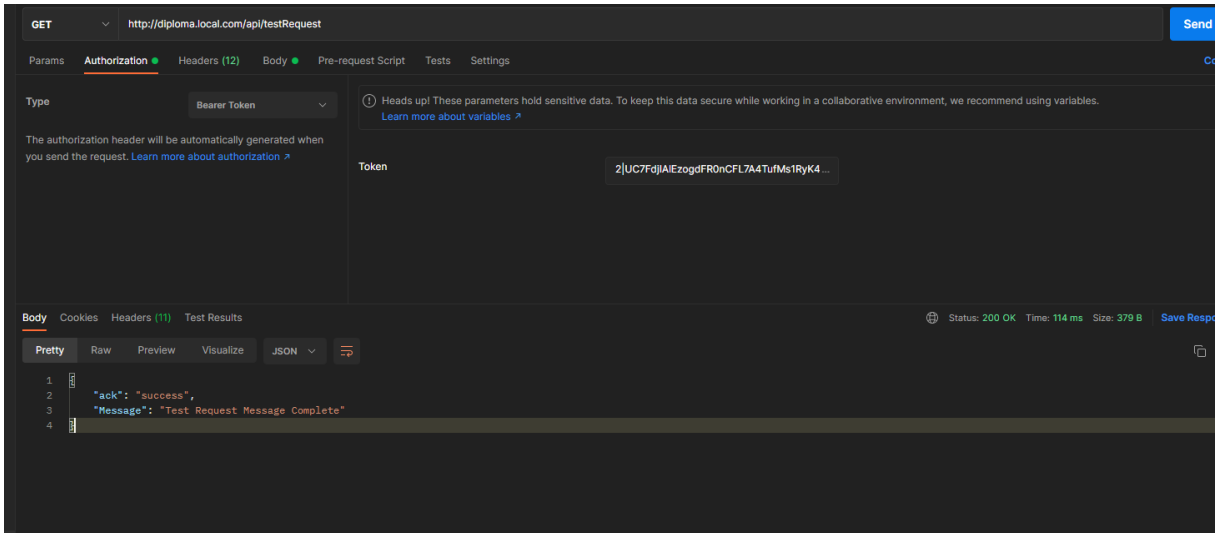


Рисунок 3.3.14 – Приклад відповіді зі токеном користувача

Дане рішення дозволить інтегрувати нашу базу даних до інших систем. Проте, для повноцінного функціоналу програми подібний функціонал черезвичайно громісткий. В ідеалі, такі функції повинні опрацьовувати повністю створений функціонал та передачі необхідної інформації... Та для його реалізації витрачається значно більше часу, аніж при побудові інтерфейсу користувача. Причиною цього є зовсім інша логіка опрацьовування запитів та зважене рішення, що є необхідним, а що ні.

Занадто багата бібліотека буде вводити користувачів у паніку, адже розібратись для такого сервісу необхідні знання та час. До того ж, виникає проблема агресивного парсингу даних, коли до нашого сервісу можуть надаватись безліч запитів постійно, для моніторингу якої небудь інформації

у реальному часі, що здатне викликати збої при недостатній потужності вираховувальної машини.

Деякі компанії цілеспрямовано блокують можливі фунції API документацій, це дозволяє не лише заощадити на різних серверах для їх підтримки, а й небажаного доступу до самого програмного забезпечення компанії. Дане практичне рішення є виправданим та додатково захищає сервіс від незапланованих вторгень із зовні.

3.6 Висновки про створення програмне забезпечення

Дана система являє собою комплексну систему опрацьовування інформації із взаємодією із локальною базою даних. Передача різного виду інформації, що обробляється являє собою масив даних із шифрованим токеном користувача, що захищає систему від підробок, видавання зловмисників за іншу особу, ризикових дій, що націлені на безпосереднє пошкодження та внесення помилок у системі.

Були реалізовані такі механізми у системі:

1) Повноцій обліковий запис користувачів та аутинтефікація
2) Хешування та створення унікальних токенів користувачів під час роботи у середовищі
3) Розподіл інформації на рівні доступу та шифрування конфіденційних та важливих даних
4) Децентралізація доступу від локальної мережі шляхом API – запитів до програмного продукту із використанням токенів користувачів
5) Ведення обліку користувачів
6) Побудована архітектура опрацьовування програми використовує методи Ajax, що дозволяє швидке генерування окремих відділів сторінки при взаємодії з користувачем, а також опрацьовування запитів «під капотом», що не дозволяє уявити методи опрацювання системи стороннім особам
7) Створено методи верифікації та підтвердження користувачів за допомогою нотіфікацій платформі Телеграм

8) Побудовано метод двухетапної аутинтефікації за допомогою QR - коду

Ціна створеної системи – може вважатись безкоштовною =).

Основні витрати являють собою час та виробничі витрати, вигляду електрики на технічне устаткування

4 ТЕСТУВАННЯ СТВОРЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Пояснення резонності проведення тестових атак на створене ПО

Етичний хакінг, відомий як етичний взлом, тест на ручку або пентест, є авторизованим модельовано згенерованим методом перевірки на рівень захисту від кібератак на комп'ютерні системі, що проводиться для оцінки безпеки системи.

Тести проводяться з метою виявлення обох слабких сторін (їх також називають вразливими місцями), включаючи можливість несанкціонованих сторін отримати доступ до функцій та даних системи, а також сильні сторони, уможливлення повного.

Подібні методи – не нові та навіть мають відповідні інститути вивчення даної проблеми.

Їх користь полягає в тому, що створений сервіс зможе гарантувати безпеку користувачів. Проте, його користь досить відносна, оскільки, недоцільні витрати для захисту нікому непотрібної інформації – є безглуздям.

У наш час, розповсюджувачі ПЗ, навіть, створюють власні лабораторії досліджень з цього приводу. Подібні існують і на військово направлених компаніях з виробки устаткувань та техніки. Питання, що піднімаються під час пентестів можуть нести критичний характер і потребуватимуть негайних рішень.

Всесвітній економічний форум назвав кібератаки п’ятим за значимістю ризиком 2020 року. Оскільки 77% керівників служб безпеки прогнозують порушення критичної інфраструктури, для великих і малих підприємств важливо бути готовими до такого розвитку подій.

Тим більше, за прогнозами до кінця 2022 року вартість IT-індустрії досягне 170,4 млрд доларів, що означає швидке зростання фінансових кібер-

ризиків. Кібератаки можуть статися в будь-який час, тому потрібно зустрічати їх напоготові та приймати правильні рішення.

Атаки можуть бути активними або пасивними, відбуватися зсередини або зовні організації. Якщо розпізнавати атаки на ранній стадії, організації можуть заощадити гроші та запобігти подальшому доступу до конфіденційної інформації, відключивши системи й повідомивши зацікавлені сторони. [18].

Також, існують окремі сервіси аутсорсу, невеличкі компанії, що займаються етичним хакінгом з метою показати прогалини захисту даної версії ПЗ.

4.2 SQL ін’єкція в систему

Це вид однієї із найпоширеніших кібератак на веб-серверні ресурси. Пов’язаний він з тим, що будь-який сервіс спілкується із базою даних від власного імені за допомогою стрічкових запитів. Таким чином, внесення певного роду коду, у стрічки, які передаватимуть інформацію до системи можуть провокувати різні дії зі сторони бази даних. Цей вид атаки може проводитись із ціллю заволодіння інформацією, завантаження різного виду шкідливого ПЗ, або з ціллю викликати внутрішні помилки, через які буде відмова у обробці запитів серверу.

Дана проблема – далеко не нова і її надійним рішенням було інтеграція до базових методів вводу валідації та комплексного захисту від самого браузера. До прикладу Google чи TorBrowser – будуть одразу блокувати запити у моделі input, якщо вони будуть схожими на шкідливі.

В створеному ПЗ, створена валідація полів вводу та використовується MiddleWare від самих розробників Laravel, нажаль, дана перевірка – не є реалізовано можливою, оскільки, будуть викликані помилки, що будуть зведені до звичайних.

Зокрема, визначенності інакшості у трактації програмного тестування забезпечення – не існує. Оскільки, тестування повинно переслідувати конкретні цілі компанії чи організації.

Всіма використовуються різні методики, практики та етапи тестування програмного забезпечення, тому що, існують чітко сформовані норми до відповідної вразливості, а шлях для досягнення – різний.

4.3 Приблизний вигляд можливих кібератак через користувачів

1) Спроба атаки на особистий комп’ютер користувача, що підлючений до мережі.

Користувач намагаючись знайти корисну для себе інформацію у мережі інтернет, надав доступ зловмиснику до власного ПК, натиснувши на невідоме посилання. Це звичайна ситуація в наш час, а особливо для людей літнього віку, котрі не дуже освідомлені в планах кібергігієни... У подібних випадках, зазвичай, спрацьовує антивірус, при переході з гіперпосилання, або завантаженні шкідливого ПО. Подібний захист оминути просто, чи повідомивши на сайті неначе «Ваш антивірус може блокувати завантаження, або сповільнювати його, вимкніть антивірус та продовжуйте», створювати – дублікати (невидимки), досить свіжі вірусні продукти, що є дочірніми до відомих програм, проте, ще не занесені до словників антивірусних програм, або ж напівшкідники – трояни, черви і т. д., програми, що є відповідними та мають невиличкі зміни на користь хакера. Або ж архівовані файли, оскільки антивірус перевіряє не лише назву файлу, а й його зміст, стиснені або зашифровані файли – неможливо перевірити на всі 100%, чим і користуються злочинці.

В даному випадку для створеної програми головною ціллю буде захистити конфіденційні данні у БД. Беручи до уваги, що безпосередні гарантії захисту надає сам продукт MySQL, навіть, у випадку його нестачі – зловмисники не зможуть використати дані з максимальною ефективністю. В їх розпорядженні буде конфіденційна інформація, проте, вони не зможуть нею скористатися через те, що вони не зберігаються у звичному вигляді.

2) Хакер отримав доступ до ПК, що підключений до мережі, що далі?

Наступним, йому необхідно оминути захист системи самої бази даних і він матиме 2 шляхи. Перший – намагатись відгадати логін та пароль користувача, щоб попередити подібні вчинки усьому персоналу були

повідомлені вказівки, щодо їх паролів та логінів, збережено двохфакторну аутинтефікацію при підозрілих вчинках над БД, використовуються складні паролі довжиною 8 символів з високим символьним діапазоном та лімітом на кількість введень паролів.

Другий спосіб – це приєднатись напряму до локального сервера та атакувати БД напряму. Тут існує декілька можливих атак. Першою є маскування під адміністратора, друге – атака самого сервісу MySQL Server. Проте, як показує практика для організації подібної атаки на сервіс, необхідно витратити значно більше коштів та часу, аніж цінність внутрішньої інформації... Отримання доступу до журналу транзакцій не зможе призвести до достатньої загрози, оскільки створені токени користувачів прив’язані строго до часу та їх особистий даних. Так, у системі залишено відкритим журнал транзакцій користувачів, проте, він зашифрований ключем RSA – 128, це створено з метою «відбою». Якщо ж сервіс вдалося взламати та йде атака на БД з використанням ПК користувача, адміністратор зможе одразу ж від’єднати ПКП від системи відкривши файл транзакцій та переглянути його на активність. Маскування під адміністраторів можливе лише з унікальними ідентифікатора адміна БД – ключа підключення до бази даних (256 – бітний згенерований ключ) чи використанням журналу автоматичного входу (перевіряє унікальний номер процесора). Нажаль сервіс MySQL Server, справді має деякі недоліки у плані захисту інформації, проте, обраховуючи цінність інформації, що знаходиться на сервері, використання ще більш сильного захисту є недоцільним.

4.4 Проведення пробних атак на створений сервіс за домопогою сторонніх програм

NetLimiter – додаткове програмне забезпечення для моніторингу та транслювання трафіку у під’єднаній мережі. Шкідливе тим, що злочинець

отримавши доступ до вай-фаю може отримувати пересланні дані, перехоплюючи їх до провайдерського хосту.

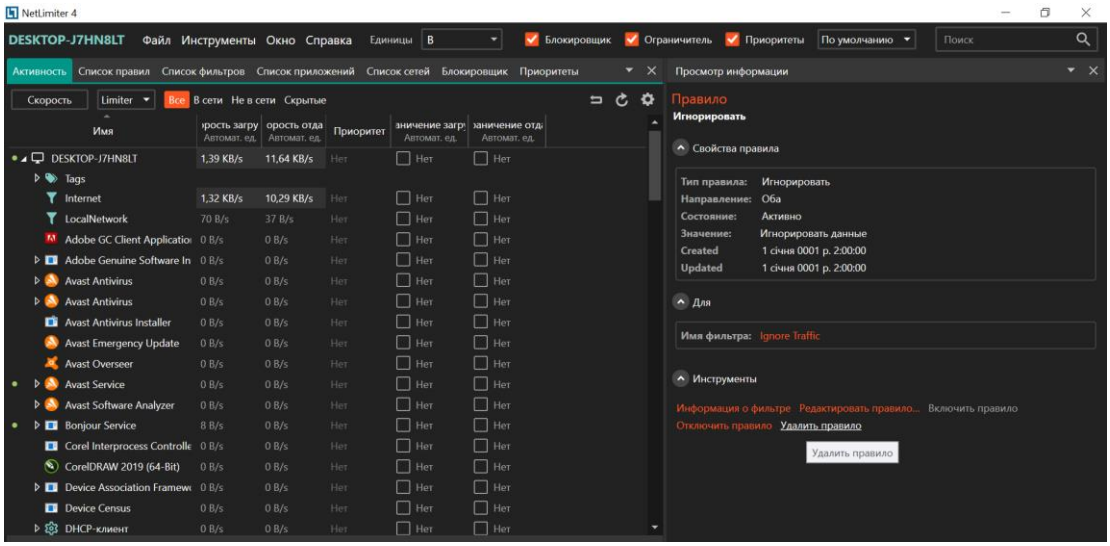


Рис. 4.4.2 – вікно програми NetLimiter та дані трафіку

Як ми бачимо, справдні є трафік на локальній підмережі, проте, беручи до уваги, що дана система є локальною, щоб виконати ці дії необхідно мати до неї доступ, або ж доступ до ПК у цій сітці, що дає більше проблем для взлому.

Проведемо ще одну атаку зі словником, намагаючись оминути блокування кількості введень:

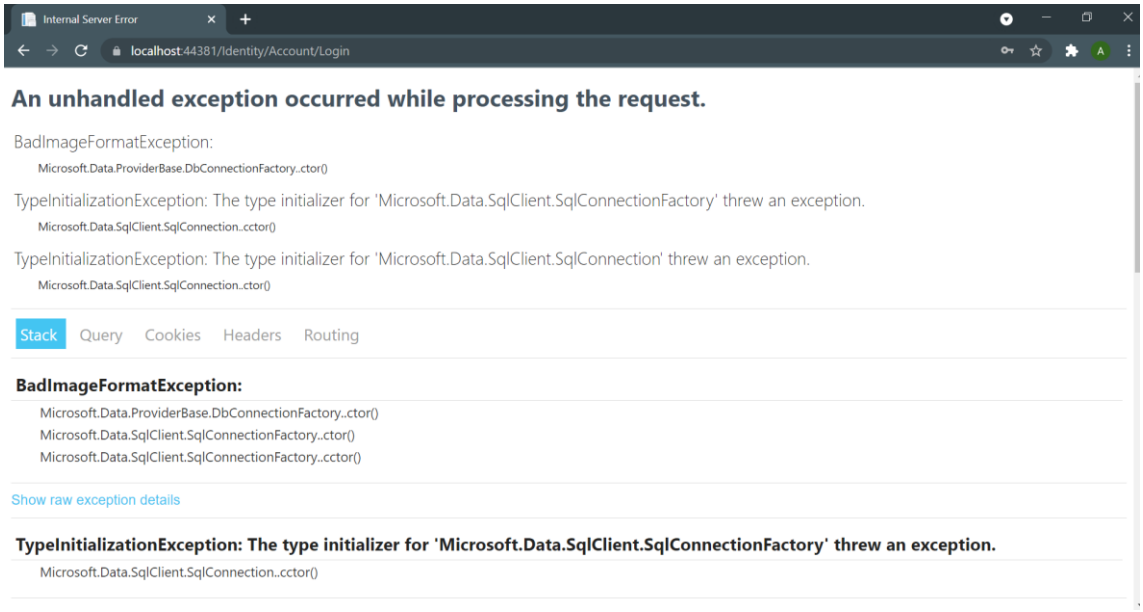


Рис. 4.4.3 – Спроба атаки словником на сервіс за допомогою програми HashCat

Що ж, нам вдалося, однак для подібної атаки ми витратили 74 хвилини, а результатом є трохи опрокинутий сервер, через величезну кількість одночасних запитів та його перезавантаження... База відмовилась виконувати своє завдання.

Якщо взяти до уваги, що дана програма використовує потужність стаціонарного ПК, для взлому локальної системи (що проходить з максимальною швидкістю трафіку) то це відмінний результат

ВИСНОВКИ

Дане програмне забезпечення є результатом здобутку знань протягом освоєння спеціальності, звичайно, тут не внесені всі отримані знання, через недостатню кількість часу та складність реалізації кожної з систем.

На основі аналізу та систематизованого підходу, було розроблено було створене середовище, що здатне гарантувати захист користувачів на відповідному рівні.

Пропозиції ринку, розуміння необхідного для замовника, пошук рішень та альтернатив, результати тестувань та характеристики технічних засобів. Дане завдання, стало втіленням результатів клопіткої праці багатьох спеціалістів різних сфер, в результаті ми отримуємо продукт, що служитиме для сервісу – захисту громадян.

Після проведення аналізу та синтезу комплексної системи захисту інформації були покращені не лише навички програмування та кіберзахисту, а ще й soft-skills, досвід інтегрування проекту в реальну область робочого процесу.

Для реалізації повноцінного продукту, необхідно було спілкування із працівниками та керівниками компанії з метою реалізації їх побажань, проведення тестового інтегрування системи та корегування її роботи.

Однак, вимушений примітити, що бездоганної комплексної системи захисту – не існує, адже неможливо попередити будь-яке проникнення, недбалу помилку персоналу, технічну чи інформаційною «ерозію», кожен злочинний крок, що намагатиметься вчините своє для власної вигоди.

Кібербезпека – це своєрідний ультиматум на беззаконня, адже наше життя просякнуте технічними засобами, що не лише мають високу собівартість, а й використовуються для передачі особистої та конфіденційної інформації, банківських послуг та зв’язку. Тому спеціалісти даної сфери будуть завжди актуальними.

СПИСОК ДЖЕРЕЛ

1. Silberschatz, Abraham; Sudarshan, S. (2011). *Database system concepts* (вид. 6). New York: McGraw-Hill. ISBN 9780073523323. OCLC 436031093
2. *С.Д. Кузнєцов.* Об'єктно-орієнтовані бази даних – основні концепції, організації та їх управління. CIT Forum, 2014 – 551 с.
3. Адамик О.В. Бази і сховища даних – інформаційний фундамент обліку та аналізу [Текст] / Оксана Василівна Адамик [та інші] // Економічні, управлінські, правові та інформаційнотехнічні проблеми діяльності підприємств: колективна монографія/ за заг. ред. Л.М. Савчук, М. Фіц. – Дніпро: Герда, 2016. – 528 с. ISBN 978-617-7097-58-6. – С. 330-341.
4. Liz Rice. Container Security: Fundamental Technology Concepts that Protect Containerized Applications. — 1005 Gravenstein Highway North, Sebastopol, CA 95472, 2020. — 4-12, 23-53 с.
5. *Стивен Хольцнер.* Ajax Біблія програміста = Ajax Bible. – М.: Діалектика, 2009. — С. 553. — ISBN 978-5-8459-1502-3.
6. РОЗРОБКА ДОДАТКУ ПІД LARAVEL ДЛЯ ПОШУКУ ДАНИХ - Wayback Machine.[ЕЛЕКТРОННИЙ РЕСУРС] http://creative-punch.net/articles/php-articles/laravel-tutorials/
7. PostMan [Електронний ресурс]: POSTMAN is an API platform for building and using APIs – Режим доступу : https://www.postman.com/. Назва з екрана.(електронне джерело)
8. The Hacker Playbook 3: Practical Guide To Penetration Testing - Peter Kim, США 2018, 2018 ISBN-13: 978-1980901754
9. Laravel Sanctum [Електронний ресурс]: Laravel Sanctum – The PHP Framework For Web Artisans – Режим доступу : https://laravel.com/docs/9.x/sanctum. Назва з екрана.(електронне джерело)
10. PHP Artisam [Електронний ресурс]: - відкрите інтернет джерело для програмування – Режим доступу : https://laravel.com/docs/9.x/artisan

11. Difference between Normalization and Denormalization [Елентронний ресурс] – Nitin Sharma Режим доступу - https://www.tutorialspoint.com/difference-between-normalization-and-denormalization#:~:text=Normalization%20is%20used%20to%20remove,it%20can%20be%20queried%20quickly.

12. Інформаційна надмінність [Електронний ресурс] – Режим доступу - https://jak.koshachek.com/articles/informacijna-nadmirnist.html

13. Efficient MySQL Performance: Best Practices and Techniques - Daniel Nichter – 2021 p. 94-147 c.

14. High Performance MySQL: Proven Strategies for Operating at Scale - Silvia Botros, Jeremy Tinley – 2021 p. – 64 – 88 c.

15. PHP and MySQL for Dynamic Web Sites - Larry Ullman – 2017 p. – 140-148 c.

16. HASH 256 [Електронний ресурс]: Що таке хеші MD5, SHA-1 і SHA-256 і як їх перевірити? Автор: Jorge Stolfi / Wikimedia [Електронний ресурс]: - https://ua.phhsnews.com/articles/howto/what-are-md5-sha-1-and-sha-256-hashes-and-how-do-i-check-them.html

17. [ЕЛЕКТРОННЕ ДЖЕРЕЛО] Найпопулярніші кібератаки 2020-2021 років – Режим доступу - https://10guards.com/ua/articles/the-most-common-types-of-cyber-attacks-in-2021/.

18. Бойко В.В., Савинков В.М. Проектування баз даних із включенням складних багаторівневих запитів. – М.: Фінанси та статистика, 2013. – 351 с.

19. PHP, MySQL, & JavaScript All-in-One For Dummies. - Richard Blum – 2018 – 210 – 234 c.

20. PHP and MySQL Recipes - Frank M. Kromann – 2016 p.- 400 c.

21. Атре Ш. Структурований підхдід до організації баз даних. – М.: Фінанси та статистика, 2003. – 320 с.

22. Molinaro, A. (2020). SQL Cookbook: Query Solutions and Techniques for All SQL Users. In R. de Graaf (Ed.), (ст. 140–322).

23. Джерело програмних ресурсів, форум допомоги програмістам [Електронний ресурс] https://habr.com/post/181772/

24. Локазюк В.М. Контроль і діагностування обчислювальних пристроїв та систем. Навч. посібник для вузів. Хмельницький, ТУП, 1996. – 175 с.

25. Меєр М. Теорія реляційних баз даних. – М.: Світ, 1997. – 608 с.

26. Хаббард Дж. Автоматизоване проектування баз даних. – М.: Мир, 2010. – 294 с.

27. *Дейт К. Дж.* “Ведення в системи баз даних”. – М.: Вільямс, 2005. – ISBN 5-8459-0788-8

28. Introducing InnoDB Cluster: Learning the MySQL High Availability Stack - Charles Bell – 2018 p.

29. Закон України "Про інформацію". https://zakon.rada.gov.ua/laws/main/2657-12 Закон України "Про доступ до публічної інформації".

30. Закон України "Про захист інформаційно- телекомунікаційних системах". https://zakon.rada.gov.ua/laws/show/80/94-%D0%B2%D1%80

31. За Закон України "Про основні засади забезпечення кібербезпеки України" https://zakon.rada.gov.ua/laws/main/2163-19

32. Закон України "Про електронні документи та електронний документообіг". https://zakon.rada.gov.ua/laws/show/851-15

33. Постанова Кабінету Міністрів України "Про затвердження Правил забезпечення захисту інформації в інформаційних, телекомунікаційних та інформаційно-телекомунікаційних системах" від 29.03.2006 № 373. https://zakon.rada.gov.ua/laws/main/373-2006-%D0%BF

34. Постанова Кабінету Міністрів України "Про затвердження Загальних вимог до кіберзахисту об'єктів критичної інфраструктури" від 19.06.2010 № 518.

35. Постанова Кабінету Міністрів України "Про затвердження Типової інструкції про порядок ведення обліку, зберігання, використання і знищення

документів та інших матеріальних носіїв інформації, що містять службову інформацію" від 19 жовтня 2016 р. № 736.

36. ДСТУ 33960-96. Захист інформації. Технічний захист інформації. Основні положення. http://www.dsszzi.gov.ua/dsszzi/control/uk/publish/article?art id=38911&cat_id=38836 12. ДСТУ 33961-96 Захист інформації. Технічний захист інформації. Порядок проведення робіт. http://www.dsszzi.gov.ua/dsszzi/control/uk/publish/article?artid=38911&cat_id= 3 8836

КвРКБ.170282.18.01.13 Е8

Telegram - хмарний месенджер

Використовується для спілкування, розповсюдження інформації та контенту. Має велику кількість сервісів із зручною інтеграцією у систему

- спеціалізований сервіс телеграм по
них допоміжних інтерфейсів

Відповідає за створення токену для взаємодії із створеним ботом. Дана система генерує хешований ПОСТІЙНИЙ, токен для «спілкування» із створеним ботом

MessageHelper - створений бот для сповіщень

Система виконує функцію надсилання повідомлень до каналу або особисті повідомлення. Повідомляє про зміни пов'язані із користувачем

База даних

Відповідає за збереження та опрацювання даних. Побудова на базі MySQL

Програмне забезпечення

PHP (Fortify, Sanctum)

Реалізований функціонал програми

Middleware

Система являє собою комплекс окремих підсистем для перевірки надходжень даних. Складається із валідатора (перевірка отриманих даних на їх відповідність по типу), middleware - система перевіряє чи має право користувач на зазначену дію, hash/token - функції перевіряють, чи продовжує сесію саме той користувач, що її розпочав.

UI - user interface

diploma.com

Інтерфейс користувача - побудований з метою надати користувачеві зручний інструмент для взаємодії із системою.

Користувач

Система API - запитів

Даний функціонал являє собою більш широке значення, адже, дозволяє не лише використовувати ПЗ повз корнстуацький інтерфейс, а й відкриває інші можливості такі як, інтеграція у інші системи, тестування, тощо

овні позначення

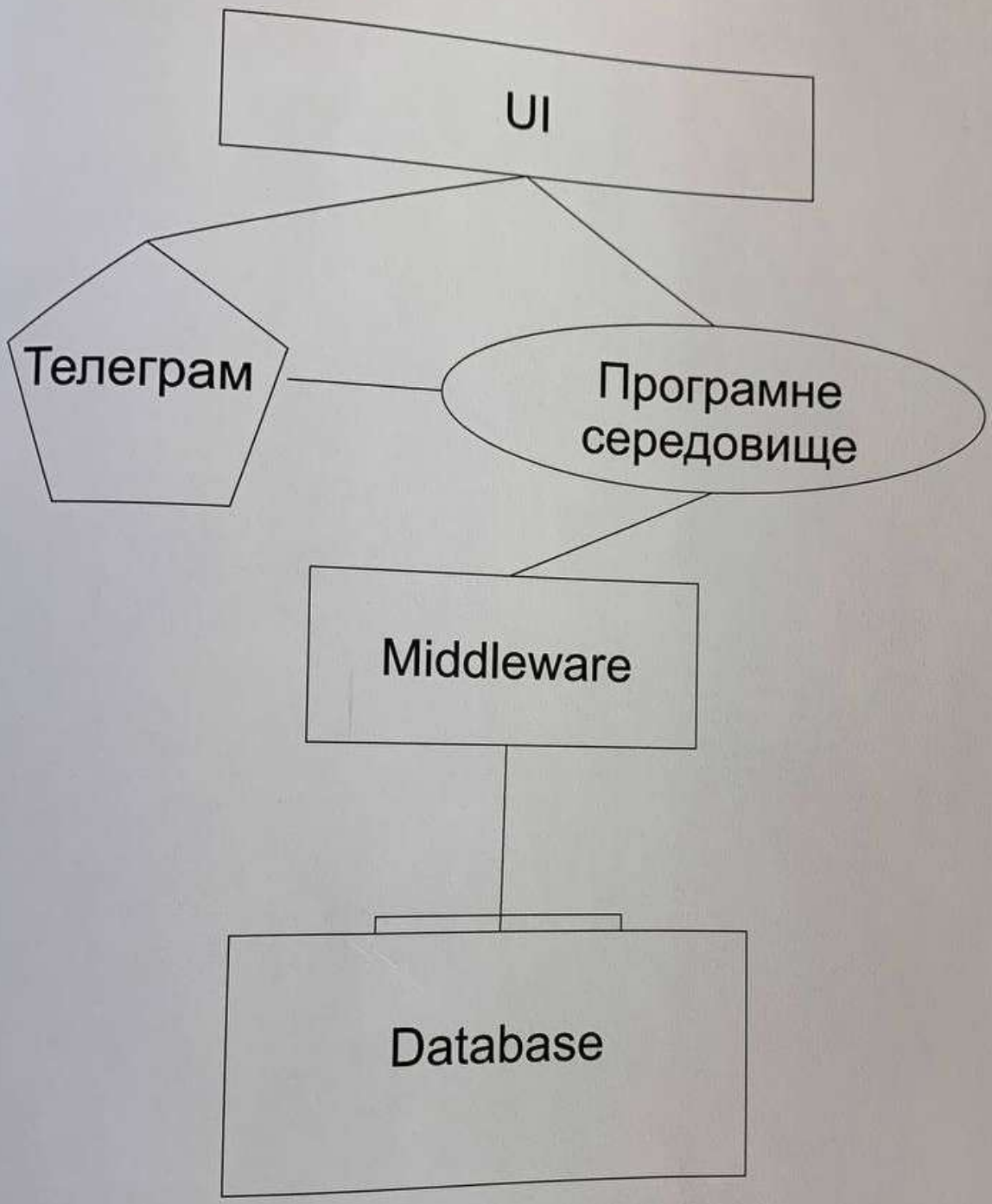
- одностороння постійна взаємодія
- постійна взаємодія
- одноразова взаємодія

Зм.	Арк.	№ докум.	Підпис	Дата	КПКБ. 180131.18.01.07 Е8
Розроб.		Нагребецький О.В.			Повна схема взаємодії та обміну даних
Перевір.		Тітов В. Ю.			
Т.контр.					
Н.контр.		Мостовий С.В.			
Затв.		Кільох Ю. П.			

Літ.	Маса	Масшт.
Н		
Аркуш	Аркушів	1

ХНУ, КБ-18-1

КПКБ. 180131.18.01.07 Е8



					КПКБ. 180131.18.01.07 Е8	
Змн.	Арк.	N докум.	Підпис	Дата	Модель опрацьовування бази даних	Літера: У; Маса; Масштаб
Розроб.		Нагребецький О.В				Аркуш 2; Аркушів
Перевір.		Тітова В. Ю..				
Н. контр.		Мостовий С.В				КБ-18-1
Зав. каф.						
Затв.		Клюоц Ю. П.				

Категорія порушника «ПБ»	Мотиви порушення	Рівень обізнаності щодо ІТС	Можливості за часом дії	Можливості за місцем дії	Сума загроз
Адміністратор	М1	К24	33	М4	12
Власник	М1	К2	32	М1	6
Користувач	М1	К2	32	М2	65
Гість	М1	К1	31	М1	4

Арк.	N докум.	Підпис	Дата
Розроб.	Нагребецький О.В.		
Перевір.	Тітова В.Ю..		
Н. контр.	Мостовий С.В.		
Затв.	Кльоц Ю. П..		

КПКБ. 180131.18.01.07 ТБ

Модель порушника

Літера	Маса	Масштаб
у		

Аркуш 2	Аркушів

КБ-18-1

КПКБ. 180131.18.01.07 Е - 8

Зм.	Арк.	N докум.	Підпис	Дата
Розроб.		Нагребецький О.В		
Перевір.		Тітова В.Ю..		
Н. контр.		Мостовий С.В.		
Т. контр.				
Затв.		Кльоц Ю. П..		

КПКБ. 180131.18.01.07 Е - 8

Схема моделювання хешу

Літера	Маса	Масштаб
у		

Аркуш 2	Аркушів

КБ-18-1

Завідувачу кафедри кібербезпеки
к.т.н., доц.Кльоцу Ю.П.
Нагребецького Олексія Валентиновича
ПІБ здобувача вищої освіти

студента ФІТ, 4 курсу, групи КБ-18-1

ЗАЯВА

З правилами чинного Положення «Про дотримання академічної доброчесності в Хмельницькому національному університеті» від 26.09.2020 (зі змінами від 26.11.2020), згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений (а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіатоповіщений (а) та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів(Unicheck та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

6.06.2022
дата

підпис

UNICHECK

Ім'я користувача:
Кафедра кібербезпеки

Дата перевірки:
02.06.2022 11:53:40 EEST

Дата звіту:
02.06.2022 12:10:21 EEST

ID перевірки:
1011428843

Тип перевірки:
Doc vs Internet + Library

ID користувача:
100008300

Назва документа: НАГРЕБЕЦЬКИЙ_КБ-18-1

Кількість сторінок: 68 Кількість слів: 10541 Кількість символів: 84053 Розмір файлу: 1.66 MB ID файлу: 1011309100

14.4%
Схожість

Найбільша схожість: 2.67% з Інтернет-джерелом (https://wikizero.com/uk/%D0%A1%D0%A3%D0%91%D0%94)

13.8% Джерела з Інтернету | 230 ..... Сторінка 70

2.43% Джерела з Бібліотеки | 134 ..... Сторінка 73

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи | 13

Anti-Plagiarism v-15.257

Максимальное совпадение с одним документом 13.0%

Словари проверки: en_US, ru_RU, ua_UA. **Ошибок в документах: 11%**

	Документ		Суммарное совпадение по Базе Данных	
ID: 104371 Название: Децентралізована захищена база даних з використанням криптографічного шифрування Добавлено в БД: 2022-06-02 Авторы: Нагребецький Олексій Валентинович Руководители: Тітова В.Ю. Консультанты: Опоненты:	Символы	Лексемы	Символы	Лексемы
	68729	575	10925 (16%)	109 (19%)

Источник плагиата

ID	Описание	Наличие плагиата в документе	
		Символы	Лексемы
98715	Название: Проєктування комплексної системи захисту ТОВ "Хмельницький – пожежне спостереження" Добавлено в БД: 2021-12-10 Авторы: Нагребецький О . В. Руководители: Орленко В.С. Консультанты: Опоненты:	8607 (13.0%)	80 (14.0%)

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ

КАФЕДРИ КІБЕРБЕЗПЕКИ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Побудова децентралізованої захищеної бази даних з використанням криптографічного шифрування

Автор: Нагребецький Олексій Валентинович

Спеціальність: 125 – Кібербезпека

Науковий керівник: Тітова Вєра Юріївна, к.т.н., доцент

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом (далі – зазначаються підстави віднесення запозичень до правомірних). Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи (далі – зазначаються детальні та аргументовані підстави віднесення запозичень до правомірних). Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи.	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
5	Інше:	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) Переважна більшість посилань на плагіат прийшлось на текстове наповнення рамок, яке є стандартним у відповідності до ДСТУ;
2) усі інші запозичення фрагментарні, є загально відомою інформацією або мають належним чином оформленні посилання.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості, складає _____%, що, з урахуванням наведених обгрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ В.Ю. Тітова

Гарант ОПП _____ В.М. Чешун

Завідувач кафедри Кб _____ Ю.П. Кльоц

Дата: 09.06.2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

освітньо-кваліфікаційного рівня «бакалавр»

Баклавр Нагребецький Олексій Валентинович

Тема Побудова децентралізованої захищеної бази даних з використанням криптографічного шифрування

Спеціальність 125 – Кібербезпека

Обсяг кваліфікаційної роботи освітньо-кваліфікаційного рівня «бакалавр»: кількість листів креслень 3; кількість сторінок записки 67

1. Короткий зміст КР та прийнятих рішень Дана кваліфікаційна робота присвячена побудові децентралізованої бази даних, захищеної з використанням криптографічного шифрування.

2. Висновок про відповідність КР завданню Кваліфікаційна робота у повній мірі відповідає поставленому завданню як в теоретичній, так і в практичній частині роботи

3. Характеристика виконання кожного розділу роботи, ступінь використання останніх досягнень науки і техніки і передових методів роботи: У вступі подана загальна характеристика поставленої задачі, сформульована актуальність. Визначені задачі, які необхідно вирішити для досягнення поставленої мети, практична цінність отриманих результатів. У першому розділі проведено огляд існуючих методів розподілу рівнів доступу до інформації згідно існуючих стандартів. В другому розділі виконано предпроектний аналіз необхідних складових для проекту та їх облаштування. В третьому розділі реалізовано синтез комплексної системи захисту інформації бази даних з використанням криптографічного шифрування. Четвертий розділ присвячено тестуванню розробленого програмного забезпечення.

4. Позитивні сторони проекту Кваліфікаційна робота має практичну цінність, яка полягає у створенні захищеної бази даних із використанням алгоритмів шифрування, методів зовнішньої взаємодії із системою. В якості середовища розробки програмного продукту використано PHP Storm та мову PHP v.7.4. За допомогою цих засобів було розроблено програмне забезпечення захищеної бази даних.

5. Негативні сторони проекту немає.

6. Оцінка графічного оформлення та пояснювальної записки роботи Графічне оформлення виконане відповідно до теми кваліфікаційної роботи з дотриманням стандартів. В загальному графічне оформлення виконане якісно, пояснювальна записка відповідає нормам щодо її оформлення.

7. Відгук про роботу в цілому В загальному кваліфікаційна робота заслуговує позитивної оцінки. Весь матеріал роботи структурований, чіткий та послідовний. Усі розділи роботи послідовні та логічні, що дозволяє чітко розуміти викладений матеріал в рамках тематики кваліфікаційної роботи. Графічний матеріал дозволяє наочно побачити доцільність та ефективність рішень, які були прийняті за основу для досягнення поставленої мети.

8. Інші зауваження немає

9. Оцінка кваліфікаційної роботи Враховуючи всі позитивні та негативні сторони представленої кваліфікаційної роботи, можна зробити висновок, що вона заслуговує оцінку «відмінно».

РЕЦЕНЗЕНТ (прізвище, ім'я, по батькові, посада, місце роботи) д.т.н., професор, професор кафедри комп'ютерної інженерії та інформаційних систем Савенко Олег Станіславович

« _____ » _____ 2022. _____ (підпис)