Міністерство освіти і науки України

«Київський національний університет імені Тараса Шевченка»

Факультет інформаційних технологій

Кафедра кібербезпеки та захисту інформації

ДОПУСТИТИ ДО ЗАХИСТУ:
завідувач кафедри
кібербезпеки
та захисту інформації
_____ Н.В. Лукова-Чуйко
« » червня 2021р.

ПОЯСНЮВАЛЬНА ЗАПИСКА
дипломної роботи
бакалавра .
(назва освітнього рівня)

галузь знань 12 Інформаційні технології
(шифр і назва галузі знань)

спеціальність 125 Кібербезпека
(код і назва спеціальності)

освітня програма Кібербезпека
(назва освітньої програми)

на тему: «Розробка програмного забезпечення «Under Lock & Key. Менеджер паролів»

Виконавець: студентка IV курсу, групи КБ-41

Галась Марія Ігорівна
(підпис) (прізвище ім'я по-батькові)

	Прізвище, ініціали	**Підпис**
Керівник	Зюбіна Р. В.	

Нормоконтроль	Даков С. Ю.	

Київ 2021

Міністерство освіти і науки України

«Київський національний університет імені Тараса Шевченка»

Факультет інформаційних технологій

Кафедра кібербезпеки та захисту інформації

ЗАТВЕРДЖЕНО:
завідувач кафедри
кібербезпеки
та захисту інформації
_____ Н.В. Лукова-Чуйко
«10» жовтня 2020 р.

ЗАВДАННЯ
на виконання дипломної роботи

спеціальності 125 Кібербезпека
(код і назва спеціальності)

освітньої програми Кібербезпека
(назва освітньої програми)

Студентці **КБ-41** (група) **Галась Марія Ігорівна** (прізвище ім'я по-батькові)

Тема дипломної роботи Розробка програмного забезпечення «Under Lock & Key. Менеджер паролів

1. ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Тема дипломної роботи затверджена на засіданні кафедри кібербезпеки та захисту інформації протокол №2 від 08.10.2020 р.

2. ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Менеджери з шифрування паролів, методи шифрування

3. ЗМІСТ РОЗРАХУНКОВО-ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ

Провести аналіз предметної галузі та об'єкта дослідження; проаналізувати літературні джерела та практичний досвід використання ІС і технологій в предметній галузі; проаналізувати та визначити специфікацію вимог до інформаційної системи/підсистеми; розробити модель інформаційної системи та інформаційного забезпечення.

4. ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Практична цінність Розробка менеджеру паролів та його практична реалізація

5. ДАТА ВИДАЧІ ЗАВДАННЯ

Дата видачі завдання:12 жовтня 2020 року

Завдання видав _____ Р. В. Зюбіна.
(підпис) (ініціали, прізвище)

Завдання прийняла до виконання _____ М. І. Галась
(підпис) (ініціали, прізвище)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Найменування етапів робіт	Строки виконання робіт (початок-кінець)	Відмітка про виконання
1	Уточнення постановки задачі	25.01.2021 – 30.01.2021	*виконано*
2	Аналіз літератури	01.02.2021 – 20.02.2021	*виконано*
3	Обґрунтування вибору рішення	12.02.2021 – 15.02.2021	*виконано*
4	Порівняння наявний програмних систем зберігання паролів	16.02.2021 – 20.02.2021	*виконано*
5	Визначення вимог і моделювання інформаційної системи	22.02.2021 – 21.03.2021	*виконано*
6	Проєктування та реалізація компонентів системи	22.03.2021 – 30.04.2021	*виконано*
7	Результати реалізації інформаційної системи	31.04.2021 – 10.05.2021	*виконано*
8	Оформлення пояснювальної записки	18.05.2021 – 08.06.2021	*виконано*
9	Підготовка до захисту дипломної роботи	09.05.2021 – 21.06.2021	*виконано*

Завдання видав _____ Р. В. Зюбіна
(підпис) (ініціали, прізвище)

Завдання прийняв до виконання _____ М. І. Галась
(підпис) (ініціали, прізвище)

Термін подання дипломної роботи до ЕК 08 червня 2021 року

РЕФЕРАТ

Дипломна робота складається зі вступу, трьох розділів, загальних висновків, списку використаних джерел, додатків, має 66 сторінки основного тексту, 3 таблиці та 18 рисунків. Список використаних джерел містить 30 найменувань і займає 3 сторінки.

У сучасному суспільстві провідним ресурсом є інформація. Інформацією володіють і використовують її всі люди без винятку. На сьогоднішній день користувачі використовують десятки інформаційних сервісів - починаючи з соціальних мереж і електронної пошти, закінчуючи державними порталами надання послуг та інформації. Переважна більшість сервісів потребує автентифікації кожного користувача. Найпростіший спосіб авторизувати користувача - логін і пароль. Паролі традиційно є найбільш поширеним і доступним засобом автентифікації. Внаслідок цього, з'являється проблема - потрібно запам'ятати десятки комбінацій логіна і пароля. Найбільш безпечний і зручний інструмент для виконання цього завдання - зберігати інформацію в менеджері паролів. Менеджер паролів є програмою, яка зберігає паролі або іншу вашу секретну інформацію в зашифрованому вигляді.

Метою даної роботи є розробка менеджеру паролів та його практична реалізація.

Об’єктом дослідження є процес зберігання паролів в менеджері.

Предмет дослідження – інформаційний менеджер паролів.

У роботі проаналізовано аналіз предметної галузі та об’єкта дослідження, а саме існуюча література з шифрування паролів, виконаний аналіз програмних продуктів зберігання паролів, порівняння, вивчення та узагальнення вітчизняної і зарубіжної практики з теми зберігання та шифрування паролів.

Також проаналізовано та визначено специфікацію вимог до інформаційної системи/підсистеми. Розроблено модель інформаційної системи та інформаційного

забезпечення та реалізоване технічне та програмне забезпечення інформаційної системи.

Результатом роботи є розробка програмного забезпечення «Under Lock & Key. Менеджер паролів.

Розроблений лістинг програми, яка зберігає паролі або іншу секретну інформацію в зашифрованому вигляді

Методи дослідження. Процес розробки ІТ буде поділений на етапи. На першому етапі роботи буде виконаний аналіз предметної області аналогічних менеджерів паролів, за допомогою методів аналізу та синтезу створена модель даних, що буде містити визначені дані для менеджера паролів. На другому етапі роботи методами моделювання створюватиметься проєкт інформаційної системи, що реалізуватиме запропоновану ІТ.

Практичне значення одержаних результатів. За результатами дослідження було розроблено менеджер, який дозволяє користувачам зберігати паролі від своїх сторінок.

Ключові слова: Захист персональних даних, зашифровані дані, програма для зберігання паролів, шифрування, інформаційна система, менеджер паролів.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

БД	-	база даних
ДПД	-	Діаграма потоків даних
ІС	-	інформаційна система
ІТ	-	інформаційна технологія
ІКТ	-	Інформаційно-комунікаційні технології
ОС	-	операційна система
ПК	-	персональний комп'ютер
ПЗ	-	Програмне забезпечення
СУБД	-	система управління базою даних
ЦОД	-	Центр обробки даних
AES	-	Advanced Encryption Standard
API	-	Application Programming Interface
(D)DoS	-	(Distributed) Denial-of-Service
IEEE	-	Institute of Electrical and Electronics Engineers
IoT	-	Internet-of-Things
IPS	-	Intrusion Prevention System
IaaS	-	Infrastructure as a service
IT	-	Information Technology
PaaS	-	Platform as a service
SSL	-	Secure Sockets Layer
SaaS	-	Software as a service
TLS	-	Transport Layer Security
VPN	-	Virtual Private Network

ЗМІСТ

ВСТУП

Актуальність теми дослідження. У сучасному суспільстві провідним ресурсом є інформація. Недарма кажуть, що той, хто володіє інформацією, той володіє світом - вона дорого коштує, а володіючи нею, можна ефективно і оптимально вибудовувати будь-яку діяльність.

Інформацією володіють і використовують її всі люди без винятку. Кожна людина вирішує для себе, яку інформацію їй необхідно отримати, яка інформація не повинна бути доступна іншим і т. д..

За останні кілька років багато видів інформації і діяльності людини перейшли в цифровий формат - практично будь-яку послугу можна отримати маючи в своєму арсеналі персональний комп'ютер і доступ в інтернет. А в деяких випадках, то, що раніше можна було отримати, звернувшись безпосередньо в організацію, тепер можливо лише за допомогою онлайн-доступу.

На сьогодні користувачі використовують десятки інформаційних сервісів - починаючи з соціальних мереж і електронної пошти, закінчуючи державними порталами надання послуг та інформації.

Переважна більшість сервісів потребує автентифікації кожного користувача. Найпростіший спосіб авторизувати користувача - логін і пароль.

Паролі традиційно є найбільш поширеним і доступним засобом автентифікації. Для користувачів це один з найбільш зручних варіантів, який, до того ж не вимагає наявності будь-якого додаткового обладнання або спеціальних навичок.

Внаслідок цього, з'являється проблема - потрібно запам'ятати десятки комбінацій логіна і пароля. Крім того, з точки зору безпеки паролі повинні бути довгими і складаються з цифр і символів в різних регістрах. Такі паролі іменуються стійкими, і запам'ятовувати їх об'єктивно складно, тому часто використовуються прості паролі, які легко зламати і отримати доступ до закритої інформації. Для того щоб краще зберегти дані рекомендується міняти паролі раз в 3-6 місяців. Тому, для

кращої безпеки даних та для кожного сервісу постійно рекомендується використовувати різні дані автентифікації і стійкі паролі.

Поступово це стає нездійсненним завданням, і користувач змушений періодично відновлювати забуті комбінації, або використовувати небезпечні методи зберігання (наприклад, браузер зі збереженням логіна і пароля).

Найбільш безпечний і зручний інструмент для виконання цього завдання - зберігати інформацію в менеджері паролів. Менеджер паролів є програмою, яка зберігає паролі або іншу секретну інформацію в зашифрованому вигляді

Мета та завдання наукової роботи. Метою роботи є розробка менеджеру паролів та його практична реалізація.

Для досягнення поставленої мети, необхідно розв'язати такі задачі:

1. провести аналіз предметної галузі та об’єкта дослідження;
2. проаналізувати літературні джерела та практичний досвід використання ІС і технологій в предметній галузі;
3. проаналізувати та визначити специфікацію вимог до інформаційної системи/підсистеми;
4. розробити модель інформаційної системи та інформаційного забезпечення;
5. розробити та реалізувати технічне та програмне забезпечення інформаційної системи.

Об’єктом дослідження є процес зберігання паролів в менеджері.

Предмет дослідження – інформаційний менеджер паролів.

Методи дослідження. Процес розробки ІТ буде поділений на етапи. На першому етапі роботи буде виконаний аналіз предметної області аналогічних менеджерів паролів, за допомогою методів аналізу та синтезу створена модель даних, що буде містити визначені дані для менеджера паролів. На другому етапі роботи методами моделювання створюватиметься проєкт інформаційної системи, що реалізуватиме запропоновану ІТ.

Практичне значення одержаних результатів. За результатами дослідження було розроблено менеджер, який дозволяє користувачам зберігати паролі від своїх сторінок.

РОЗДІЛ 1
ХАРАКТЕРИСТИКА ТА АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1. Програмна система зберігання паролів

Програмна система зберігання паролів (далі менеджер паролів) - програмне забезпечення, яке допомагає користувачеві працювати з паролями і PIN-кодами. У подібного програмного забезпечення зазвичай є місцева база даних або файли, які містять зашифровані дані пароля [2].

Багато менеджерів паролів також працюють як заповнювач форми, тобто вони заповнюють поле користувач і дані пароля автоматично в формах. Зазвичай вони реалізовані як розширення браузера.

Менеджери паролів діляться на три основні категорії:

Локально встановлені програмні програми - зберігають паролі до програмного забезпечення, встановленого на жорсткому диску комп'ютера.

Портативні - зберігають паролі до програмного забезпечення на мобільних пристроях, таких як КПК, смартфон або до портативних додатків на USB флеш-накопичувачі [7].

Мережеві - менеджери паролів онлайн, де паролі збережені на вебсайтах провайдерів [1].

Менеджери паролів можуть також використовуватися як захист від фішінга (вид інтернет-шахрайства, метою якого є отримання доступу до конфіденційних даних користувачів - логінів і паролів) [6].

На відміну від людей, програма менеджер паролів може звертатися з автоматизованим скриптом логічно не вразливу до візуальних імітацій, які схожі на вебсайт [3]. З цією вбудованою перевагою використання менеджера паролів вигідно, навіть якщо у користувача є всього кілька паролів, які він пам'ятає.

Менеджери паролів зазвичай використовують обраний користувачем основний пароль, або секретну фразу, щоб сформувати ключ, який

використовується для шифрування збережених паролів. Цей основний пароль повинен бути досить складним, щоб встояти при атаках зловмисників.

Якщо основний пароль буде зламаний, то будуть розкриті всі збережені в базі даних програми паролі. Це демонструє зворотний зв'язок між зручністю використання і безпекою: єдиний пароль може бути більш зручний, але якщо він буде зламаний, то поставить під загрозу всі збережені паролі.

Основний пароль може також бути атакований і виявлений при використанні криптоаналізу. Така загроза може бути знижена шляхом використання віртуальної клавіатури.

Деякі менеджери паролів включають генератор паролів. Згенеровані паролі можуть бути відгадані, якщо менеджер пароля не використовує криптографічний безпечний генератор випадкових чисел [10].

Виходячи з вищесказаного, визначено, що: менеджер паролів - це спеціальний тип програм, призначених для зберігання конфіденційних даних користувачів.

Менеджер паролів повинен вирішувати за людину таке важливе питання, як генерування складних і надійних паролів. Це означає, що наш менеджер паролів слід забезпечити генератором паролів - функцією, яка у випадковому порядку формує набір нелогічних символів більшою чи меншою мірою складності, що залежить від заданих користувачем параметрів [6], що допоможе користувачам не сидіти і ламати голову, який би пароль придумати, та так, щоб він і не менше 10-ти символів був, і в ньому були як цифри, так букви. Куди простіше доручити цю справу програмному процесу, який видасть результат миттєво [3].

Безпечний пароль - це складний пароль. Його, в принципі, і можна було б згодом запам'ятати, але, якщо таких паролів багато, все це в голові зберігати не вдасться [12]. Тому оптимальніше всього один раз налагодити роботу зі своїми даними авторизації, а потім підтримувати все це в актуальному стані, що дозволить робити менеджер паролів.

Так само, менеджер паролів буде розроблений кросплатформним, тобто працюючим більш ніж на одній апаратній платформі і / або операційній системі [11].

Типовим прикладом платформ є програмне забезпечення, призначене для роботи в операційних системах Linux, Windows і Mac одночасно.

1.2 Порівняння наявний програмних систем зберігання паролів

Зараз на ринку представлений великий вибір різних менеджерів паролів. Проведемо аналіз найбільш популярних додатків. Для аналізу виділені наступні параметри, які дозволять найбільш повноцінно оцінити функціональність самих менеджерів паролів [8, 12]:

–ціна за доступ до додатка;

–зручність для користувача інтерфейсу (де 5 - зовнішній вигляд приємний і немає перевантаженості, і 1 неприємний зовнішній вигляд і перевантажений інтерфейс);

–підтримка російської мови;

–кросплатформеність;

–генератор паролів;

–можливість експорту та імпорту файлів;

–зміна майстер-пароля;

–перенесення бази даних;

–пошук по записах;

–автоматичне введення даних в браузер.

Дані параметри вже наявних програмних систем дозволять поповнити і остаточно сформувати перелік функцій, необхідних для розробки менеджера паролів.

1.2.1 Додаток KeePass – програма для зберігання паролів

KeePass Password Safe - вільна програма для зберігання паролів, яка поширюється на умовах ліцензії GPL. Програма розроблена Домініком Райхле (нім. Dominik Reichl) для операційної системи Windows. KeePass підтримує алгоритми

Advanced Encryption Standard (AES (256-біт), Rijndael). Програма перекладена більш ніж на 40 мов, включаючи російську [29].

Експорт у формати TXT, HTML, XML і CSV, а також імпорт з безлічі різних форматів (рис. 1.1) [4].

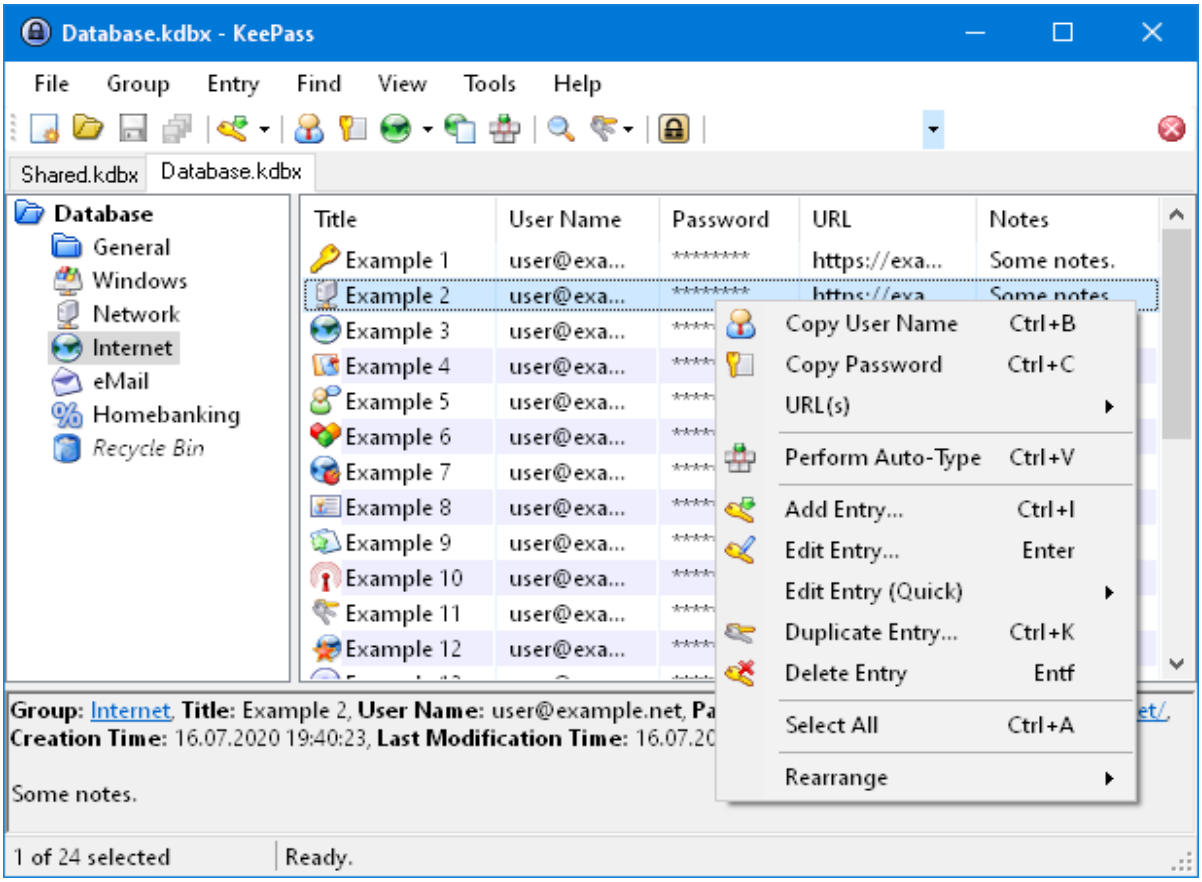

Рисунок 1.1 – Інтерфейс KeePass

У KeePass безліч різних функцій. Однак офіційна реалізація є тільки під Windows.

База даних паролів шифрується з допомогу симетричного AES-256, а майстер-пароль шифрується з SHA-256.

Для синхронізації зазвичай використовують флешкарту, або один з хмарних сервісів, наприклад Dropbox. Деякі мобільні клієнти вміють зі сховищем в Dropbox працювати автоматично [4].

Для KeePass є багато плагінів і додаткових інструментів: утиліти для імпорту / експорту паролів з БД, браузерні плагіни, що дозволяють автоматом заповнювати форми логіна, і додаткові кошти резервного копіювання і синхронізації. Все це зібрано на окремій сторінці.

На цей момент у цього менеджера паролів є очевидний великий мінус: зараз використовується дві версії (1 і 2) бази даних, несумісних один з одним [24]. При цьому є клієнти, що підтримують тільки одну з версій. Так само, всупереч тому, що основний додаток безплатний, існують і платні клієнти, наприклад під i0S [5]. До того ж інтерфейс у деяких клієнтів залишає бажати кращого.

Параметри KeePass представлені в табл. 1.1.

Таблиця 1.1

Параметри KeePass

Ціна за доступ до додатка	Офіційна версія безплатно для Windows, так само є платні версії програми
Зручність для користувача інтерфейсу	3
підтримка російської мови	є
Кросплатформеність	є
генератор паролів	є
Можливість експорту та імпорту файлів	є
Зміна майстер-пароля	є
Перенесення бази даних	є
Пошук по записах	є
Автоматичне введення даних в браузер	є

Крім цього, у KeePass є такі функції як [26]:

– створення запису.

– дублювання записи.

– сортування записів - по стовпцях, за тегами.

– групи записів - дерево і сортування.

– копіювання даних записи - подвійний клік по полю для копіювання, видалення з буфера обміну скопійованої інформації через певний проміжок часу.

– зберігання дат.

– кнопка блокування - при повторному вході програма знову запитує майстер-пароль.

– налаштування бази і програми.

– тригери.

1.2.2 RoboForm – менеджер зберігання паролів

RoboForm - це програма для автоматичного заповнення форм на веб-сайтах і зберігання паролів для операційної системи Windows, яка поширюється на умовах ліцензії Shareware [4].

RoboForm - найстаріший серед менеджерів паролів, серед аналізованих у цій роботі. Перша версія RoboForm була випущена в кінці 1999 року (Рис. 1.2) [5].

Програма підтримує операційні системи Windows і Mac OS X, а так само випускає додатки для мобільних платформ iOS, Android, Windows Mobile, PalmOS і Symbian. При цьому десктопна версія не передбачає імпорту даних з Chrome [4].

Синхронізується з використанням хмарної технології, база зашифрована за стандартом AES-256.

У RoboForm є одна унікальна функція, що дозволяє користувачеві одночасно авторизуватися на декількох сайтах. Дуже зручно, якщо користувач щодня користується декількома сервісами. Також є портативна версія RoboForm, яку можна встановити на флешкарті.

Як і інші менеджери паролів, RoboForm підтримує всі основні браузери і пристрої. Можна вибрати хмарний сервіс для синхронізації на всіх пристроях або зберігати дані локально [21]. Однак в останньому випадку не можна отримати доступ до сховища з інших комп'ютерів і мобільних пристроїв [27].

У корпоративної версії RoboForm можна налаштовувати групові політики, інтеграцію з Active Directory, відновлення майстер-пароля і спільне використання облікових записів.

Підтримується автоматичне створення акаунтів для груп користувачів і облікових записів, обмежених у часі [4]. Один з головних недоліків даного менеджера це його програму з закритим кодом.

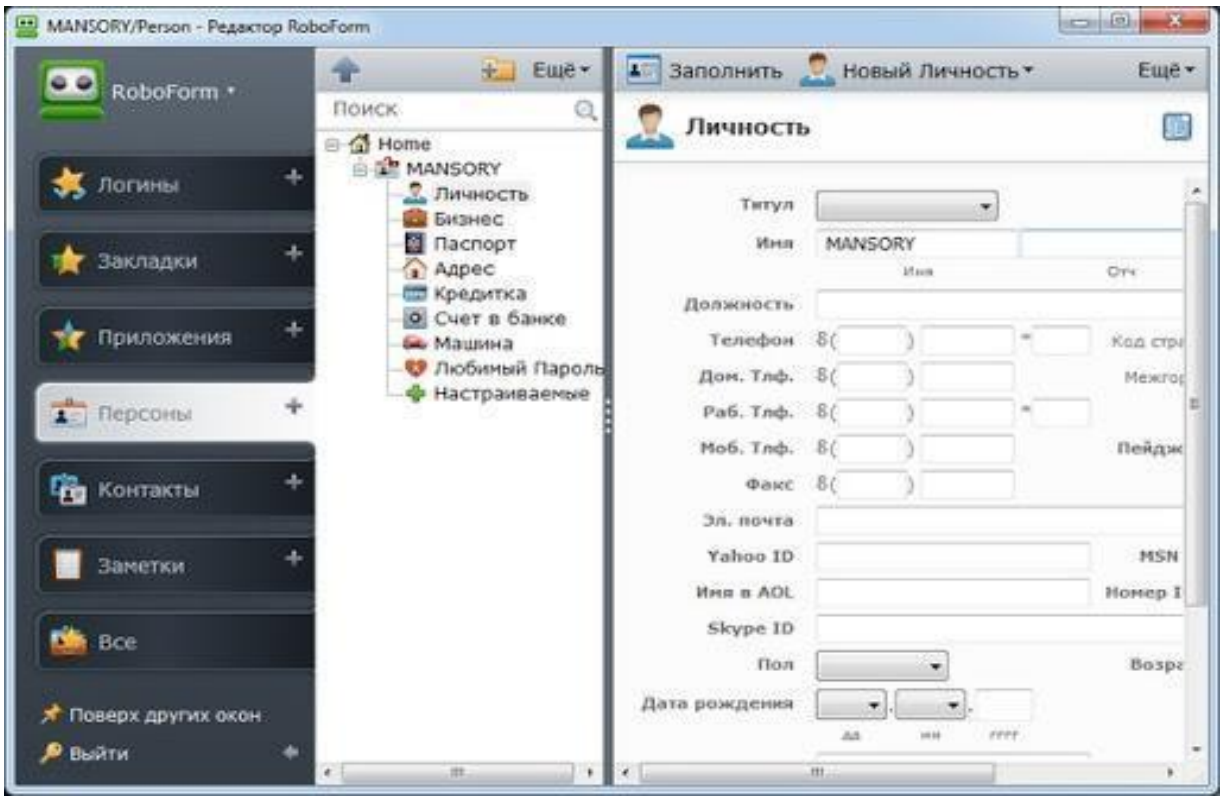

Рисунок 1.2 – Інтерфейс RoboForm

Закритий вихідний код - це особливо важливо для програм, що використовують шифрування даних, через те, що закритість коду не дозволяє оцінити стійкість використовуваних в програмі криптографічних алгоритмів [5].

Параметри RoboForm представлені в табл. 1.2.

Таблиця 1.2

Параметри RoboForm

Ціна за доступ до додатка	перший місяць безплатно, далі 1 рік - 9,95 $
Зручність для користувача інтерфейсу	5
підтримка російської мови	є
Кросплатформеність	є
генератор паролів	є
Можливість експорту та імпорту файлів	тільки імпорт
Зміна майстер-пароля	є
Перенесення бази даних	немає
Пошук по записах	є
Автоматичне введення даних в браузер	є

Крім цього, у RoboForm є такі функції як [21]:

– створення записів.

– друк - логіни, персони, замітки.

– додатковий захист кожного запису.

– для відкриття даних потрібне введення майстер-пароля.

– посилання записів по e-mail - потрібне введення майстер-пароля.

– створення ярликів на робочому столі і в браузері.

– інтеграція з Windows Login.

– можливість відкриття декількох вікон програми.

– адміністрування: створення, редагування груп користувачів, шеринг записів на групи користувачів, синхронізація баз при редагуванні користувачами, перевірка, до яких записів звертався конкретний користувач.

– профілі різних користувачів на одній копії програми.

– керуючі символи Unicode.
– заповнення довгих форм в інтернет-магазинах одним кліком.
– екранна клавіатура.

1.2.3. Менеджер зберігання паролів 1Password

1Password менеджер паролів, що дозволяє використовувати спільні облікові записи (Рис. 1.3) [5]. 1Password підтримує більшість браузерів, автоматично заповнює форми і має версії як для iOS, так і для Android платформ [4].

Розробником даного менеджера паролів є компанія AgileBits.

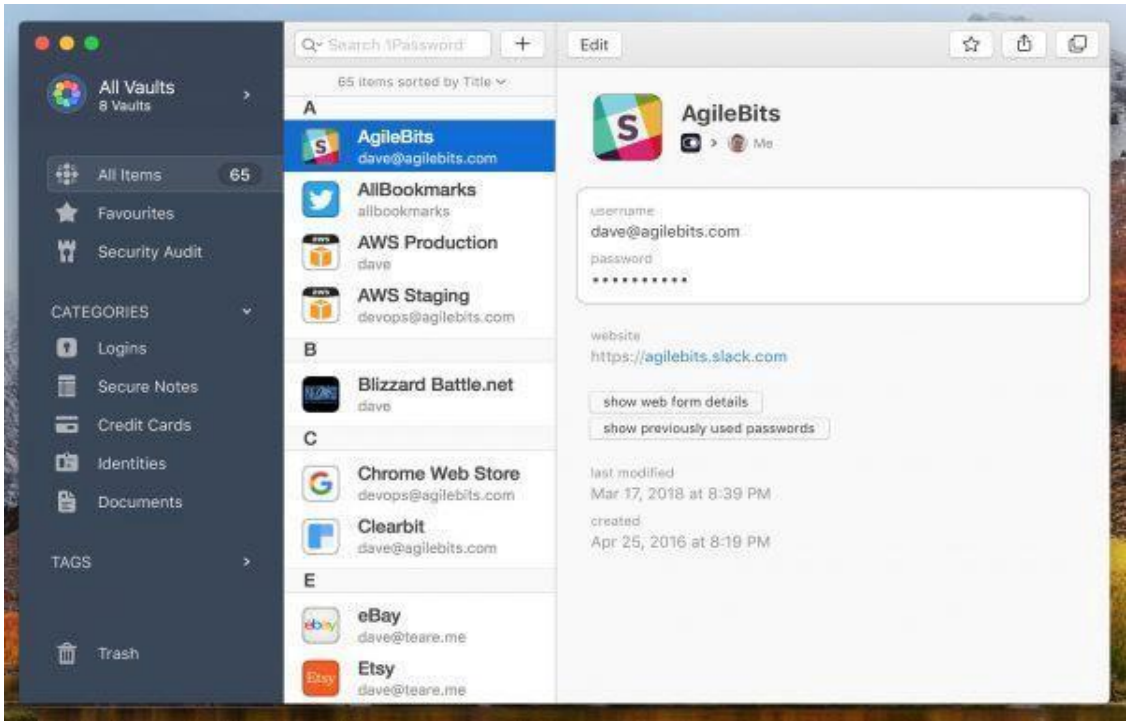

Рисунок 1.3 - Інтерфейс 1Password

1Password один з найпопулярніших менеджерів паролів. Перше, з чим стикаєшся при роботі з ним, - неймовірно продуманий до дрібниць і зручний інтерфейс. При додаванні нового веб-сервісу програма автоматично завантажує його іконку і робить знімок з екрана головної сторінки [21].

В комплекті з додатком також є можливість установки браузерних клієнтів. Вони дозволяють автоматично додавати нові паролі в базу (подібно вбудованим в браузер функцій запам'ятовування паролів), а також автозаповнення форм з логіном [4].

База даних, починаючи з цього року, тепер зашифрована з AES-256. Як синхронізації доступні два варіанти: Dropbox і iCloud. Як в додатку, так і в плагінах є також зручний генератор безпечних паролів [4].

Додатково можна відзначити portable-версію, написану на JS і HTML, яка працює в браузері. Детальніше про неї написано на офіційному сайті програми [4].

Але є і свої нюанси. Наприклад, висока ціна: десктопних програм коштує близько 50 доларів. Під Linux клієнта немає взагалі, а в Android 1Password вміє тільки переглядати паролі, а не редагувати їх або додавати нові.

Параметри 1Password представлені в табл. 1.3.

Таблиця 1.3

Параметри 1Password

Ціна за доступ до додатка	перший місяць безплатно, далі 1 місяць-5 $ або назавжди 64,99$
Зручність для користувача інтерфейсу	5
підтримка російської мови	є
Кросплатформеність	є
генератор паролів	є
Можливість експорту та імпорту файлів	є
Зміна майстер-пароля	є
Перенесення бази даних	є
Пошук по записах	є
Автоматичне введення даних в браузер	є

Крім цього, у 1Password є такі функції як [22]:

- створення записів - швидкі кнопки для створення різних типів.
- навігаційна колонка - все записи, вибраного, категорії, тека.
- вибране.
- модуль для браузерів.
- поділитися доступами: групи, iMessage, e-mai1.
- сек'юріті-аудит - записи зі слабким, старим, дублює паролем.
- очищення буфера обміну через певний проміжок часу.
- кошик.

1.2.4. Dashlane – генератор паролів

У Dashlane привабливий, зручний інтерфейс, який дозволяє легко генерувати надійні паролі і надійно зберігати їх. Після того, як користувач зберігає пароль, Dashlane автоматично заповнює його кожен раз, коли це потрібно, що економить користувачу час (рис. 1.4).

Крім того, можна використовувати Dashlane для зберігання іншої особистої інформації та платіжних реквізитів. Він також дозволяє зберігати замітки, документи, вкладення і іншу важливу інформацію за допомогою функції Secure Notes [15].

Цей менеджер паролів дозволяє ділитися своїми обліковими даними з колегами, сім'єю і друзями. Користувачу не потрібно видавати паролі, щоб забезпечити їм доступ.

Dashlane надає доступ до комплексної панелі ідентифікації, яка оцінює стан безпеки в мережі і дозволяє дізнатися, що користувач можете поліпшити, щоб підвищити рівень безпеки [19].

Він навіть регулярно відстежує глибокий інтернет і відправляє користувачу персоналізовані оповіщення, якщо виявляє крадіжку або витік даних.

Цей менеджер паролів пропонує безплатну версію, яка обмежує користувача одним пристроєм і дозволяє зберігати до 50 паролів [16]. Оновлення до

преміумверсії включає необмежене сховище і кількість підключених пристроїв, а також доступ до функції моніторингу глибокого інтернету.

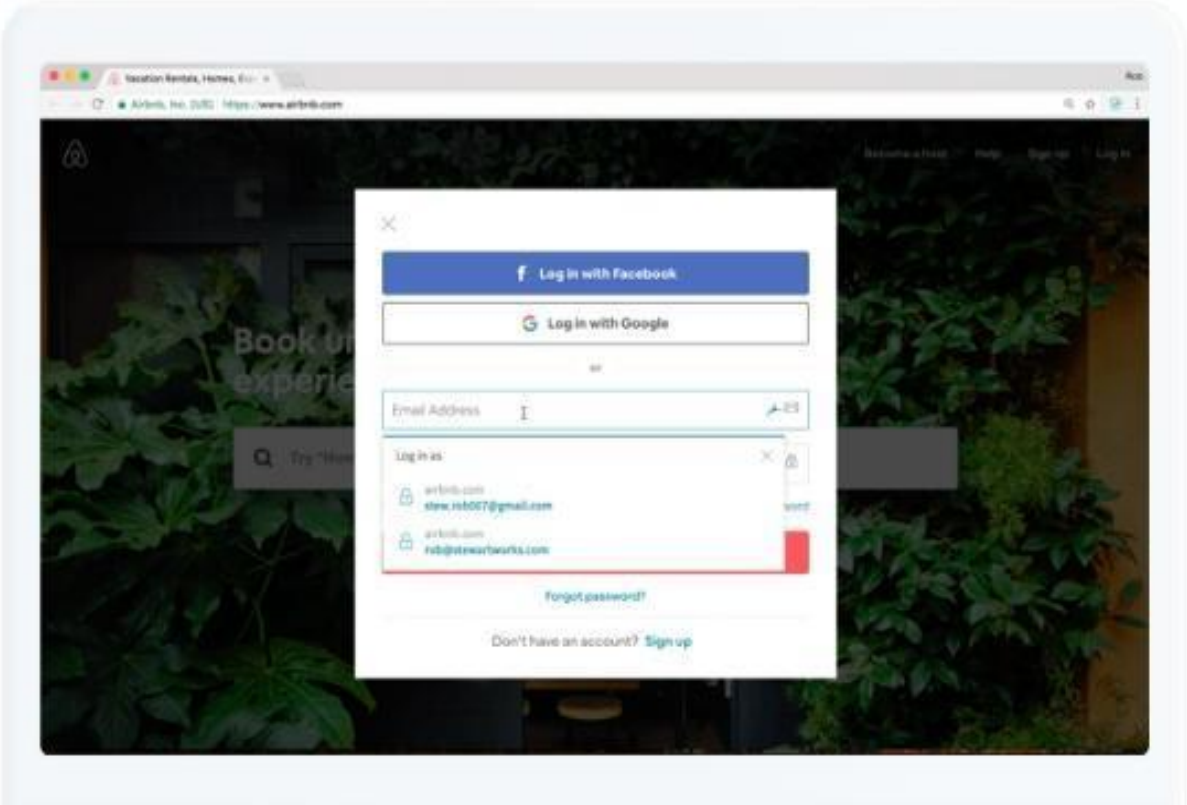

Рисунок 1.4 - Інтерфейс Dashlane

У Dashlane також є план Premium Plus, що забезпечує кредитний моніторинг і страхування від крадіжки особистих даних. Користувач може протестувати преміумплани завдяки 30-денній гарантії повернення грошей Dashlane [20].

Підтримувані платформи: Windows, macOS, Android, iOS і Linux.

Підтримувані браузери: Chrome, Safari, Firefox, Internet Explorer і Edge.

Шифрування: AES 256-бітний ключ з CBC-HMAC і вибором між Argon2d або PBKDF2-SHA2.

1.2.5. Keeper - ефективний менеджер паролів

Keeper - ефективний менеджер паролів, націлений на бізнес, але також добре він працює як персональний менеджер паролів. Програмне забезпечення пройшло

кілька незалежних аудитів, які підтвердили, що Keeper є безпечним і надійним сервісом (рис. 1.5)..

Менеджер використовує пропрієтарну архітектуру безпеки з нульовим розголошенням, щоб гарантувати, що ніхто, включаючи Keeper, не має доступу до приватної інформації, яку користувач зберігає за допомогою диспетчера паролів.

Keeper завжди шифрує дані на пристрої, перш ніж відправити їх в хмарне сховище Keeper. Єдиний спосіб розшифрувати і отримати доступ до даних, що зберігаються в Keeper, це використовувати майстер-пароль, доступ до якого є тільки у користувача [19] (рис. 1.5).

Якщо користувачу потрібен додатковий рівень безпеки, то можна скористатися багатофакторної аутентифікації Keeper.

Рисунок 1.5 - Інтерфейс Keeper

Keeper пропонує традиційну двофакторну автентифікацію, а також біометричний логін і ДНК Keeper, що дозволяє підтвердити особистість користувача за допомогою Apple Watch або пристрою Android Wear. Користувач навіть може

використовувати відбиток пальця, щоб отримати доступ до сховища Keeper за допомогою Touch ID на iPhone або iPad.

Keeper пропонує обмежений безплатний тарифний план, який дозволяє використовувати менеджер паролів на одному мобільному пристрої з місткістю сховища даних 100 МБ.

Підтримувані платформи: Windows, macOS, Android, iOS і Linux.

Підтримувані браузери: Chrome, Firefox, Safari, Internet Explorer, Microsoft Edge і Opera [20].

Шифрування: AES 256-біт з PBKDF2-SHA2.

1.2.6. Програмне забезпечення LastPass – генератор зберігання паролів

LastPass обіцяє спростити життя користувача і це саме те, що він робить. Добре розроблене програмне забезпечення працює у фоновому режимі вашого пристрою або браузера і захищаючи, зберігаючи і автоматично заповнюючи реєстраційні дані (рис. 1.6).

У LastPass є вбудований генератор логінів і паролів, який допоможе користувачу швидко створити супербезпечні облікові дані для всіх ваших облікових записів.

Користувач отримує всі переваги безпеки ідеально захищених паролів без необхідності їх запам'ятовувати.

Конфіденційність користувача гарантується, через те, що навіть LastPass не може отримати доступ до майстер-паролю або ключів шифрування.

Всі дані шифруються і розшифровуються на рівні пристрою.

Його функція автоматичного заповнення не тільки заповнить дані вашого пароля для облікових записів, які користувач додає до своїх, але також може заповнити такі дані, як платіжна інформація, коли користувач робить покупки в інтернеті.

Сервіс також може зберігати іншу конфіденційну інформацію, таку як страхові карти і членство в цифровому файлі записів.

Преміум-план містить додаткові функції, такі як безпечний загальний доступ, екстрений доступ і пріоритетна технічна підтримка. Також пропонуються спеціальні сімейні плани [23].

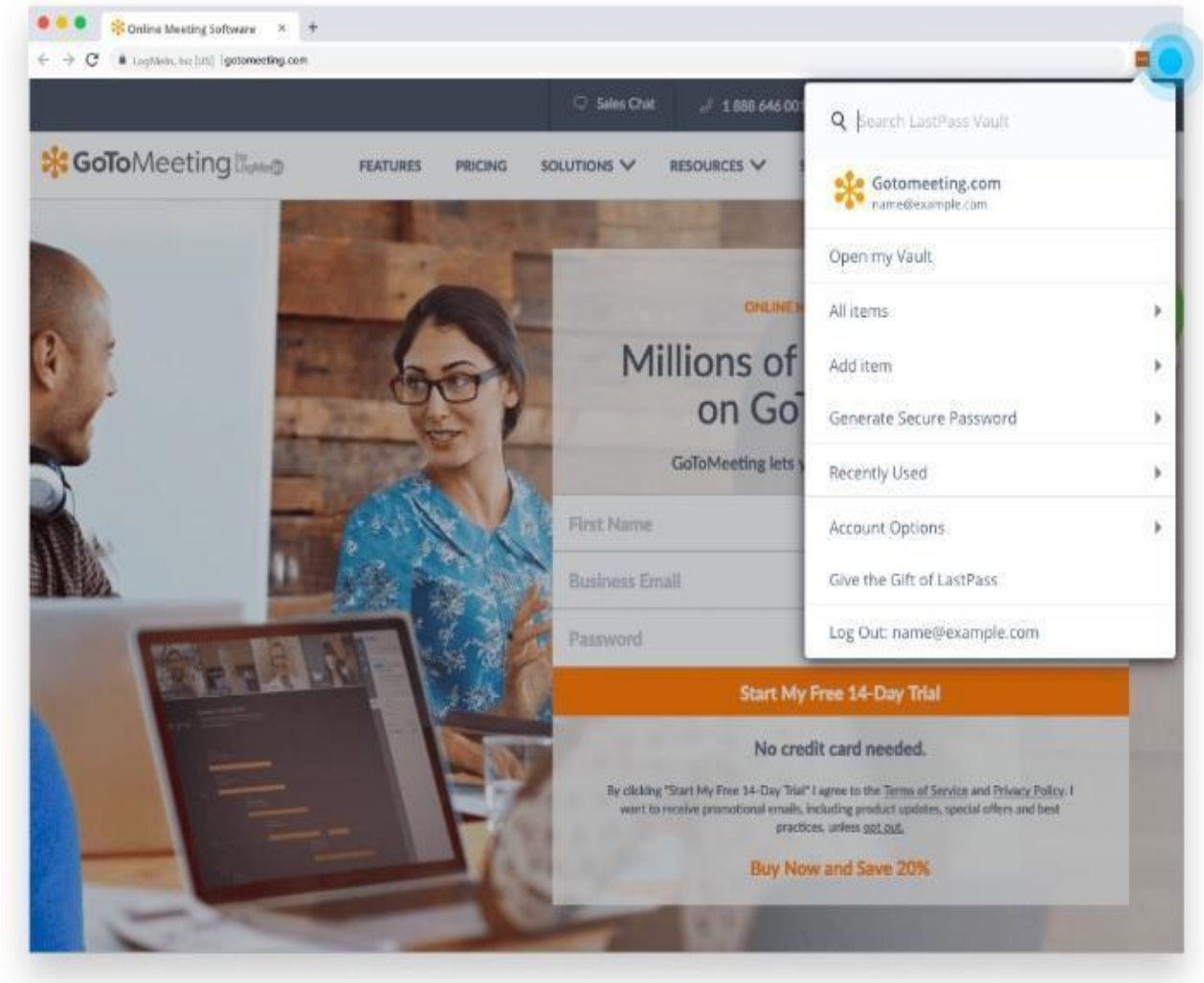

Рисунок 1.6 - Інтерфейс LastPass

Підтримувані платформи: Windows, macOS, iOS, Android і Linux.

Підтримувані браузери: Chrome, Firefox, Safari, Internet Explorer, Opera і Microsoft Edge.

Шифрування: AES 256-біт з PBKDF2 SHA-256.

1.2.7. Сервіс для зберігання паролів RememBear

RememBear - це не просто мила назва. Сервіс використовує найкращі функції безпеки, щоб захистити паролі максимально комфортним для користувача способом.

Щоб переконатися, що користувач єдиний, хто може отримати доступ до майстер-паролю, RememBear використовує безпечний віддалений пароль з протоколом обміну Діффі-Хеллмана, який перевіряє пароль користувача без підключення до мережі [25] (рис. 1.7).

RememBear захищає дані, що зберігаються на його серверах, за допомогою сервісів управління ключами Amazon. Він також додає додатковий рівень шифрування через протокол транспортного рівня, щоб звести до мінімуму його залежність від HTTPS.

Цей менеджер паролів продемонстрував свою прозорість під час незалежного аудиту фахівцем з кібербезпеки Cure53, який підтвердив відсутність критичних проблем в безпеці RememBear (рис. 1.7).

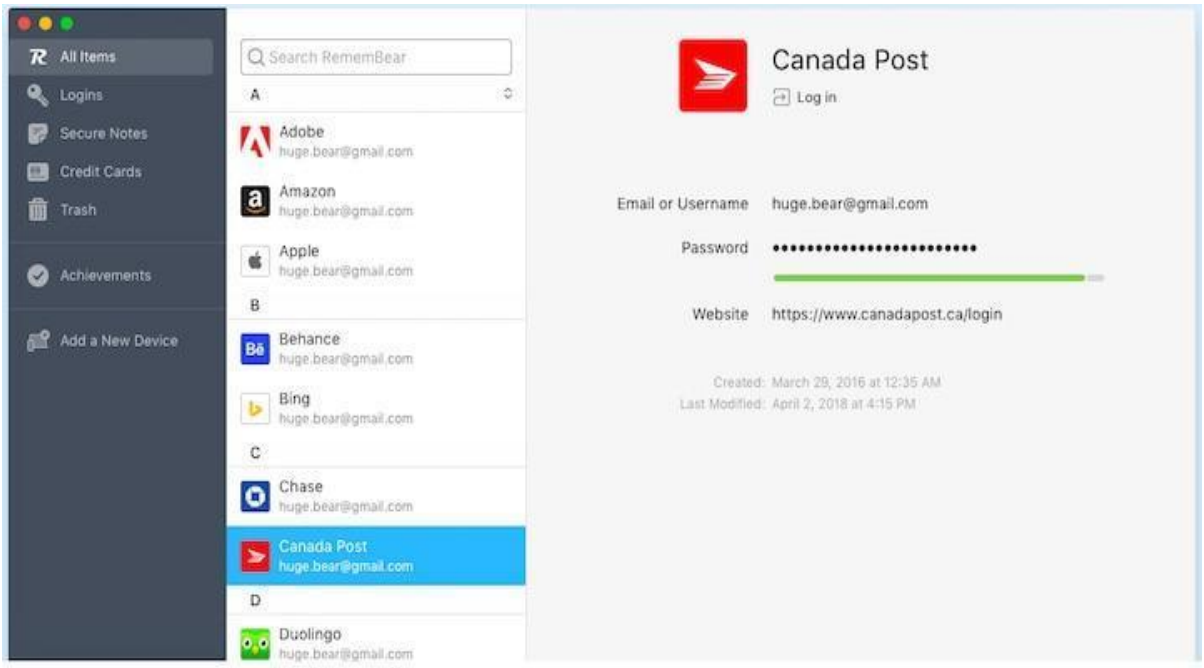

Рисунок 1.7 - Інтерфейс RememBear

RememBear пропонує безкоштовну версію, яку користувач можете використовувати на одному пристрої, і преміум-версію, яка буде працювати на всіх ваших пристроях.

Підтримувані платформи: Windows, macOS, Linux, iOS і Android.

Підтримувані браузери: Chrome, Firefox і Safari.

Шифрування: 256-бітове AES з PBKDF2 SHA256.

1.2.8 Додаток Intuitive Password

Intuitive Password серйозно ставиться до безпеки користувача. Він зберігає всі дані за шарами брандмауерів, а доступ до його баз даних строго контролюється сертифікатами додатків. Він також пропонує надійну антивірусну програму, яка регулярно оновлюється (рис. 1.8).

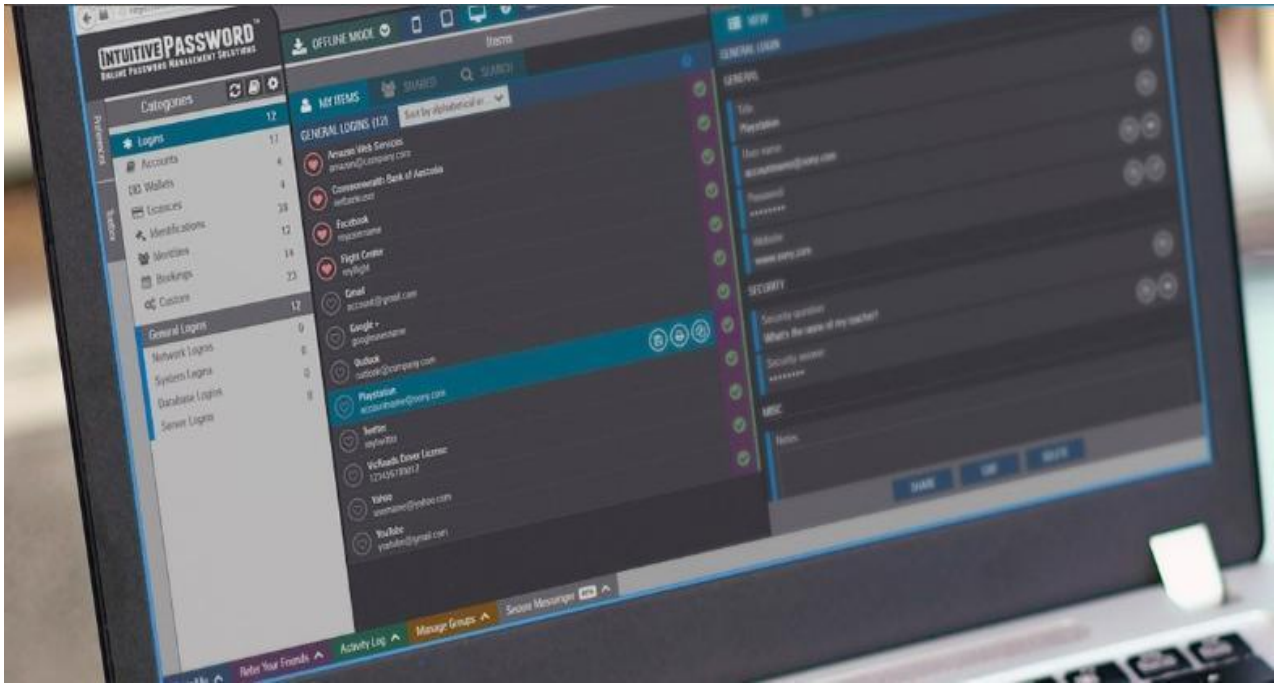

Рисунок 1.8 - Інтерфейс Intuitive Password

Intuitive Password захищає мережу від DDoS-атак і використовує сервіс сканування шкідливих програм в режимі реального часу, щоб забезпечити безпеку своїх серверів в будь-який час.

Intuitive Password не зберігає ваші паролі у вигляді звичайного тексту.

Він використовує PBKDF2 з унікальною сіллю для кожного облікового запису, щоб захистити ваші паролі від хакерів. Він також пропонує двофакторну аутентифікацію з використанням алгоритму TOTP і алгоритму одноразового пароля на основі обміну SMS [30].

Intuitive Password пропонує безплатний і преміум-опції і сумісний з Windows, Android, macOS і iOS.

Підтримувані платформи: Windows, macOS, Linux, iOS і Android.

Підтримувані браузери: Microsoft Edge, Firefox, Safari, Chrome і Opera.

Шифрування: AES 256-біт з PBKDF2.

1.2.9 Сервіс Sticky Password

Sticky Password - ще один надійний вибір. Сервіс пропонує поліпшену двофакторну аутентифікацію і підтримує біометричну аутентифікацію з використанням відбитка пальця на сумісних пристроях (рис. 1.9).

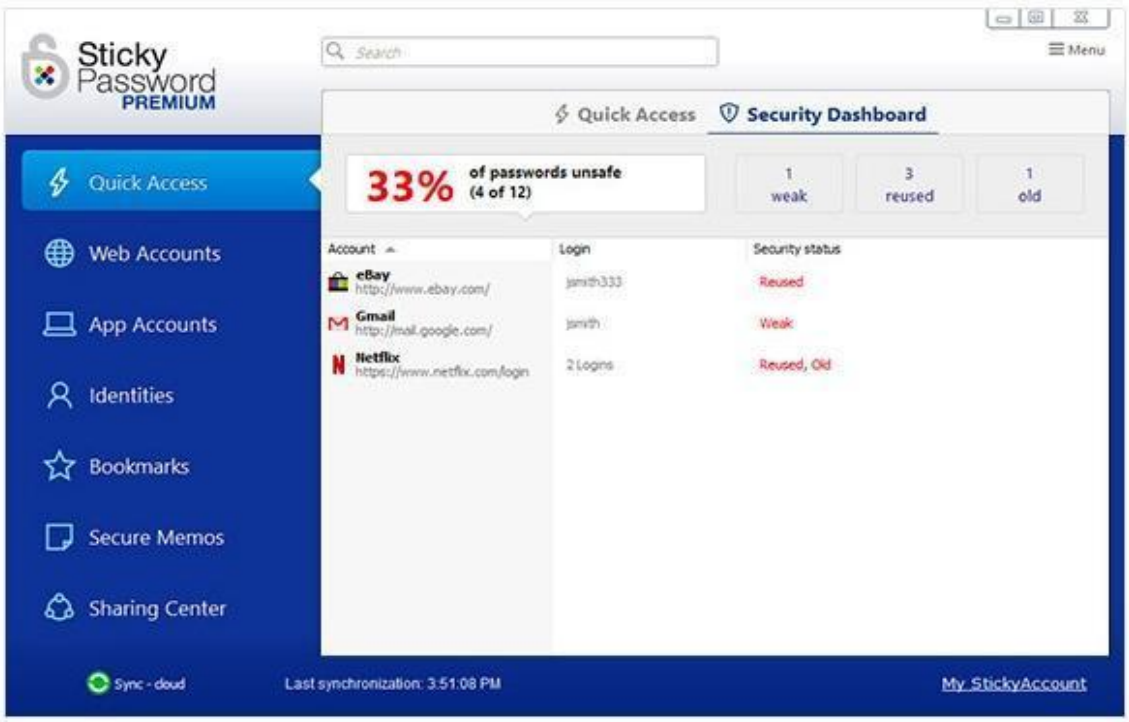

Рисунок 1.9 - Інтерфейс Sticky Password

Ваші паролі будуть автоматично заповнюватися, а майстер-пароль може бути збережений тільки вами і ніколи не з'явиться на серверах або пов'язаних пристроях [28].

В надійному сховищі сервера буде безпечно зберігатися вся інформація про вашу кредитну картку, вона також буде синхронізована з усіма пристроями користувача для легкого вилучення в один клік.

Sticky Password пропонує безплатний і преміумплан, а також 30-денну пробну версію плану преміум. Компанія навіть жертвує на порятунок ламантинів за кожну продану ліцензію.

Підтримувані платформи: Windows, macOS, Linux, iOS і Android.

Підтримувані браузери: Chrome, Firefox, Internet Explorer, Opera, Chromium, Seamonkey, Yandex, Comodo Dragon і Pale Moon.

Шифрування: 256-бітове AES з PBKDF2 SHA256.

1.3. Виділення важливих функцій програмних систем для зберігання паролів

На підставі аналізу існуючих систем, а також на підставі опитування користувачів (в тому числі і потенційних) був складений загальний перелік вимог до розроблюваної програмної системи зберігання паролів:

1) Кросплатформеність - є для нас важливим критерієм, оскільки ця функція дозволяє використовувати програму на різних пристроях [18].

Ви можете бути володарями персональних комп'ютерів і ноутбуків з різними операційними системами, але користуватися з них одними і тими ж інтернет-ресурсами, і вкрай незручно було б використовувати на них різні менеджери паролів - проробляючи одну і ту ж роботу двічі [14];

2) Зручний інтерфейс користувача - є важливим критерієм, тому, що саме інтерфейс дозволяє користувачеві взаємодіяти з програмою.

Важлива зрозумілість для користувача інтерфейсу, легкість навчання роботі з ним, трудомісткість вирішення певних завдань з його допомогою, продуктивність роботи користувача з ПО, частота появи помилок і скарг на незручності [17]. Важливо, щоб інтерфейс був простим і очевидним.

3) Генератор паролів - як уже говорилося раніше, ця функція, яка у випадковому порядку формує набір нелогічних символів більшою чи меншою мірою складності, що залежить від заданих користувачем параметрів, служить для того, щоб допомагати користувачам не сидіти і ламати голову, який би пароль придумати. Це суттєво економить час користувача.

4) Безкоштовне розповсюдження - менеджер паролів розробляється з метою спрощення буднів користувачів Інтернет-ресурсами, а не з метою отримання матеріальної вигоди .[25]

Не знайшовши відповідних за критеріями менеджерів на Linux та iOS ми вирішили зробити свій менеджер. Після проведеного аналізу нами були сформовані вимоги до реалізації менеджеру паролів – наша програмна реалізація буде схожою на Keepass, тому що на нашу думку це один з найкращих менеджерів шифрування

паролів. Дані вимоги і критерії до програмної системи зберігання паролів будуть враховані нами при її розробці.

Висновки до 1 розділу

В розділі було проведено аналіз програмних систем та виділено значні функції програмних систем для зберігання паролів.

Програмна система зберігання паролів (далі менеджер паролів) - програмне забезпечення, яке допомагає користувачеві працювати з паролями і PIN-кодами.

Менеджер паролів повинен вирішувати за людину таке важливе питання, як генерування складних і надійних паролів. Це означає, що наш менеджер паролів слід забезпечити генератором паролів - функцією, яка у випадковому порядку формує набір нелогічних символів більшою чи меншою мірою складності, що залежить від заданих користувачем параметрів, що допоможе користувачам не сидіти і ламати голову, який би пароль придумати, та так, щоб він і не менше 10-ти символів був, і в ньому були як цифри, так букви. Куди простіше доручити цю справу програмному процесу, який видасть результат миттєво [9].

Зараз на ринку представлений великий вибір різних менеджерів паролів. Найбільш популярними додатками є: KeePass, RoboForm, 1Password, Dashlane, Keeper, LastPass, RememBear, Intuitive Password, Sticky Password.

На підставі аналізу теперішніх систем, а також на підставі опитування користувачів (в тому числі і потенційних) був складений загальний перелік вимог до розроблюваної програмної системи зберігання паролів [13]:

- кросплатформеність;
- зручний інтерфейс користувача;
- генератор паролів;
- безплатне розповсюдження.

Не знайшовши відповідних за критеріями менеджерів на Linux та iOS було вирішено розробити менеджер. Після проведеного аналізу були сформовані вимоги до реалізації менеджеру паролів – програмна реалізація буде схожою на Keepass,

тому що це один з найкращих менеджерів шифрування паролів. Дані вимоги і критерії до програмної системи зберігання паролів будуть враховані при її розробці.

РОЗДІЛ 2
ВИЗНАЧЕННЯ ВИМОГ І МОДЕЛЮВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Аналіз вимог до інформаційної системи

1. Найменування і підстава для створення найменування: «Розробка кросплатформеної програмної системи зберігання паролів».

2. Призначення і цілі створення системи

Призначення: Зберігання в зашифрованому вигляді і використання даних користувача для аутентифікації.

Мета створення системи: Дослідження методів захисту інформації в навчальних цілях.

3. Вимоги до структури та функціонування системи

Вимоги до структури: проєкт повинен бути розбитий на кілька частин (по функціональності і виконуваних завдань).

Вимоги до функціонування:

- для дешифрування даних система повинна вимагати «майстер-пароль» і шлях до файлу;
- у програмній системі має бути дерево навігації створеними записами; .
- повинні бути передбачені такі функції як: створення, видалення, редагування записів.

4. Вимога до інтерфейсу системи

- інтерфейс повинен відображати настроювані параметри системи;
- інтерфейс повинен бути розділений на 2 основні частини: дерево навігації та інформація про обрані записи.
- інтерфейс повинен бути інтуїтивно зрозумілим;

5. Вимоги до технічного забезпечення

- базова мова реалізації - C++, C#.
- версія компілятора - version clang 6.0.0 - 1ubuntu2;

– система збірки – Cmake + MakeFiles використовувані в середовищі розробки;
– IDE – Qtcreator.

Найважливішими вимогами до менеджера паролів як сервісу зі зберігання інформаційних даних є:

- підвищення достовірності та персональної відповідальності даних збору та візуалізації;
- автоматизація процесу збору даних;
- функції обробки та збереження масивів даних;
- створення нової інформації за допомогою аналізу та синтезу наявних даних;
- підтримка тематичної настройки вхідних потоків даних під інформаційні потреби методології підтримки прийняття рішень;
- багатокористувацький інтерфейс;
- кросплатформеність.

2.2. Метод шифрування

2.2.1. Шифрування AES-256

Шифрування - один із найпоширеніших способів захисту конфіденційних даних. Шифрування працює, беручи звичайний текст і перетворюючи його в текст шифру, який складається з, здавалося б, випадкових символів. Розшифрувати його може лише той, хто має спеціальний ключ [25].

AES використовує симетричне шифрування ключів, яке передбачає використання лише одного секретного ключа для шифрування та розшифровки інформації.

Розширений стандарт шифрування (AES) - це перший і єдиний загальнодоступний шифр, затверджений Агентством національної безпеки США (NSA) для захисту надсекретної інформації. AES вперше отримав назву Rijndael на честь двох розробників - бельгійських криптографів Вінсента Ріймена та Джоан Дамен.

На рис. 2.1 показано, як працює симетричне шифрування ключів [30]:

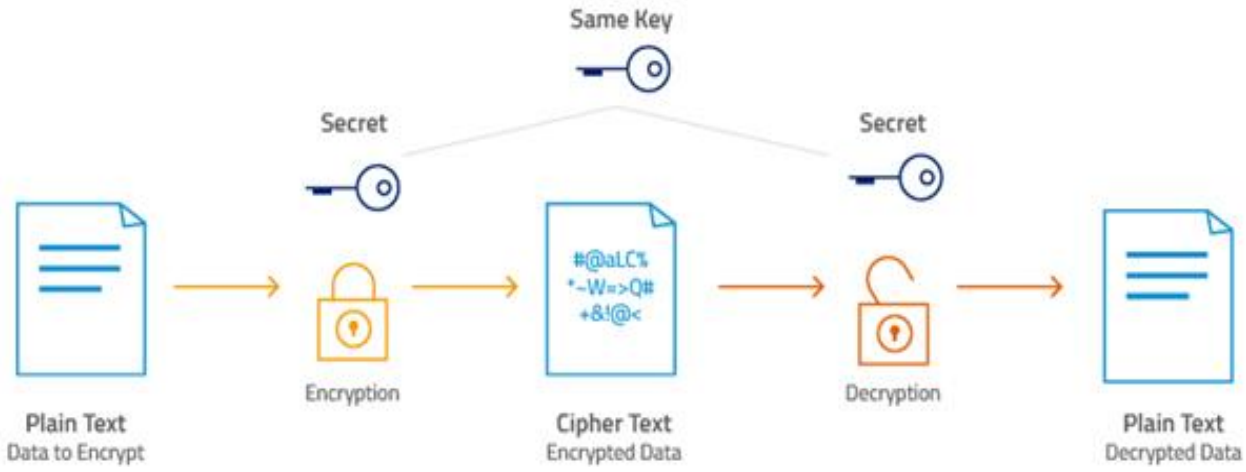

Рисунок 2.1 - Симетричне шифрування ключа

AES-256, який має довжину ключа 256 біт, підтримує найбільший бітовий розмір і практично не ламається за рахунок грубої сили на основі поточної обчислювальної потужності, що робить його найсильнішим стандартом шифрування. Наступна таблиця показує, що можливі комбінації клавіш експоненціально збільшуються із збільшенням розміру ключа.

Сьогодні AES - це довірена система з широким поширенням.

Бібліотеки AES розроблені для мов програмування, включаючи C, C ++, Java, Javascript та Python. AES використовується програмами для стиснення файлів, включаючи 7 Zip, WinZip та RAR; системи шифрування дисків, такі як BitLocker та FileVault; та файлові системи, такі як NTFS. Це важливий інструмент у шифруванні баз даних, а також у системах VPN, таких як IPsec та SSL / TLS.

Менеджери паролів, такі як LastPass, KeePass та 1Password, використовують AES, як і програми обміну повідомленнями, такі як WhatsApp та Facebook Messenger.

Набір інструкцій AES інтегрований у всі процесори Intel і AMD. Навіть відеоігри, такі як Grand Theft Auto IV, використовують AES для захисту від хакерів.

Основний принцип усього шифрування полягає в тому, що кожна одиниця даних замінюється іншою відповідно до ключа безпеки. Більш конкретно, AES був розроблений як мережа заміщення-перестановки. AES забезпечує додаткову безпеку, оскільки використовує процес розширення ключів, в якому початковий ключ використовується для створення серії нових ключів, які називаються круглими клавішами. Ці круглі ключі генеруються під час кількох раундів модифікації, кожен з яких ускладнює розрив шифрування.

Спочатку початковий ключ додається до блоку за допомогою шифру XOR («ексклюзивний»), який є операцією, вбудованою в апаратне забезпечення процесора. Потім кожен байт даних замінюється іншим, слідуючи заздалегідь визначеній таблиці [26].

Далі рядки масиву 4 × 4 зміщуються: байти у другому рядку переміщуються на один пробіл ліворуч, байти в третьому рядку - два пробіли, а байти в четвертому - три. Потім стовпці змішуються - математична операція поєднує чотири байти в кожному стовпці.

Нарешті, круглий ключ додається до блоку (подібно до того, як був початковий ключ), і процес повторюється для кожного раунду. Це дає шифротекст, який кардинально відрізняється від простого тексту. Для дешифрування AES той самий процес здійснюється в зворотному порядку.

Кожен етап алгоритму шифрування AES виконує важливу функцію. Використання іншого ключа для кожного раунду забезпечує набагато складніший результат.

Заміна байтів модифікує дані нелінійно, затемнюючи зв’язок між оригінальним та зашифрованим вмістом. Зміщення рядків та змішування стовпців розсіює дані, транспонуючи байти, щоб ще більше ускладнити шифрування. Зсув дифузує дані по горизонталі, а змішування - вертикально.

Результат - надзвичайно складна форма шифрування.

2.2.2. Хешифрування SHA-256

SHA (Secure Hash Algorithm) - одна з ряду криптографічних хеш-функцій.

Криптографічний хеш - це як підпис для набору даних. Якщо користувач хоче порівняти два набори вихідних даних (джерело файлу, текст або подібне), завжди краще їх хешувати і порівняти значення SHA256. Це як відбитки пальців даних. Навіть якщо змінено лише один символ, алгоритм видасть різне хеш-значення.

SHA256, генерує майже унікальний, фіксований розмір 256 біт (32 байт) хеш. Хеш - це одностороння функція. Це робить його придатним для перевірки цілісності ваших даних, виклику хеш-аутентифікації, захисту від несанкціонованого доступу, цифрових підписів, блокчейну [28] (рис. 2.2).

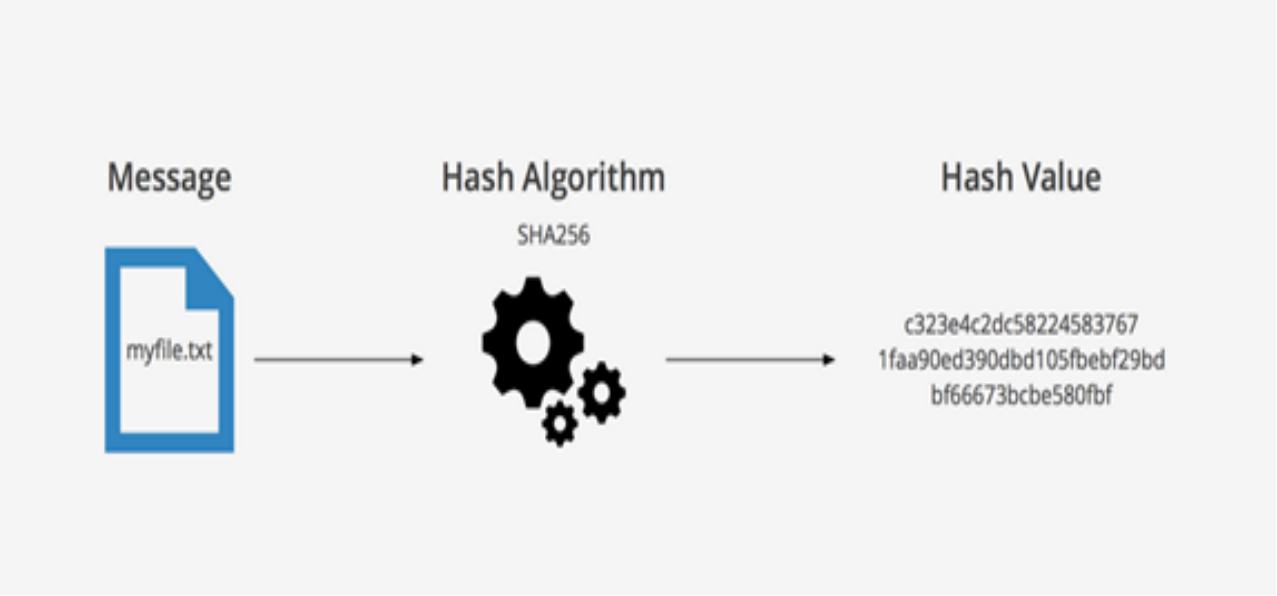

Рисунок 2.2 – Криптографічний сенс SHA-256

Завдяки вдосконаленням найновішого апаратного забезпечення (CPU та GPU) стало можливим розшифрувати алгоритм SHA256 назад. Тому більше не рекомендується використовувати його для захисту паролем або інших подібних випадків використання.

Алгоритм SHA256 все ще можна використовувати, щоб переконатися, що користувач отримав ті самі дані, що й оригінальний. Наприклад, якщо завантажувати щось, користувач можете легко перевірити, чи не змінилися дані через помилки мережі або введення зловмисного програмного забезпечення. Користувач може порівняти хеші файлу з оригінальним, який зазвичай надається на веб-сайті, з якого ви отримуєте дані або файл.

SHA-256 є однією з наступних хеш-функцій SHA-1 і є однією з найсильніших доступних хеш-функцій. За допомогою цього онлайн-інструменту можна легко генерувати хеші SHA256.

Первісна версія алгоритму SHA-256 була створена Агентством національної безпеки США навесні 2002 року. Через кілька місяців Національний метрологічний університет опублікував новоявлений протокол шифрування в прийнятому на федеральному рівні стандарті безпечної обробки даних FIPS PUB 180-2.

Взимку 2004 він поповнився другою версією алгоритму [27].

Протягом наступних 3 років АНБ випустила патент на SHA другого покоління під ліцензією Royalty-free. Саме це дало старт застосування технології в цивільних сферах.

Крім того, SHA-256 має досить непогані технічні параметри:

- показник розміру блоку (байт) - 64.
- гранично допустима довжина повідомлення (байт) - 33.
- характеристика розміру дайджесту повідомлення (байт) - 32.
- стандартний розмір слова (байт) - 4.
- параметр довжини внутрішнього становища (байт) - 32.
- число ітерацій в одному циклі - всього 64.
- досягається протоколом швидкість (mib / s) - приблизно 140.

Робота алгоритму SHA-256 базується на методі побудови Меркле-Дамгарда, відповідно до якого початковий показник відразу після внесеної зміни розділяється на блоки, а ті, в свою чергу, на 16 слів [29].

Набір даних проходить крізь цикл, що нараховує 80 або 64 ітерації. Кожен етап характеризується запуском хешірування зі складових блок слів. Пара з них обробляються інструментарієм функції.

Далі результати перетворення складаються, видавши в підсумку вірний показник хеш-коду. Для генерації чергового блоку використовується значення попереднього. Перетворювати їх окремо один від одного не вийде.

2.3. Моделювання інформаційної системи

Здійснення процесу ІС у вигляді менеджеру паролів потребує наявності технічних та програмних засобів збирання, зберігання та оброблення даних.

Процес збереження та обробки даних зі зберігання паролів передбачає шифрування даних.

Спочатку вносимо дані паролю (вносимо дані облікового запису сайту користувача, код пароль та додаткову інформацію у вигляді посилання або коментарю, якщо такий потрібен), отримуємо упорядковані дані з шифрування, далі контролюємо та управляємо збереженими даними, та на завершальній стадії зберігаємо зашифровані дані в менежері паролів в БД.

Модель даних ІС у вигляді менеджеру паролів представлена на рис. 2.1.

Однією зі складових процесу підтримки прийняття рішень є ІС у вигляді менеджеру паролів.

ІС у вигляді менеджеру паролів ґрунтується на таких принципах, як здійснення збереження та візуалізації даних в режимі реального часу, що забезпечує оперативність збирання даних; відповідність між введеними даними та строками паролів; мінімізація ризику збирання помилкових даних; наявність алгоритмів оброблення зібраних даних; представлення даних за запитом.

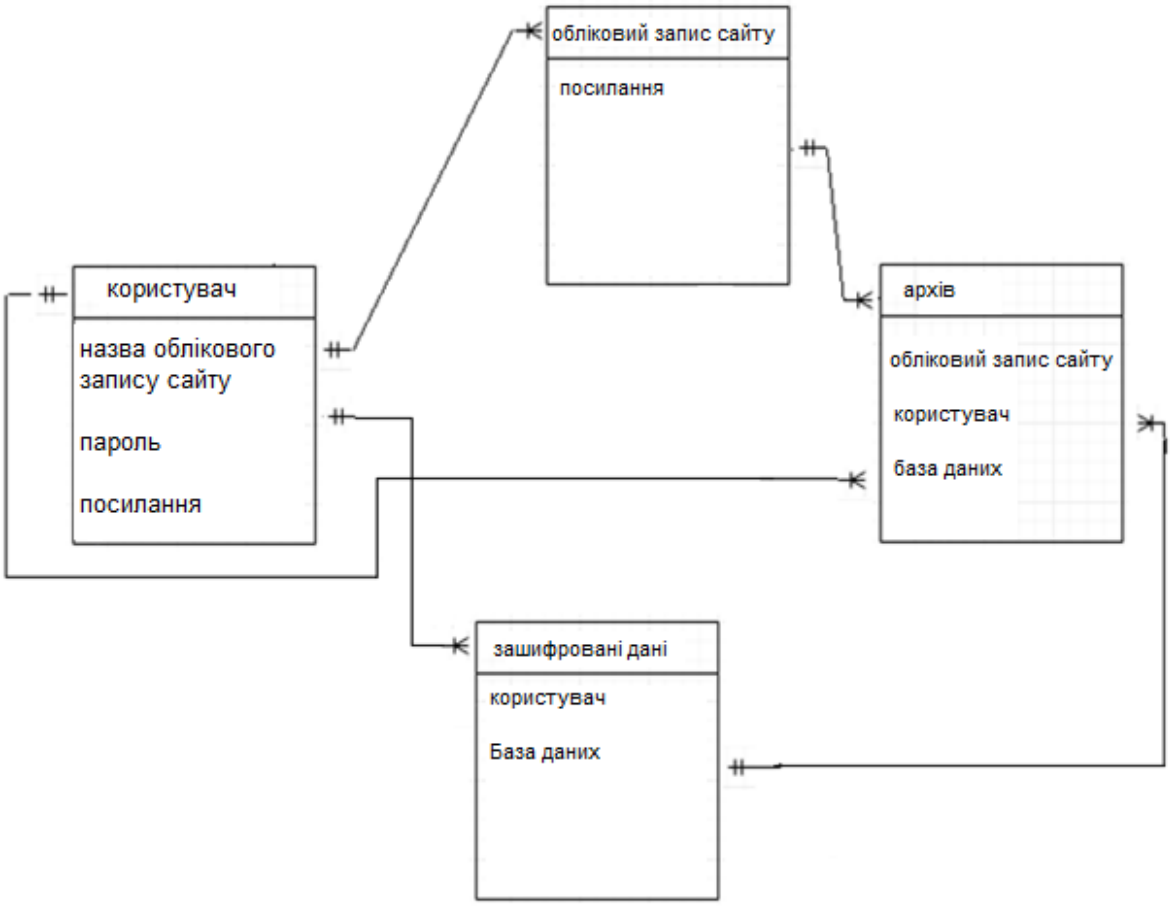

Рисунок 2.1 - Модель даних ІС

У цьому випадку система ІС спрямована на реалізацію функцій, що полягають у збиранні даних, які вводять користувачі, розміщенні зібраних даних у БД, візуалізації даних у вигляді зашифрованих даних-паролів, формуванні автоматичного відображення паролю на обліковому записі сайту.

На рис. 2.2 зображено схему процесу ІС у вигляді менеджеру паролів в загальному вигляді.

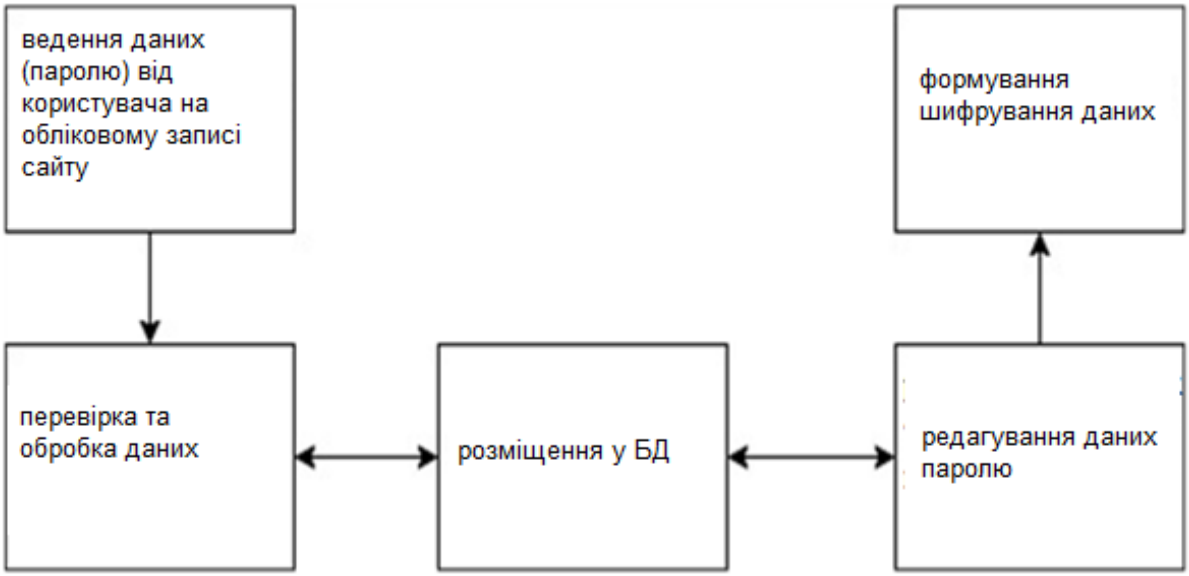

Рисунок 2.2 - Схема процесу ІС у вигляді менеджеру паролів в загальному вигляді

Для досягнення мети ефективної ІС у вигляді менеджеру паролів необхідна повна і своєчасна інформація від користувачів при здійсненні входу в обліковий запис. У зв'язку з цим необхідна розробка ІС, що дозволяє сформувати єдиний інформаційний менеджер з БД з зашифровування даних паролю.

Розробка такої ІС призначена вирішити такі завдання:

- забезпечити прийняття і обробку даних з шифрування паролю в режимі реального часу від користувачів;
- забезпечити можливість роботи зі спеціалізованими прогнозними і аналітичними програмами обробки даних;
- формувати зашифровування коду паролю.

ІС розроблялася у вигляді менеджеру, який створювався на мові програмування – C++, C, C#, html, css, javascript, sql.

Для збереження великих об’ємів даних управління проектами використовували БД MySql. Для передачі даних в найкоротший термін використовували сервер NGINX.

Відповідно до поставленого завдання пропонуємо застосувати ІС у вигляді менеджеру паролів, представлену на рис.2.3.

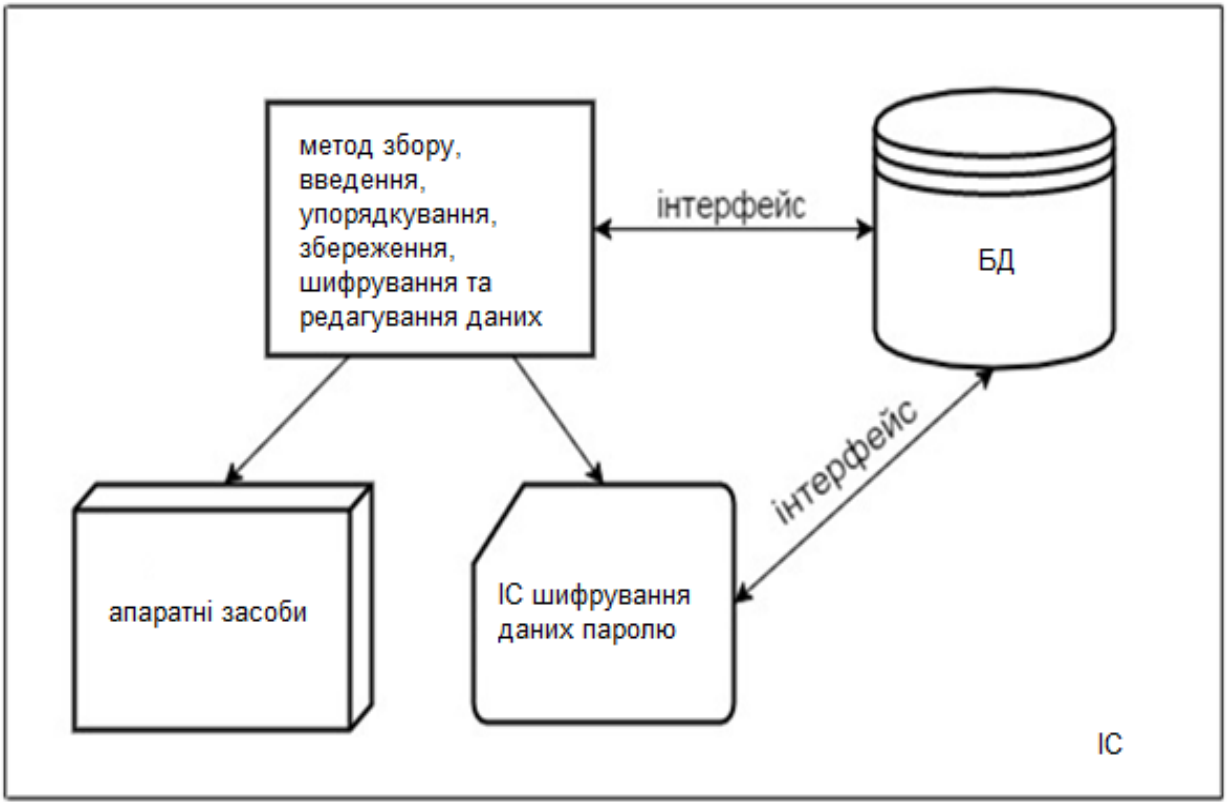

Рисунок 2.3 - Структура ІС у вигляді менеджеру паролів

ІС у вигляді менеджеру паролів повинна поєднувати ІС із застосуванням web-елементів та даних, що зберігаються у БД. Для зручності користувача повинен бути створений ергономічний інтерфейс.

Складові інформаційного забезпечення системи ІС у вигляді менеджеру паролів зображено на рис. 2.4.

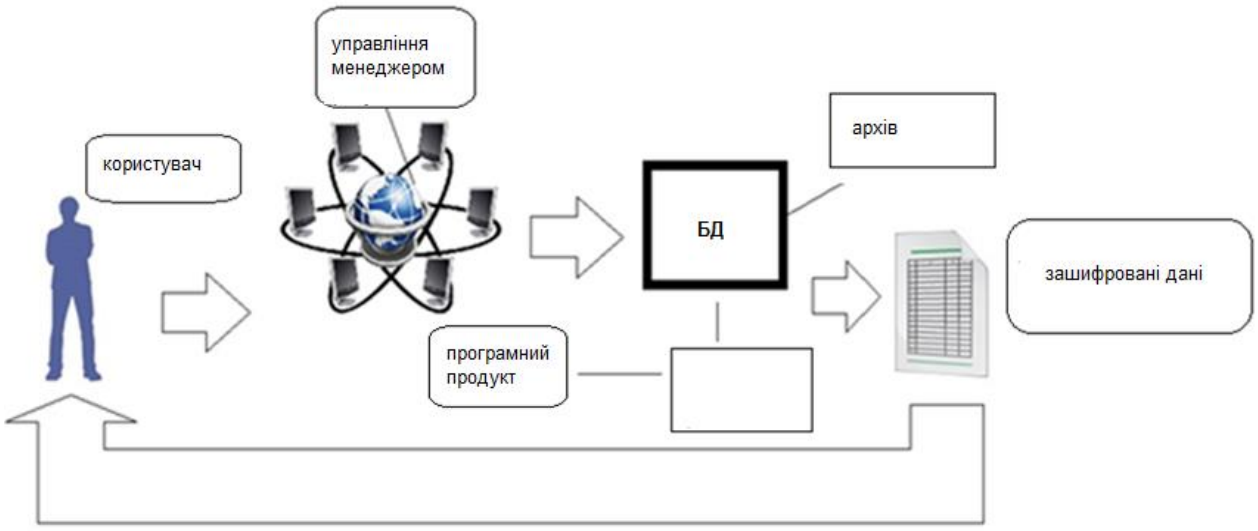

Рисунок 2.4 - Складові інформаційного забезпечення процесу підтримки прийняття рішень ІС у вигляді менеджеру паролів

ІС розробляється з метою вирішення наукових і прикладних завдань зі збереження та шифрування даними користувачів.

Основними вимогами до ІС є повнота, охоплення всіх сторін інформаційного, програмного, технічного забезпечення, які зустрічаються в процесі експлуатації системи.

Система повинна бути комплексною. Основні переваги ІС у порівнянні з традиційними методиками, полягають у можливостях спільного аналізу великих груп даних і їх взаємозв’язку. ІС повинна імітувати технологію шифрування та генерування. Система повинна бути відкритою, забезпечуючи легкість модифікації і експлуатації до нових умов для підтримки її на сучасному рівні.

ІС у вигляді менеджеру паролів будь-яких ресурсів має у зручному для користувача вигляді подавати певну інформацію у прив’язці до замовника даних. Такими системами є ІС з шифрування паролів.

Для збереження, візуалізації та шифрування даних була розроблена ІС для обробки даних, що зберігається у БД. Для зручності користувача розроблюється ергономічний інтерфейс.

Просторові дані та їх зв'язки із табличними даними погано описуються реляційною моделлю, тому повна модель даних в ІТ (рис. 2.5) має змішаний характер, тоді як семантична інформація об'єктів буде представлена реляційними таблицями.

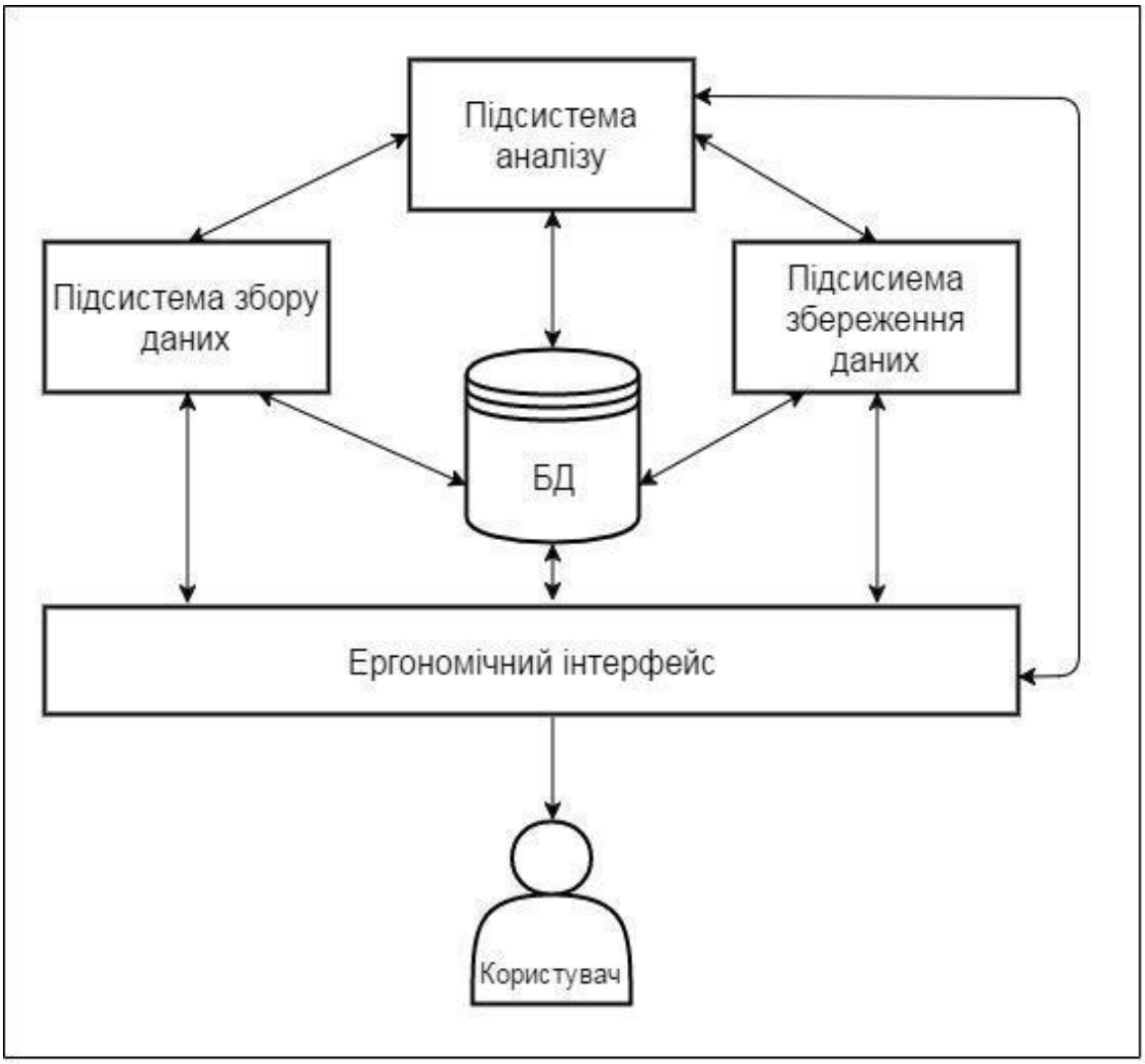

Рисунок 2.5 - Структура ІС у вигляді менеджеру паролів

Користувацька модель даних повинна задовольняти такі умови:

- БД повинна легко розширюватися при реорганізації та розширенні предметної зони;

- БД має легко переноситися при зміні програмного та апаратного середовища;

- дані до включення у БД слід перевіряти на достовірність;

- доступ до даних, які розміщуються в БД, повинні мати лише особи з відповідними повноваженнями;

- дані мають розміщуватись у форматах, доступних для розробленої ІС.

Процес шифрування паролів передбачає виконання функцій вводу та шифрування інформації облікового запису сайту.

ІС у вигляді менеджеру паролів реалізована на основі трирівневої клієнт-серверної архітектури побудови менеджерів - додатків.

Узагальнену архітектуру ІС, яка містить засоби збереження даних та маніпулювання даними (сервер БД), менеджер, робочу станцію, а також сукупність програмних модулів, що реалізують обробку та шифрування даних, представлено на рис. 2.6.

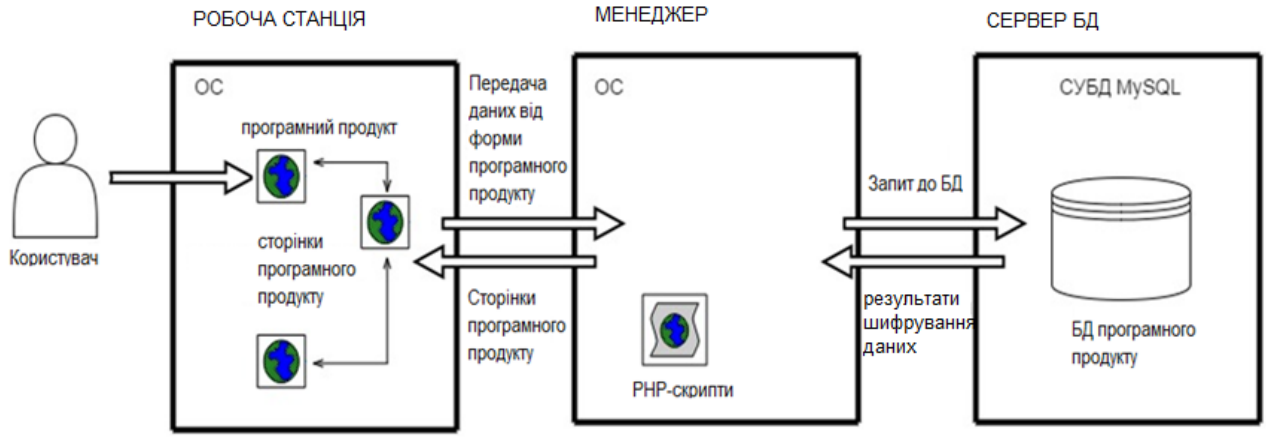

Рисунок 2.6 - Загальна архітектура ІС у вигляді менеджеру паролів

Застосування ІС у вигляді менеджера паролів, побудованої на основі представленої архітектури, забезпечує оперативне відслідковування поточних даних з шифрування паролів, а також інформаційне забезпечення контролю збереження даних.

Модель даних ІТ у вигляді менеджера паролів представлена у вигляді структури БД (рис. 2.7).

Проектована БД представлена на рис. 2.8.

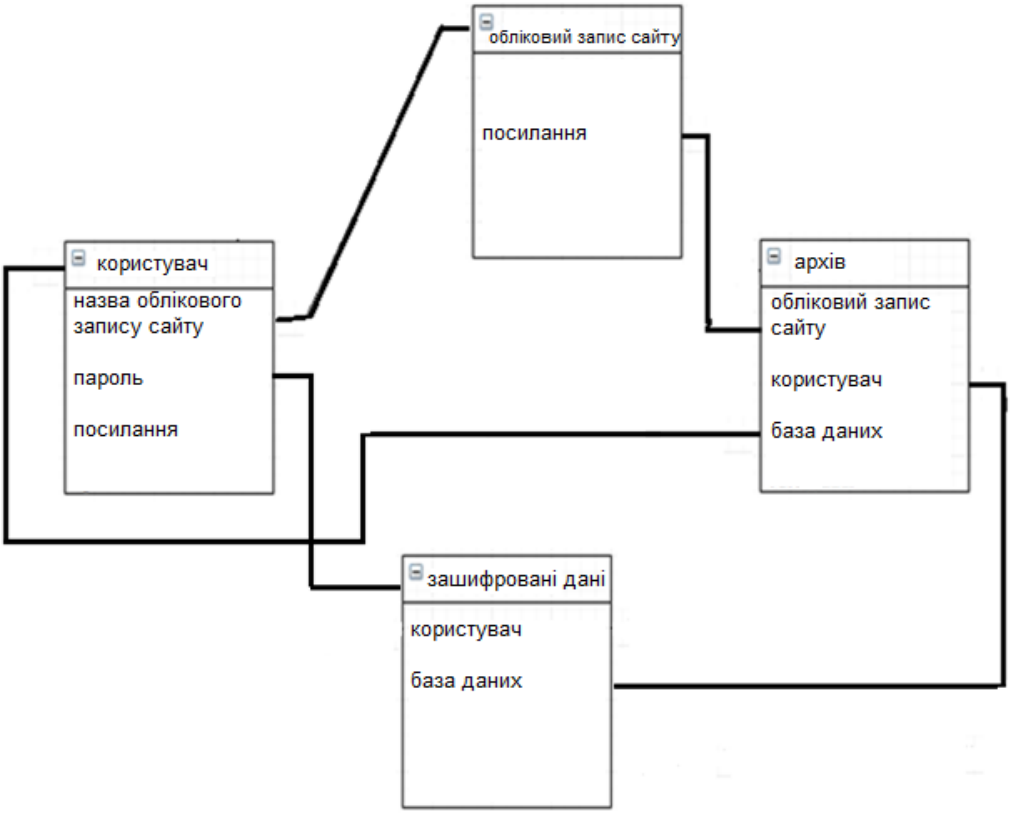

Рисунок 2.7 - Структура БД (проєктована)

Рисунок 2.8 - Структура БД (реалізована)

БД розробленої ІТ (рис. 2.8) містить в собі такі таблиці:

1) користувач – таблиця із даними облікового запису сайтів, посилання на сайт облікового запису, та містить дані паролю;

2) обліковий запис сайту – містить посилання на сайт;

3) архів – дані облікового запису сайтів, які містять інформацію про пароль сайту, дані користувача. Це все зберігається в БД;

4) зашифровані дані – містить дані користувача та базу даних;

Поняття варіанту використання (прецедент) вперше ввів Івар Якобсон і надав йому таку значущість, що в цей час варіант використання перетворився в основний елемент розробки та планування проєкту.

Варіант використання являє собою послідовність дій (транзакцій), виконуваних системою у відповідь на подію, що ініціюється деяким зовнішнім об'єктом (дійовою особою).

Варіант використання описує типову взаємодію між користувачем і системою. У найпростішому випадку варіант використання визначається в процесі обговорення з користувачем тих функцій, які він хотів би реалізувати.

Основні функції ІС у вигляді менеджеру паролів представлено на рис. 2.9 у вигляді UML діаграми варіантів використання.

Рисунок 2.9 - Діаграма варіантів використання ІС у вигляді менеджеру паролів

В ІС у вигляді менеджеру паролів головними акторами є адміністратор, користувач та БД.

Користувач вводить дані сайту та облікового запису, вводить дані паролю, може переглядати пароль та редагувати його.

БД містить дані про збереженого пароля, дані облікового запису сайту. Також БД містить раніше збережені та завершені паролі, генерує дані та зберігає шифровані дані.

Для реалізації запропонованої моделі та інформаційної технології розроблено ІС у вигляді менеджеру паролів, що є програмним комплексом, реалізованим у вигляді системи з трирівневою архітектурою. Модель даних ІС представлено у вигляді БД. Також для розробленої системи наведено опис функціональних можливостей розробленого менеджера, варіанти використання представлено у вигляді UML діаграми варіантів використання.

Висновки до 2 розділу

В розділі було розроблено вимоги і модельовано інформаційну систему, а також було сформовано постановку задачі, проведено аналіз і специфікація вимог до інформаційної системи.

Найменування і підстава для створення найменування: «Розробка кросплатформеної програмної системи зберігання паролів».

Призначення: Зберігання в зашифрованому вигляді і використання даних користувача для аутентифікації.

Мета створення системи: Дослідження методів захисту інформації в навчальних цілях.

Вимоги до структури: проєкт повинен бути розбитий на кілька частин (по функціональності і виконуваних завдань).

Вимоги до функціонування:

– для дешифрування даних система повинна вимагати «майстер-пароль» і шлях до файлу;

– у програмній системі має бути дерево навігації створеними записами;

– повинні бути передбачені такі функції як: створення, видалення, редагування записів.

Вимоги до інтерфейсу системи:

– інтерфейс повинен відображати настроювані параметри системи;

– інтерфейс повинен бути розділений на 2 основні частини: дерево навігації та інформація про обрані записи.

– інтерфейс повинен бути інтуїтивно зрозумілим;

Вимоги до технічного забезпечення:

– базова мова реалізації - C++, C#.

– версія компілятора - version clang 6.0.0 - 1ubuntu2;

– система збірки – Cmake + MakeFiles використовувані в середовищі розробки;

– IDE – Qtcreator.

Визначено, що найважливішими вимогами до менеджера паролів як сервісу зі зберігання інформаційних даних є:

– підвищення достовірності та персональної відповідальності даних збору та візуалізації;

– автоматизація процесу збору даних;

– функції обробки та збереження масивів даних;

– створення нової інформації за допомогою аналізу та синтезу наявних даних;

– підтримка тематичної настройки вхідних потоків даних під інформаційні потреби методології підтримки прийняття рішень;

– багатокористувацький інтерфейс;

– кросплатформеність.

РОЗДІЛ 3
ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ КОМПОНЕНТІВ СИСТЕМИ

3.1 Опис інструментальних засобів та підходів до розробки з обґрунтуванням їх вибору

3.1.1 Інформаційне забезпечення

ІС має достатньо складну структуру і розглядається як комплекс апаратного, програмного та інформаційного забезпечення.

Інформаційне забезпечення складається з:

- сукупності масивів інформації,
- систем кодування,
- класифікації інформації.

Інформаційне забезпечення це реалізовані рішення за видами, обсягами, розміщенням і формами організації інформації, зокрема пошук і оцінка джерел даних, набір методів введення даних, проєктування БД, їх ведення та метасупровід.

Дані про замовників, проєкт, та їх документацію і пов’язані з ними табличні дані заповнюються самим користувачем чи адміністратором, або у процесі перетворення та синтезу інфраструктури даних.

3.1.2 Технічне забезпечення

Технічне забезпечення складається з комплексу апаратних засобів, що застосовуються при функціонуванні ІТ:

- персональний комп'ютер (ПК),
- пристрої введення-виведення інформації,
- пристрої обробки і зберігання даних,
- засобів телекомунікації.

ПК є ядром будь-якої інформаційної системи і призначений для управління роботою ІТ та виконання процесів обробки даних, основаних на обчислювальних або логічних операціях. Створена ІТ здатна оперативно обробляти невеликі масиви інформації, упорядковувати і візуалізувати результати.

Введення даних реалізується за допомогою технічних засобів і методів:

– з клавіатури,
– за допомогою дигітайзера,
– через зовнішні комп'ютерні системи.

Всі дані отримуються методом генерування даних про паролі мережі, інтернету тощо, а також електронними приладами.

Пристрої для обробки та зберігання даних сконцентровані в системному блоці, в який входить центральний процесор, оперативна пам'ять, зовнішні запам'ятовувальні пристрої і користувальницький інтерфейс.

Пристрої виводу даних забезпечують наочне подання результатів на моніторі.

Для функціонування програмного продукту необхідно наступне технічне забезпечення з наступними мінімальними характеристиками:

Серверна частина:

– комп'ютер з процесором Intel Pentium 4 / Athlon 64 або пізнішої версії з підтримкою SSE2;
– оперативна пам'ять 4 Гб (рекомендується і більше для швидкої обробки даних);
– місце на жорсткому диску від 2 Гб (рекомендується і більше для швидкої обробки даних).

Технічні характеристики сервера будуть уточнені після завершення системи і широкого тестування всіх модулів порталу.

Клієнтська частина:

– комп'ютер з процесором Pentium IV 1.5 ГГц (рекомендується і більше для швидкої обробки даних);
– оперативна пам'ять 512 Мб (рекомендується і більше для швидкої обробки даних).

3.1.3 Програмне забезпечення

Програмне забезпечення складається з сукупності програмних засобів, що реалізують функціональні можливості ІТ і програмних документів, необхідних при їх експлуатації.

Структурно програмне забезпечення ІТ охоплює базові та прикладні програмні засоби.

До базових програмних засобів належать:

- операційні системи (ОС),
- програмні середовища,
- мережеве програмне забезпечення,
- системи управління базами даних (СУБД).

ОС призначені для управління ресурсами електронно-обчислювальної техніки і процесами, які використовують ці ресурси.

ІС з міжнародних та внутрішніх перевезень працює з даними для обміну регулярними даними для різних успадкованих різноформатних систем. Для їх ведення програмне забезпечення містить СУБД, а також модулі управління засобами введення і виведення даних, систему візуалізації даних і модулі для виконання аналізу.

Прикладні програмні засоби призначені для розв'язання спеціалізованих задач й реалізувалися у вигляді окремих програм і утиліт.

Для оптимальної ефективності радимо дотримуватися наступних вимог до програмного продукту:

Вимоги до програмного забезпечення серверної частини. Для функціонування програмного продукту необхідно наступне програмне забезпечення:

- операційна система - Windows XP з пакетом оновлень 2 +, Windows Vista, Windows 7, Windows 8, Windows 10; Linux та Macos.
- СУБД - MySQL версії не нижче 3.23.

Програмний продукт повинен бути працездатний - інформація, розташована на ньому, повинна бути доступна.

3.2. Опис програми

Структура файлу з базою даних (.kdb, .kdbx) складається з 3 частин:

– підпис (не зашифровані);
– тема (не зашифровані);
– дані (зашифровані).

Далі дешифруються базу даних lebudka.kdbx.

Розшифровка бази даних має наступну послідовність дій :

1) читання підпису бази даних,
2) читання заголовку бази даних,
3) генерація майстер-ключа,
4) розшифрування бази даних,
5) перевірка даних на цілісність,
6) якщо файл був стиснутий, розпаковуємо його,
7) розшифровуємо паролі.

У .kdbx файлах кожне поле заголовка складається з 3 частин:

– ID поля (1 байт): можливі значення від 0 до 10.
– довжина даних (2 байти).
– дані ([довжина даних] байт)

Тема .kdbx файлу складається з наступних полів:

– ID = 0x01 Comment: дане поле може бути представлено в заголовку, але в моїй базі даних його не було.
– ID = 0x02 Cipher ID: UUID, який вказує на використовуваний метод шифрування (наприклад, для AES 256 UUID = [0x31, 0xC1, 0xF2, 0xE6, 0xBF, 0x71, 0x43, 0x50, 0xBE, 0x58, 0x05, 0x21, 0x6A, 0xFC, 0x5A, 0xFF]).
– ID = 0x03 Compression Flags: ID алгоритму, що використовується для стиснення бази даних: 0x00: None; 0x01: GZip.

- ID = 0x04 Master Seed: використовується для створення майстер-ключа.
- ID = 0x05 Transform Seed: використовується для створення майстер-ключа.
- ID = 0x06 Transform Rounds: використовується для створення майстер-ключа.
- ID = 0x07 Encryption IV: використовується для розшифровки даних.
- ID = 0x08 Protected Stream Key: використовується для розшифровки паролів.
- ID = 0x09 Stream Start Bytes: перші 32 байта розшифрованої бази даних. Вони використовуються для перевірки цілісності розшифрованих даних і коректності майстер-ключа. Ці 32 байта рандомно генеруються кожен раз, коли в файлі зберігаються зміни.
- ID = 0x0A Inner Random Stream ID: ID алгоритму, що використовується для розшифровки паролів: 0x00: None; 0x01: ARC4; 0x02: Salsa20.
- ID = 0x00 End of Header: останнє поле заголовка бази даних, після нього починається сама база даних.

Генерація майстер-ключа має наступні псевдокоди:

- пароль - sha256 (sha256 (password))
- файл-ключ- sha256 (sha256 (keyfile))
- пароль + Файл-ключ - sha256 (concat (sha256 (password), sha256 (keyfile)))
- Windows User Account (WUA) -sha256 (sha256 (WUA))
- пароль + Файл-ключ + (WUA)- sha256 (concat (sha256 (password), sha256 (keyfile), sha256 (WUA)))

Згенерований майстер-ключ шрифта представлений в Додатку Б.

За допомогою SHA256 отримали хеш від зашифрованого складеного ключа. З'єднали Master Seed з заголовка з отриманим хешем. За допомогою SHA256 отримуємо хеш від об'єднаної послідовності - майстер-ключ.

Код програми розшифровування даних представлено в Додатку В.

3.3. Результати реалізації інформаційної системи

Запуск програми починається з файлу ulk.kdbx.

Після запуску менеджеру відкривається головна сторінка програмного продукту (рис. 3.1.).

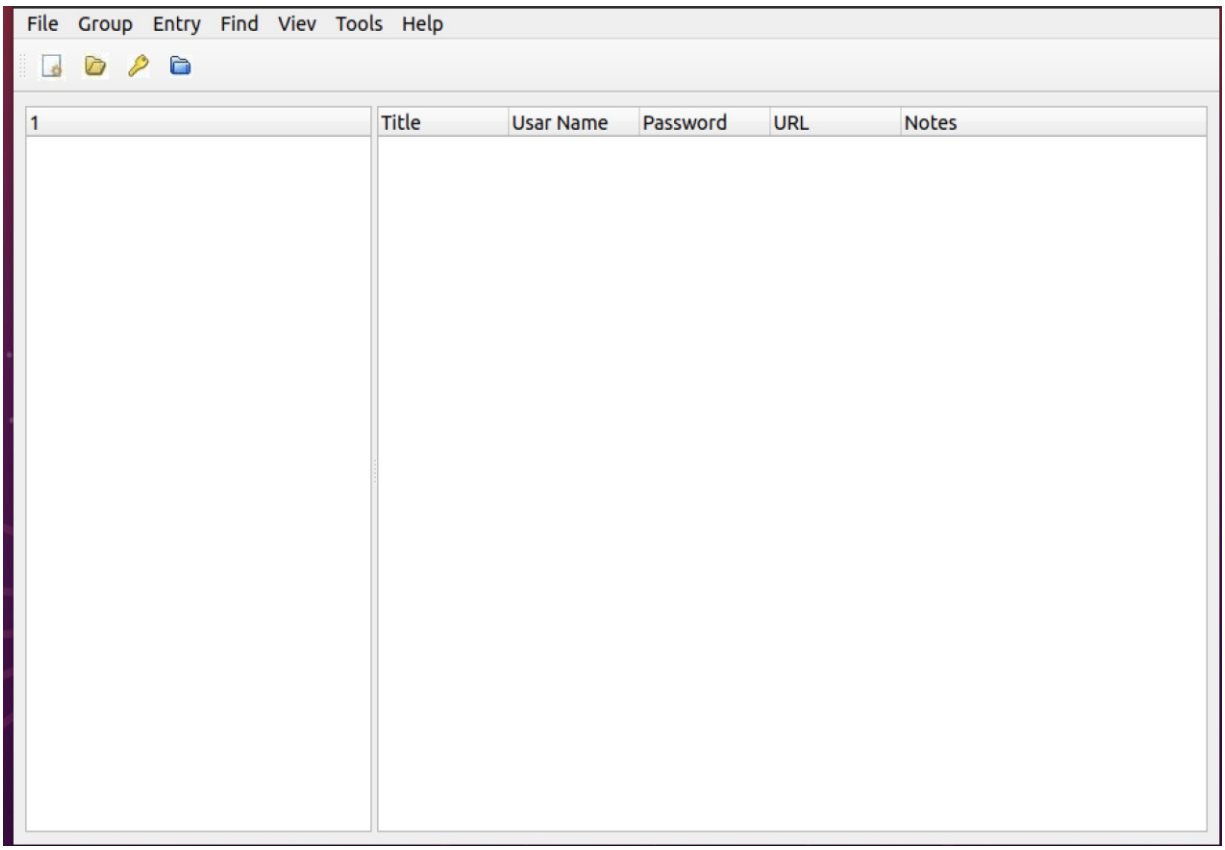

Рисунок 3.1 – Головне вікно менеджеру паролів (інтерфейс)

На головному вікні розташовані іконки, меню та головні вікна зображені на рис. 3.2 та рис. 3.3.

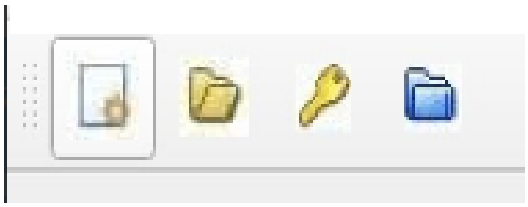

Рисунок 3.2 – Іконки менеджеру паролів

Іконки призначені для того, щоб можливо було створити новий елемент в теці, відкрити необхідну теку, створити ключ взаємозв’язку та створювати нові теки та підтеки.

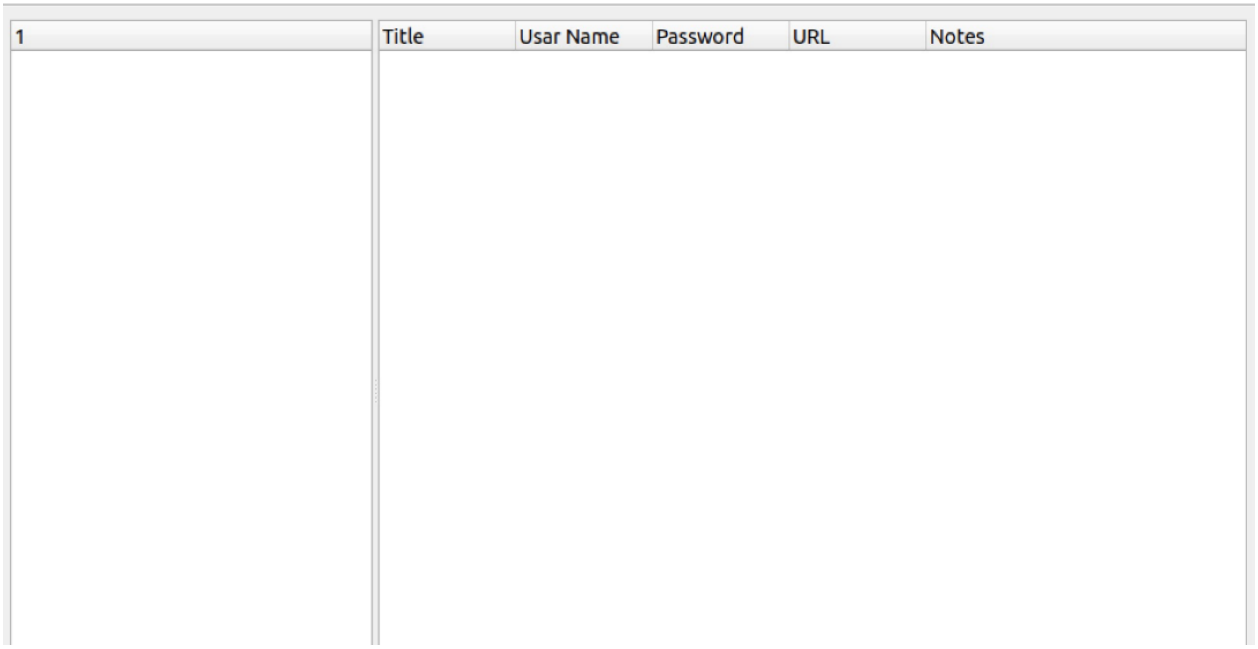

Рисунок 3.3 –Головні вікна менеджеру паролів

Головні вікна менеджеру паролів призначені для розділення дерева каталогу та відображення необхідної інформації.

З лівого боку створено вікно для дерева каталогу, в якому відображаються складові, чиї паролі зберігаються в менеджері.

З правого боку відображаються дані по ключу – його назва, ім’я користувача та пароль. Також тут можна указати посилання на сторінку в інтернеті (наприклад, будь то посилання на соціальну мережу користувача, або на контактні дані, тощо). Також є поле, де можна вказувати будь-яку інформацію – notes.

Далі розглянемо складові кожного елементу дерева каталогу (рис. 3.4 – 3.8).

Вікно General (загальні дані) містить дані про ключ – його назву, ім’я адміністратора, пароль, поле для посилання та нотаток.

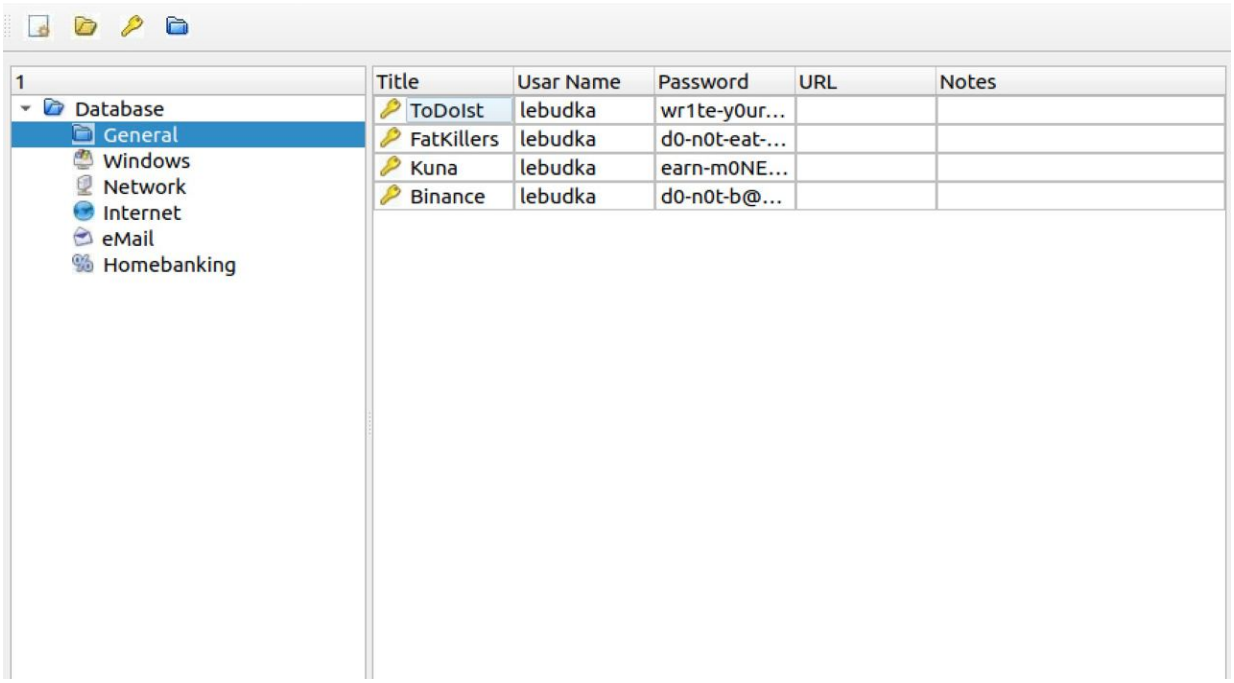

Рисунок 3.4 – Вікно General

Вікно користувачів системи Windows містить інформацію про дані про ключ – його назву, ім’я користувача, пароль, поле для посилання та нотаток (рис. 3.5).

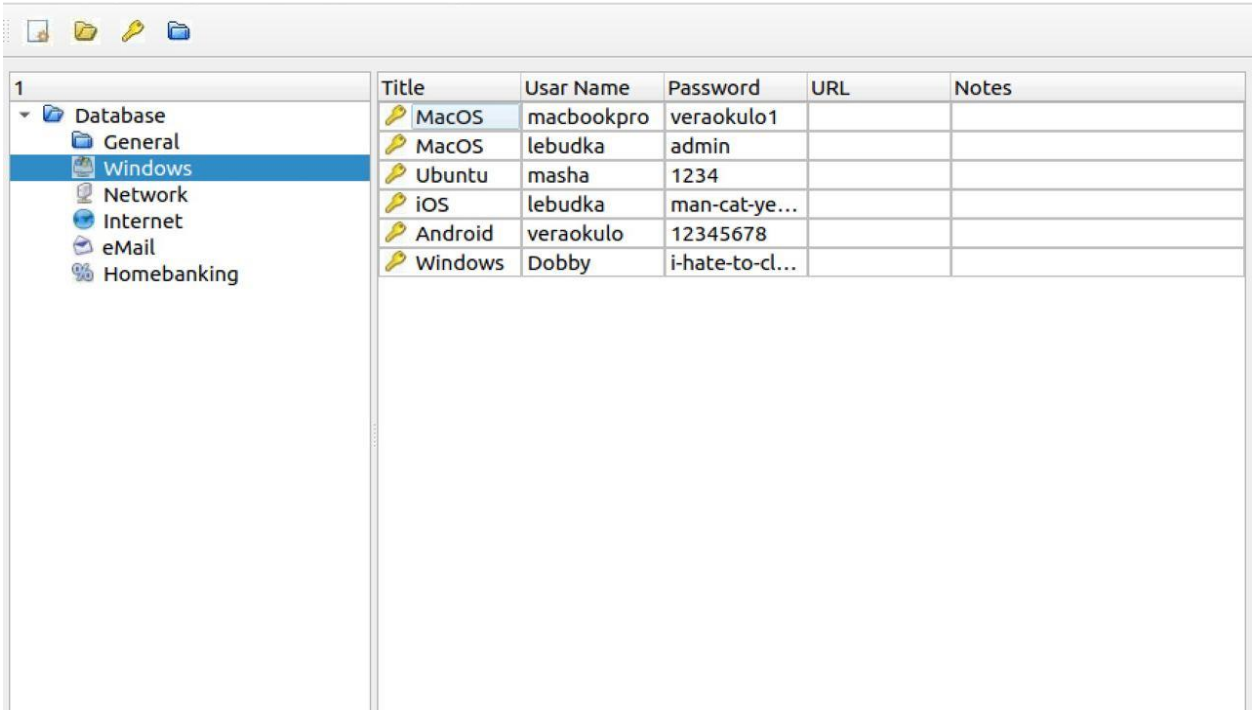

Рисунок 3.5 – Вікно Windows

Пароль може містити в своєму складі як літери, так і цифри, а також тире.

В даному вікні ключем є назва операційної системи, наприклад, MacOS, як в першому та другому рядках.

Наступним є вікно мереж (рис. 3.6).

Вікно Network аналогічно в своєму складі містить інформацію про дані про ключ – назву типу мережі, наприклад, wi-fi, ім’я користувача, пароль, поле для посилання та нотаток.

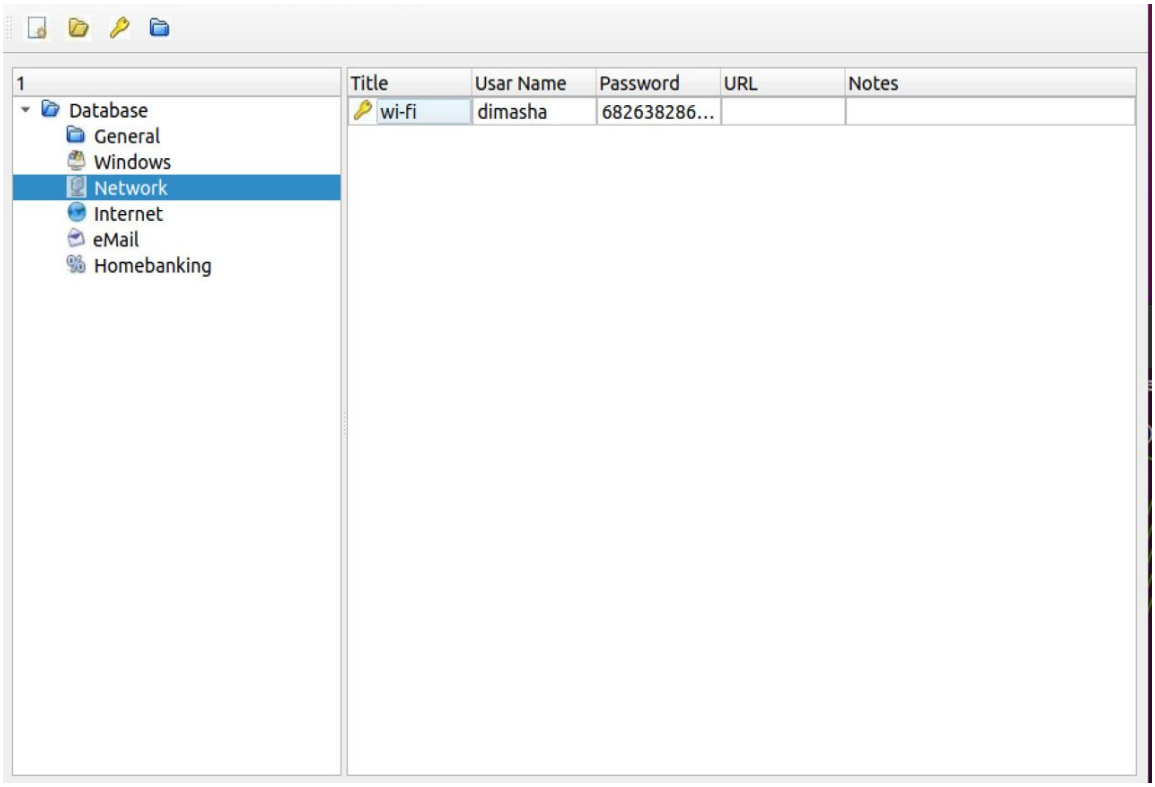

Рисунок 3.6 – Вікно Network

Пароль в такій ситуації складається тільки з цифр.

Вікно eMail також в своєму складі містить інформацію про дані та ключ, але в каталогу eMail це домен електронної пошти (рис. 3.7). Це може бути Gmail. yahoo. ukr.net, тощо. Також містить дані електронної пошти користувача, пароль, поле для посилання та нотаток.

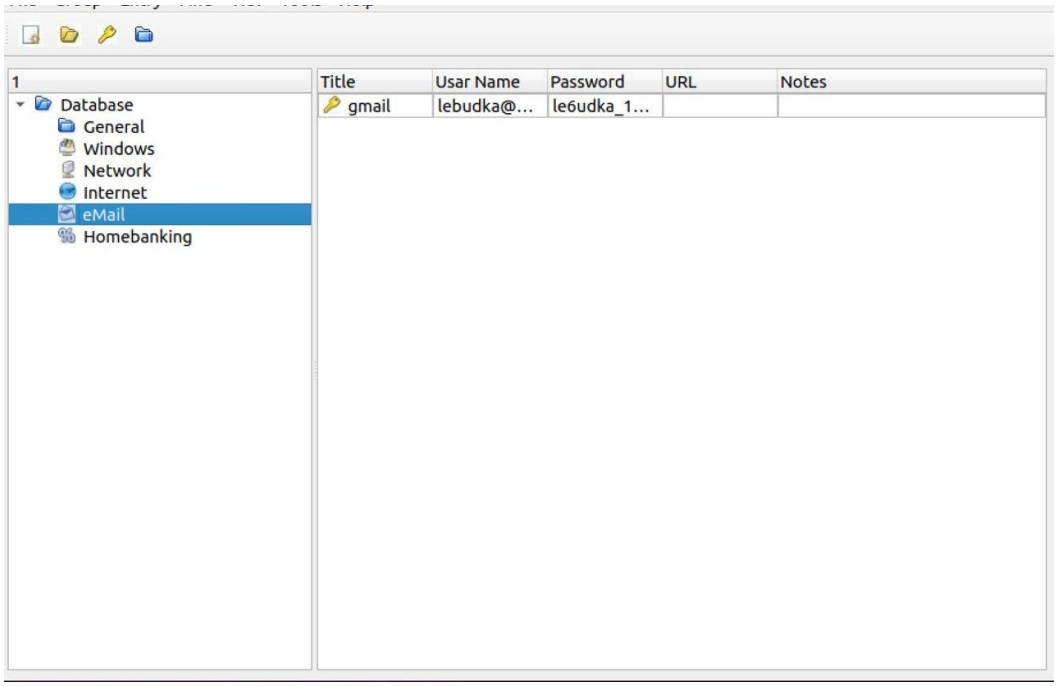

Рисунок 3.7 – Вікно eMail

Пароль в даному вікні eMail може складатися з цифр, літер, знаків.

Вікно Homebanking в своєму складі містить інформацію про дані про ключ – назву банку, наприклад, monobank, privat24, abank24, alfa-mobile Ukraine, oschad 24/7,тощо, ім’я користувача, пароль, поле для посилання та нотаток (рис. 3.8).

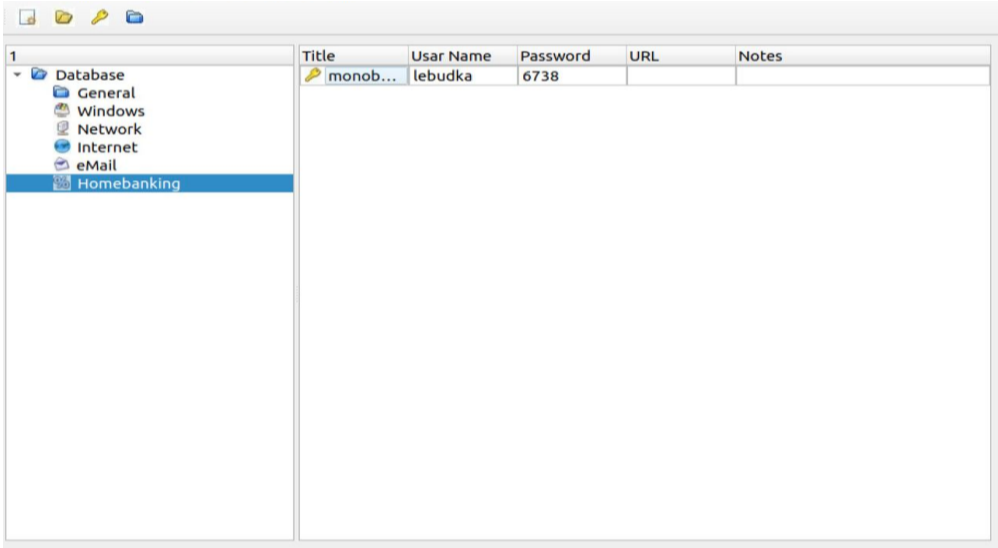

Рисунок 3.8 – Вікно Homebanking

Пароль в цьому випадку складається тільки з цифр.

Узагальнений вигляд програмного продукту з всіма даними представлений на рис. 3.9.

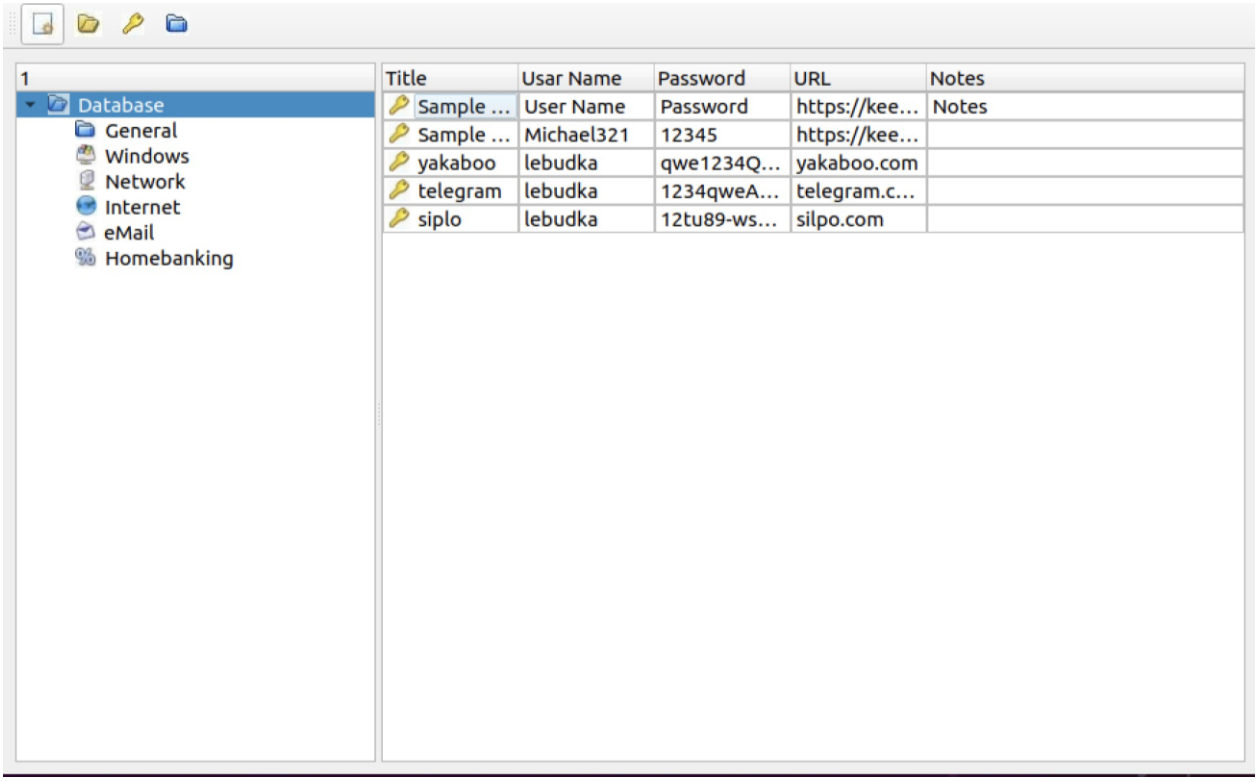

Рисунок 3.9 – Загальний вигляд програмного продукту з паролями

Висновки до 3 розділу

В розділі було спроєктовано та реалізовано компоненти системи, а саме було проведено опис інструментальних засобів та підходів до розробки з обґрунтуванням їх вибору та візуалізовано результати розробки інформаційної системи.

Таким чином, було досягнуто мету роботи, яка полягає у створенні менеджеру паролів, що дозволяє вчасно шифрувати дані користувача.

ВИСНОВКИ

Програмна система зберігання паролів (далі менеджер паролів) - програмне забезпечення, яке допомагає користувачеві працювати з паролями і PIN-кодами. У подібного програмного забезпечення зазвичай є місцева база даних або файли, які містять зашифровані дані пароля.

Менеджери паролів можуть також використовуватися як захист від фішінга (вид інтернет-шахрайства, метою якого є отримання доступу до конфіденційних даних користувачів - логінів і паролів).

На підставі аналізу наявних систем, а також на підставі опитування користувачів (в тому числі і потенційних) був складений загальний перелік вимог до розробленої програмної системи зберігання паролів:

1) кросплатформеність - є для нас важливим критерієм, через те, що ця функція дозволяє використовувати програму на різних пристроях.

2) зручний інтерфейс користувача - є важливим критерієм, тому, що саме інтерфейс дозволяє користувачеві взаємодіяти з програмою.

3) генератор паролів - як уже говорилося раніше, ця функція, яка у випадковому порядку формує набір нелогічних символів більшою чи меншою мірою складності, що залежить від заданих користувачем параметрів, служить для того, щоб допомагати користувачам не сидіти і крутити мозком над, тим який й би пароль придумати. Це суттєво економить час користувача.

4) безкоштовне розповсюдження - менеджер паролів розробляється з метою спрощення буднів користувачів Інтернет-ресурсами, а не з метою отримання матеріальної вигоди.

Для реалізації завдання наукової роботи необхідно виконання наступних завдань:

– виконати огляд застосування ІТ у вигляді менеджеру паролів;

– визначити структуру інформації менеджеру паролів;

– провести аналіз моделей, методів та інструментальних засобів для зберігання і обробки даних менеджеру паролів;

– розробити та реалізувати менеджер паролів для реалізації запропонованої ІС.

Розроблений програмний продукт має забезпечити спрощення вводу даних паролів, зберігання даних паролів, а також буде містити архів паролів.

Вимоги до функціонування:

– для дешифрування даних система повинна вимагати «майстер-пароль» і шлях до файлу;

– у програмній системі має бути дерево навігації створеними записами;

– повинні бути передбачені такі функції як: створення, видалення, редагування записів.

4. Вимога до інтерфейсу системи

– інтерфейс повинен відображати настроюються параметри системи;

– інтерфейс повинен бути розділений на 2 основні частини: дерево навігації та інформація про обрані записи.

– інтерфейс повинен бути інтуїтивно зрозумілим;

5. Вимоги до технічного забезпечення

– базова мова реалізації - C++, C#.

– версія компілятора - version clang 6.0.0 - 1ubuntu2;

– система збірки – Cmake + MakeFiles використовувані в середовищі розробки;

– IDE – Qtcreator.

На сьогодні найважливішими вимогами до менеджера паролів як сервісу зі зберігання інформаційних даних є:

– підвищення достовірності та персональної відповідальності даних збору та візуалізації;

– автоматизація процесу збору даних;

– функції обробки та збереження масивів даних;

– створення нової інформації за допомогою аналізу та синтезу наявних даних;

– підтримка тематичної настройки вхідних потоків даних під інформаційні потреби методології підтримки прийняття рішень;

– багатокористувацький інтерфейс;

– кросплатнорменність.

В процесі роботи було узагальнено і досліджено:

– проведено аналіз літературних джерел та практичного досвіду використання інформаційних систем і технологій в предметній галузі.

– виконано аналіз інформації для формування паролів та зашифрованих даних.

– визначено структуру інформаційної технології та складові інформаційного забезпечення процесу збереження та шифрування даних в менеджері паролів.

– розроблено модель даних інформаційної технології та сформовано функціональну модель процесу збереження та візуалізації даних менеджеру паролів.

– на основі розроблених моделей та розробленої інформаційної технології запропоновано інформаційну систему, для якої побудовано архітектуру, спроєктувати та здійснено програмну реалізацію.

Таким чином, було досягнуто мету роботи, яка полягає у створенні менеджеру паролів, що дозволяє вчасно шифрувати дані користувача.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Арянова Т. Ethereum для начинающих: Полное руководство [Електронний ресурс] / Тая Арянова. – 2017. – Режим доступу до ресурсу: https://ru.ihodl.com/tutorials/2017-06-30/ethereum-dlya-nachinayushih-polnoerukovodstvo/.

2. Виявлення та розслідування злочинів, що вичиняються у сфері інформаційних технологій: Наук.-практ.посіб./ За заг.ред.проф. Я.Ю.Кондратьєва. – К., 2004.

3. Выростков Д. Обзор способов и протоколов аутентификации в вебприложениях [Електронний ресурс] / Дмитро Выростков. – 2015. – Режим доступу до ресурсу: https://habr.com/ru/company/dataart/blog/262817/.

4. Документация FreeIPA. URL: https://www.freeipa.org/page/Documentation (дата обращения 17.11.2019)

5. Інформаційна безпека людини як споживача телекомунікаційних послуг: Монографія / І.В. Арістова, Д. В. Сулацький ; НДІ інформатики і права НАПрН України. — К. : Право України; Х. : Право, 2013. — 184 с.

6. Кормич Б.А. Інформаційна безпека: організаційно-правові основи: Навч. посібник. — К.: Кондор, 2004. — 384 с.

7. Лукацкий А. Обнаружение атак. — СПб.: БХВ-Петербург, 2001. — 624 с.

8. Луцкер А. Авторское право в цифровых технологиях и СМИ. – М., 2005.

9. Мандиа К., К. Просис. Защита от вторжений. Расследование компьютерных преступлений.– М., 2005.

10. Малюк А.А. Информационная безопасность: концептуальные и методологические основы защиты информации: Учеб. пособие. – М.: Горячая линия – Телеком, 2004. – 280 с.

11. Ніколаюк С.І., Никифорчук Д.Й., Томма Р.П., Барко В.І. Протидія злочинам у сфері інтелектуальної власності. – К., 2006.

12. Руководство по выживанию Kerberos. [Електронний ресурс] – Режим доступу до ресурсу: https://pro-ldap.ru/tr/zytrax/tech/kerberos.html (дата обращения 16.11.2019)

13. Соколов А.В., Шаньгин В.Ф. Защита информации в распределенных корпоративних сетях и системах. – М.: ДМК Пресс, 2002. – 656 с.

14. Угрозы несанкционированного доступа [Електронний ресурс]. – 2008. – Режим доступу до ресурсу: https://zakonbase.ru/content/part/1183301.

15. Чумак О.В Энтропии фракталы в анализе данных – НИЦ «Регулярная и хаотическая динамика», Институт компьютерных исследований – М.–Ижевск 2011 – С. 164.

16. Щеглов А.Ю. Защита компьютерной информации от несанкционированного доступа. — СПб.: «Наука и техника», 2004. — 384 с.

17. Якимов В.И Защита информации в компьютерных сетях. Методы защиты информации – Москва, 2013 – С. 476.

18. adminCraft. Введение в Ethereum и Solidity [Електронний ресурс] / adminCraft. – 2018. – Режим доступу до ресурсу: https://craftappmobile.com/vvedenie-v-%E2%80%8B%E2%80%8Bethereum%E2%80%8B%E2%80%8B-is%E2%80%8Bolidity/.

19. Blockchain Privacy Policy [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://www.blockchain.com/ru/legal/privacy.

20. Desmond B., Richards J. Active Directory: Designing, Deploying, and Running Active Directory O'Reilly Media, Fifth edition – May 31, 2013 – P. 738.

21. Gupta R. Hands-On Cybersecurity with Blockchain [Електронний ресурс] / Rajneesh Gupta. – 2018. – Режим доступу до ресурсу: https://subscription.packtpub.com/book/networking_and_servers/97817889901 89.

22. Gerald C. LDAP System Administration. Putting Directories to Work / O'Reilly Media – 2009 – P. 321

23. Muehlfeld M., Hanzelka F., Maňásková L. Red Hat Enterprise Linux 7. Linux Domain Identity, Authentication, and Policy Guide – 2019 – P. 544.

24. Malan. Классификация механизмов аутентификации пользователей и их обзор [Електронний ресурс] / Malan. – 2013. – Режим доступу до ресурсу: https://habr.com/ru/post/177551/.

25. Moffett J. Security & Distributed Systems [Електронний ресурс] / Jonathan Moffett – Режим доступу до ресурсу: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.22.2312&rep=rep1& type=pdf.

26. Nemeth E. Snyder G. UNIX and Linux System Administration Handbook 5th Edition Addison-Wesley Professional – 2017 – P. 1232.

27. Nitish Singh. Смарт-контракты: основное пособие для начинающих [Електронний ресурс] / Nitish Singh. – 2018. – Режим доступу до ресурсу: https://101blockchains.com/ru/%D1%81%D0%BC%D0%B0%D1%80%D1%8 2-%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%B0%D0%BA%D1% 82%D1%8B-%D0%B1%D0%BB%D0%BE%D0%BA%D1%87%D0%B5%D0%B9%D0% BD/#7.

28. Panda Security. Серьезное влияние блокчейна на информационную безопасность [Електронний ресурс] / Panda Security. – 2019. – Режим доступу до ресурсу: https://www.cloudav.ru/mediacenter/security/blockchainprofound-effect-cybersecurity/.

29. Timothy A.H., Smith M.C., Gordon S.G. Understanding and Deploying LDAP Directory services – 2003 – P. 430

30. Web Application Security [Електронний ресурс] – Режим доступу до ресурсу: http://java.boot.by/wcd-guide/ch05s03.html.

ДОДАТОК А

Лістінг EntryForm

```
#ifndef ENTRYFORM_H
#define ENTRYFORM_H

#include <QDialog>
#include <QMainWindow>
#include <QTableWidget>
#include <QTableWidgetItem>

#include <memory>
#include <algorithm>

#include <entry.hh>


#include <Form.h>

class MainWindow;
class IconChooser;

using namespace keepass;

namespace Ui {
   class EntryForm;
}

class EntryForm : public QDialog, private Form
{
   Q_OBJECT

public:
   explicit EntryForm(  MainWindow *main,
                        QWidget *parent = nullptr );
   ~EntryForm();

   void setEntry( std::shared_ptr<Entry> entry );
   std::shared_ptr<Entry> getEntry( );
```

```
    int exec( ) override;

private slots:
    void on_del_clicked();
    void on_save_clicked();
    void on_icon_clicked( );

private:
    Ui::EntryForm  *ui;
    IconChooser *icon_chooser;

    std::shared_ptr<Entry> m_entry;

    friend class IconChooser;
    friend class IconContainer;
};

#endif // ENTRYFORM_H
```

ДОДАТОК Б

Лістінг Form

```
#ifndef FORM_H
#define FORM_H

#include <cstdint>

class MainWindow;

class Form
{
public:
    Form( MainWindow *main );

    uint32_t m_icon;
    MainWindow *main_window;
};

#endif // FORM_H
```

ДОДАТОК В

Лістінг GroupForm

```
#ifndef GROUPFORM_H
#define GROUPFORM_H

#include <QDialog>

#include <group.hh>

#include "Form.h"

using namespace keepass;

class IconChooser;
class IconContainer;

namespace Ui {
class GroupForm;
}

class GroupForm: public QDialog, private Form
{
    Q_OBJECT

public:
    explicit GroupForm( MainWindow *main,
                        QWidget *parent = nullptr);
    ~GroupForm();

    void setGroup( std::shared_ptr<Group> group );
    std::shared_ptr<Group> getGroup( );

    int exec() override;

private slots:
    void on_del_clicked();
    void on_save_clicked();
    void on_icon_clicked();
```

```
private:
    Ui::GroupForm *ui;
    IconChooser *icon_chooser;
    std::shared_ptr<Group> m_group;

    friend class IconChooser;
    friend class IconContainer;
};

#endif // GROUPFORM_H
```

ДОДАТОК Г

Лістінг mainwindow

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QTreeWidget>
#include <QTreeWidgetItem>
#include <QIcon>
#include <QClipboard>
#include <QMessageBox>
#include <QFileDialog>
#include <QInputDialog>

#include <iostream>
#include <exception>
#include <memory>

#include <kdbx.hh>
#include <kdb.hh>
#include <key.hh>

#include "ArgsParser.h"
#include "UTreeWidgetItem.h"
#include "EntryForm.h"
#include "GroupForm.h"

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

using namespace keepass;
using namespace std;

enum FileType
{
    NoFile = 0,
    Kdbx,
    Kdb,
```

```
    ErrorType
};

typedef struct EntryInfo
{
    std::string title;
    std::string user_name;
    std::string password;
    std::string url;
    std::string notes;
    int icon;

} EntryInfo;

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow( int argc, char* argv[], QClipboard *clipboard, QWidget *parent =
nullptr);
    ~MainWindow( );

    void fillTreeViev( UTreeWidgetItem *parent = nullptr,
                const std::vector<std::shared_ptr<Group>>* groups = nullptr );

private slots:
    void on_actionOpen_File_triggered();
    void on_actionNew_File_triggered();
    void on_actionAdd_Group_triggered();
    void on_actionAdd_Entry_triggered();

private slots:
    void on_treeWidget_itemDoubleClicked(QTreeWidgetItem *item, int column);
    void on_passList_itemDoubleClicked(QTreeWidgetItem *item, int column);
    void on_treeWidget_itemClicked( QTreeWidgetItem *item, int column );

private:


private:
    Ui::MainWindow *ui;
    std::unique_ptr< Database > dt;
```

```
    ArgsParser args;
    FileType type;

    std::array<uint8_t, 16> current_uuid;

    QClipboard *m_clipboard;

    EntryForm *m_entry_form;
    GroupForm *m_group_form;

    friend class Form;
    friend class EntryForm;
    friend class GroupForm;
    friend class IconChooser;
};
#endif // MAINWINDOW_H
```

ДОДАТОК Д

Лістінг UTreeWidgetItem

```
#pragma once

#include <QTreeWidgetItem>

#include <array>
#include <memory>

#include <group.hh>

using namespace keepass;

class UTreeWidgetItem : public QTreeWidgetItem
{
public:
    UTreeWidgetItem( QTreeWidget *parent );
    UTreeWidgetItem( QTreeWidgetItem *parent );

    void setEntry( std::shared_ptr<Entry> entry );
    std::shared_ptr<Entry> getEntry( );

    void setGroup( std::shared_ptr<Group> group );
    std::shared_ptr<Group> getGroup( );

private:
    std::shared_ptr<Group> m_group;
    std::shared_ptr<Entry> m_entry;
};
```

ДОДАТОК Ж

Лістінг IconChooser

```
#ifndef ICONCHOOSER_H
#define ICONCHOOSER_H

#include <QDialog>
#include <QPaintEvent>
#include <QPainter>
#include <QHBoxLayout>

#include "EntryForm.h"
#include "mainwindow.h"

class IconContainer : public QWidget
{
   Q_OBJECT

public:
   IconContainer( IconChooser *parent,
                  std::string path );

   void paintEvent(QPaintEvent  *event) override;
   void mousePressEvent(QMouseEvent *event) override;

private:
   std::string path_to_icons;
   IconChooser *m_parent;
};

namespace Ui {
class IconChooser;
}

class IconChooser : public QDialog
{
   Q_OBJECT

public:
   explicit IconChooser( Form *main,
```

```
                    QWidget *parent = nullptr );
    ~IconChooser( );

private:
    Ui::IconChooser *ui;
    Form *m_form;
    IconContainer *m_icon_container;

    friend class IconContainer;
};

#endif // ICONCHOOSER_H
```

ДОДАТОК З

Лістинг шифрування

```
src.seekg(0, std::ios::beg);
  std::vector<char> header_data;
  header_data.resize(header_end);
  src.read(header_data.data(), header_end);

  std::array<uint8_t,  32> header_hash;
  SHA256_CTX sha256;
  SHA256_Init(&sha256);
  SHA256_Update(&sha256, header_data.data(), header_data.size());
  SHA256_Final(header_hash.data(),  &sha256);

  // Produce the final key used for encrypting the contents.
  std::array<uint8_t,  32> transformed_key = key.Transform(
      db->transform_seed(), db->transform_rounds(),
      Key::SubKeyResolution::kHashSubKeys);
  std::array<uint8_t,  32> final_key;

  SHA256_Init(&sha256);
  SHA256_Update(&sha256, db->master_seed().data(), db->master_seed().size());
  SHA256_Update(&sha256, transformed_key.data(), transformed_key.size());
  SHA256_Final(final_key.data(),  &sha256);

  std::unique_ptr<Cipher<16>> cipher;
  switch (db->cipher()) {
    case Database::Cipher::kAes:
      cipher.reset(new AesCipher(final_key,  db->init_vector()));
      break;
    case Database::Cipher::kTwofish:
      cipher.reset(new TwofishCipher(final_key,  db->init_vector()));
      break;
    default:
      assert(false);
      break;
  }

  // Decrypt the content.
  std::stringstream content;
```

```
  try {
    decrypt_cbc(src, content, *cipher);
  } catch (std::exception& e) {
    throw PasswordError();
  }

  std::array<uint8_t, 32> content_start_bytes_tst;
  content.read(reinterpret_cast<char*>(content_start_bytes_tst.data()),
               content_start_bytes_tst.size());
  if (!content.good() || content_start_bytes != content_start_bytes_tst)
    throw PasswordError();

  // Prepare deobfuscation stream.
  std::array<uint8_t, 32> final_inner_random_stream_key;
  SHA256_Init(&sha256);
  SHA256_Update(&sha256,
                db->inner_random_stream_key().data(),
                db->inner_random_stream_key().size());
  SHA256_Final(final_inner_random_stream_key.data(), &sha256);
  RandomObfuscator obfuscator(final_inner_random_stream_key,
                              kKdbxInnerRandomStreamInitVec);

  // Parse XML content.
  hashed_istreambuf hashed_streambuf(content);
  std::istream hashed_stream(&hashed_streambuf);

  if (db->compress()) {
    gzip_istreambuf gzip_streambuf(hashed_stream);
    std::istream gzip_stream(&gzip_streambuf);

    ParseXml(gzip_stream, obfuscator, *db.get());
  } else {
    ParseXml(hashed_stream, obfuscator, *db.get());
  }

  // Validate header hash.
  if (header_hash_ != header_hash)
    throw FormatError("Header checksum error in KDBX.");

  return db;
}
```

ДОДАТОК К

Лістінг згенерованого майстер-ключ шрифта

```
std::array<uint8_t,32>Key::Transform(const std::array<uint8_t, 32>& seed,
                                     const uint64_t rounds,
                                     SubKeyResolution resolution) const {
  AesCipher cipher(seed);
  std::array<uint8_t, 32> transformed_key = key_.Resolve(resolution);
  for (uint32_t i = 0; i < rounds; ++i)
transformed_key = encrypt_ecb(transformed_key, cipher);
  SHA256_CTX sha256;
  SHA256_Init(&sha256);
  SHA256_Update(&sha256, transformed_key.data(), transformed_key.size());
  SHA256_Final(transformed_key.data(), &sha256);
  return transformed_key;
}
```

ДОДАТОК Л

Лістінг розшифровування даних

```
std::unique_ptr<Database> KdbxFile::Import(const std::string& path,
                                           const Key& key) {
  Reset();
  std::ifstream src(path, std::ios::binary);
  if (!src.is_open())
throw FileNotFoundError();
  // Read header.
  KdbxHeader header;
  try {
header = consume<KdbxHeader>(src);
  } catch (std::exception& e) {
throw FormatError("Not a KDBX database.");
  }
  if (header.signature0  != kKdbxSignature0 ||
      header.signature1 != kKdbxSignature1) {
throw FormatError("Not a KDBX database.");
  }
  uint32_t kdb_ver =
      header.version & kKdbxVersionCriticalMask;
  uint32_t req_ver =
      kKdbxVersionCriticalMin  & kKdbxVersionCriticalMask;
  if (kdb_ver > req_ver) {
throw FormatError(
          Format() << "KDBX version " << header.version << " is not supported.");
  }
```

```
  std::array<uint8_t, 32> content_start_bytes = { { 0 } };
  std::unique_ptr<Database> db(new Database());
  // Read header fields.
  bool done = false;
  while (!done && src.good()) {
KdbxHeaderField header_field = consume<KdbxHeaderField>(src);
// Read the header field into a separate buffer before parsing. This is to
// guard against reading outside the field as well as for making sure to
// read the complete field regardless of how much of it that we parse.
    std::stringstream field;
    std::generate_n(std::ostreambuf_iterator<char>(field),
                    header_field.size,
                    [&src]() { return src.get(); });
if (!src.good())
        throw IoError("Read error.");
    assert(field.str().size() == header_field.size);
switch (header_field.id) {
        case KdbxHeaderField::kEndOfHeader:
        done = true;
        break;
        case KdbxHeaderField::kCipherId:
        if (consume<std::array<uint8_t, 16>>(field) != kKdbxCipherAes)
        throw FormatError("Unknown cipher in KDBX.");
       db->set_cipher(Database::Cipher::kAes);
        break;
        case KdbxHeaderField::kCompressionFlags: {
        uint32_t comp_flags = consume<uint32_t>(field);
        if (comp_flags > static_cast<uint32_t>(kKdbxCompressionFlags::kCount))
        throw FormatError("Unknown compression method in KDBX.");
       db->set_compress(comp_flags ==
```

```
    static_cast<uint32_t>(kKdbxCompressionFlags::kGzip));
  break;
  }
  case KdbxHeaderField::kMasterSeed:
db->set_master_seed(consume<std::vector<uint8_t>>(field));
  break;
  case KdbxHeaderField::kTransformSeed:
  if (header_field.size != 32)
  throw FormatError("Illegal  transform seed size in KDBX.");
db->set_transform_seed(consume<std::array<uint8_t, 32>>(field));
  break;
  case KdbxHeaderField::kTransformRounds:
db->set_transform_rounds(consume<uint64_t>(field));
  break;
  case KdbxHeaderField::kExcryptionInitVec:
  if (header_field.size != 16)
  throw FormatError("Illegal  initialization  vector size in KDBX.");
db->set_init_vector(consume<std::array<uint8_t, 16>>(field));
  break;
  case KdbxHeaderField::kInnerRandomStreamKey:
  if (header_field.size != 32)
  throw FormatError("Illegal  protected stream key size in KDBX.");
db->set_inner_random_stream_key(
    consume<std::array<uint8_t, 32>>(field));
  break;
  case KdbxHeaderField::kContentStreamStartBytes:
  if (header_field.size != 32)
  throw FormatError("Illegal  stream start sequence size in KDBX.");
   content_start_bytes = consume<std::array<uint8_t, 32>>(field);
  break;
```

```
        case KdbxHeaderField::kInnerRandomStreamId: {
        uint32_t inner_random_stream_id = consume<uint32_t>(field);
        if (inner_random_stream_id !=
        static_cast<uint32_t>(kKdbxRandomStream::kSalsa20)) {
        throw FormatError("Unknown random stream in KDBX.");
        }
        break;
        }
        default:
        throw FormatError("Illegal header field in KDBX.");
        break;
}
 }
 // Compute the header hash.
 std::streampos header_end = src.tellg();
```