**Міністерство освіти і науки України**
**Тернопільський національний технічний університет імені Івана Пулюя**

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету )
Кафедра кібербезпеки
(повна назва кафедри)

# КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(назва освітнього ступеня)

на тему: Розробка спеціалізованої системи керування паролями з метою підвищення ефективності функціонування та безпеки підприємств

Виконав(ла): студент(ка) 6 курсу, групи СБм-61
спеціальності 125 Кібербезпека
(шифр і назва спеціальності)

(підпис) Ревнюк О.А. (прізвище та ініціали)

Керівник (підпис) Александр М.Б (прізвище та ініціали)

Нормоконтроль (підпис) Лобур Т.Б. (прізвище та ініціали)

Завідувач кафедри (підпис) Загородна Н.В. (прізвище та ініціали)

Рецензент (підпис) (прізвище та ініціали)

Тернопіль
2020

Міністерство освіти і науки України
**Тернопільський національний технічний університет імені Івана Пулюя**

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра кібербезпеки
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Загородна Н.В
(підпис) (прізвище та ініціали)
« » 20___ р.

# ЗАВДАННЯ
## НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Магістр
(назва освітнього ступеня)

за спеціальністю _____
(шифр і назва спеціальності)

студенту _____
(прізвище, ім'я, по батькові)

1. Тема роботи _____

Керівник роботи _____
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «___» _____ 20___ року № _____

2. Термін подання студентом завершеної роботи _____

3. Вихідні дані до роботи _____

4. Зміст роботи (перелік питань, які потрібно розробити)

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)

6. Консультанти розділів роботи

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| Охорона праці | | | |
| Безпека в надзвичайних ситуаціях | | | |

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів роботи | Термін виконання етапів роботи | Примітка |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

Студент _____ _____
(підпис) (прізвище та ініціали)

Керівник роботи _____ _____
(підпис) (прізвище та ініціали)

# АНОТАЦІЯ

Розробка спеціалізованої системи керування паролями з метою підвищення ефективності функціонування та безпеки підприємств // Дипломна робота ОР «Магістр» // Ревнюк Олександр Андрійович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра кібербезпеки, група СБм-61 // Тернопіль, 2020 // С. 93 , рис. 17 , табл. – , кресл. – , додат. 4 .

Ключові слова: безпека облікових записів, менеджер паролів, система керування доступом.

Магістерська робота охоплює тему безпеки облікових записів та доступом працівників організації до паролів підприємства. В першому розділі здійснено аналіз безпеки існуючих менеджерів паролів. В другому розділі було проведено дослідження можливих вразливостей для створеної системи, а також методики безпечного шифрування паролів з можливістю зворотнього розшифрування. В третьому розділі розроблено систему керування паролями, яка може бути використана як для персональних потреб, так і для підвищення ефективності керування обліковими записами та їх захисту на підприємстві. Забезпечено два способи формування ключа для шифрування паролів, щоб передбачити зручність та безпеку використання програмного забезпечення і пришвидшення роботи з обліковими записами. В четвертому розділі зроблено огляд охорони праці в галузі розробки програмного забезпечення та безпеку в надзвичайних ситуаціях.

# ANNOTATION

Development of a specialized password management system to improve the efficiency of enterprise functioning and security// Thesis of the Master degree // Revnuk Oleksandr Andriyovich // Ternopil Ivan Puluj National Technical University, Department of Computer Information Systems and Software Engineering, Department of Cybersecurity // Ternopil, 2020 // P. 93, Fig. 17, Tables – , Diagrams. -, Annexes. -, References. 4.

Keywords: account security, password manager, access control system.

The master's thesis covers the topic of account security and access of employees of the organization to the passwords of the enterprise. The first section analyzes the security of existing password managers. In the second section, a study of possible vulnerabilities for the created system, as well as methods of secure encryption of passwords with the possibility of reverse decryption. The third section develops a password management system for personal use and use in the enterprise. There are two ways to generate a key for encrypting passwords to provide convenience and security of software use and speed up work with accounts. The fourth section provides an overview of occupational safety and health in the field of software development and safety in emergencies.

# СПИСОК СКОРОЧЕНЬ

API (Application Programming Interface) – прикладний програмний інтерфейс

CSP (Content Security Policy) - проект стандарту комп'ютерної безпеки для боротьби з XSS атаками

CSRF (Cross-Site Request Forgery) - міжсайтова підробка запиту

DCL (Data Control Language ) - комп'ютерна мова, також частина SQL, що використовуються в комп'ютерних програмах або користувачами баз даних для контролю доступу до даних в базах даних.

DDL (Data Definition Language) - сімейство комп'ютерних мов, що використовуються в комп'ютерних програмах або користувачами баз даних для опису структури даних.

DML (Data Manipulation Language) - сімейство комп'ютерних мов, що використовуються в комп'ютерних програмах або користувачами баз даних для отримання, вставки, видалення або зміни даних в базах даних.

MITM (Man-in-the-middle) - атака «Людина посередині

OTP – одноразовий пароль

XSS (Cross Site Scripting ) - міжсайтовий скриптинг

ПЗ – програмне забезпечення

# ЗМІСТ

ВСТУП ..... 9

РОЗДІЛ 1. ВИЗНАЧЕННЯ РІВНЯ ЗАХИЩЕНОСТІ ПАРОЛІВ В ПРОГРАМНОМУ ЗАБЕЗПЕЧЕННІ ..... 12

1.1. Аналіз поняття «сильного» пароля ..... 12

1.2. Методи побудови захищеного пароля для корпоративного програмного забезпечення ..... 18

1.3. Порівняльний аналіз існуючих менеджерів паролів ..... 21

1.4 Висновки до першого розділу ..... 24

РОЗДІЛ 2. ДОСЛІДЖЕННЯ ВРАЗЛИВОСТЕЙ МЕНЕДЖЕРІВ ПАРОЛІВ ДЛЯ ВИБОРУ АЛГОРИТМУ ШИФРУВАННЯ ..... 25

2.1. Дослідження варіативних вразливостей менеджерів паролів ..... 25

2.2. Дослідження алгоритму шифрування інформації в програмному забезпеченні ..... 33

2.3. Дослідження стратегії безпечного управління інформацією в базі даних. 37

2.4. Висновки до другого розділу ..... 41

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ СПЕЦІАЛІЗОВАНОЇ СИСТЕМИ КЕРУВАННЯ ПАРОЛЯМИ ..... 43

3.1. Основні відомості про веб-розробку ..... 43

3.2. Вибір засобів для створення програмного забезпечення ..... 45

3.3. Специфікації вимог до програмного забезпечення ..... 47

3.4. Розробка захищеної системи керування паролями ..... 50

3.5. Висновки до третього розділу ..... 57

РОЗДІЛ 4. ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ ..... 58

4.1. Охорона праці ..... 58
4.2. Безпека в надзвичайних ситуаціях ..... 61
ВИСНОВКИ ..... 64
СПИСОК ЛІТЕРАТУРИ ..... 66
ДОДАТКИ ..... 69
Додаток А ..... 70
Додаток Б ..... 72
Додаток В ..... 80
Додаток Г ..... 81

# ВСТУП

**Актуальність** захисту персональних даних в мережі Інтернет зростає з кожним створеним обліковим записом користувачем. Сьогодні, під час використання Інтернету, кожна людина працює під своїм обліковим записом. Інтернет магазини, банківські сервіси, розважальні портали та багато іншого містять особистий кабінет з авторизацією. У свою чергу, користувачі повинні кожен раз, під час створення облікового запису, вигадати новий пароль та запам’ятати його.

Користувачі звикли думати, що для того, щоб надійно захистити свої персональні дані паролем - останній повинен бути незрозумілим рядком, що складається із випадкових символів, великих та малих літер, цифр. Але більшість людей – не задумуються методами зберігання паролів і використовують одноманітні слабкі паролі, повторяючи їх на різноманітних ресурсах. А все тому, що запам’ятати один складний пароль – не така легка задача в повсякденному житті. Простіше запам’ятати те, що оточує людину або фразу, яка асоціюється з предметами із повсякденного життя. Це можуть бути герої фільмів або відео-ігор з використанням додаткових цифр, клички тварин, дати народжень, назви вулиць, міст і т.д… Таку інформацію не потрібно запам’ятовувати і тому такі комбінації використовував кожен, хоча б один раз в житті. Зловмисники – теж люди і вони розуміють цю вразливість краще аніж хто інший.

Використання менеджера паролів значно спрощує життя, вирішуючи вище згадані проблеми, оскільки замість того, щоб пам’ятати безліч паролів - потрібно пам’ятати лише один ключ, а все інше робить система. Така система здатна генерувати та запам'ятовувати захищені паролі. Це означає, що довші паролі використовують комбінацію літер, цифр та символів, уникаючи впізнаваних слів. У свою чергу, будь-хто, хто намагається «зламати» пароль, займе для цього експоненціально більше часу. Вони також не зможуть скористатися словниковою атакою, яка має на меті скоротити процес,

намагаючись спробувати слова, а не випадкові рядки символів. Багато менеджерів паролів дозволяють легко встановити відповідні критерії захищеного пароля, перш ніж створювати пароль.

Більшість менеджерів паролів розраховані на роботу з обліковими даними одного користувача. Власникам підприємств потрібно використовувати спеціалізовані засоби, щоб точно контролювати доступ працівників до різних корпоративних послуг. Таким чином виникає багато загроз під час передачі ключів та паролів доступу працівникам. Спеціалізована система керування паролями дасть можливість власникам організацій самостійно здійснювати контроль доступу до корпоративних паролів тими чи іншими працівниками. Розробка програмного забезпечення, яке дає можливість не тільки розподіляти зашифровані паролі організації між користувачами – але й якісно захищати від впливу сторонніх чинників – основна ідея роботи. Система буде корисною, як для власного користування, так і для організацій.

**Метою** дослідження є: розробка захищеної системи керування паролями, яка підвищить ефективність функціонування та безпеку особистих даних в організації.

**Об’єктом** дослідження є управління безпекою паролів на підприємстві та керування доступом працівників до них

**Предметом** дослідження є програмне забезпечення для автоматизованого управління паролями і оцінкою рівня безпеки та керування доступом до корпоративних паролів.

Для досягнення поставленої мети, потрібно вирішити такі завдання:

1. Провести аналіз різних типів вразливостей існуючих менеджерів паролів.
2. Дослідити можливі варіанти безпечного зберігання паролів в базі даних.
3. Вибрати надійний метод шифрування із можливістю зворотнього розшифрування.

4. Розробити програмне забезпечення з надійними методи валідації запитів, що надходять на сервер, прогресивним генератором паролів та оцінки їхньої безпеки, системою контролем доступу користувачів до ресурсів в організації.

У рамках даного дослідження будуть використані наступні **методи**: аналіз, порівняння – для вивчення наукової літератури, системний і порівняльний аналіз – для дослідження вразливостей існуючих менеджерів паролів.

**Наукова новизна** дослідження: розроблене програмне забезпечення має спеціалізований функціонал для використання в організаціях з передбаченим механізмом розподілу ролей працівників та можливістю вибору формування ключа, що дозволяє підвищити ефективність захисту та безпеки облікових записів на підприємстві.

**Практичне значення** дослідження: безпечна система керування паролями, яка дає можливість користування як звичайному користувачеві, так і власнику підприємства із функціоналом керування доступом працівників до паролів організації.

Результати роботи апробовані на VIII науково-технічній конференції «Інформаційні моделі, системи та технології» Тернопільського національного технічного університету імені Івана Пулюя.

Пояснювальна записка обсягом 93 сторінок містить: 17 рисунків, 4 додатки. Список використаних джерел розміщується на 2 сторінках і містить 17 джерел.

# РОЗДІЛ 1. ВИЗНАЧЕННЯ РІВНЯ ЗАХИЩЕНОСТІ ПАРОЛІВ В ПРОГРАМНОМУ ЗАБЕЗПЕЧЕННІ

## 1.1. Аналіз поняття «сильного» пароля

Аутентифікація - одна з найважливіших сфер комп’ютерної безпеки яка передбачає використання традиційних текстових паролів. Однак цей тип аутентифікації має недоліки. Різні альтернативні схеми автентифікації, які спрямовані на узгодження безпеки та зручності використання варіюються від графічної автентифікації пароля до автентифікації на основі місцезнаходження Однак жодна з цих схем не змогли подолати простоту та доступність введення послідовності символів клавіатури, щоб дозволити автентифікацію користувачів в системі. Як результат, традиційні текстові паролі все ще залишаються найпопулярнішим механізмом автентифікації в Інтернеті, ймовірно залишаться таким найближчим часом.

На жаль, з точки зору юзабіліті, текстовий пароль є досить проблематичним. Потрібний хороший пароль, щоб було «легко запам’ятовувати і важко вгадувати» одночасно. Однак паролі, які легко запам’ятати, як правило, короткі або базуються на словникових словах (або невеликі варіації). Тому ці паролі стають вразливими до атак на словники. Паролі, в тому числі особиста інформація також запам'ятовуються, але вони ризиковані і можуть бути вгаданими люди, близькими до власника пароля і зловмисниками, які зібрали інформацію про користувача. Паролі вважаються одним із найбільш значущих ризиків з точки зору безпеки в інформаційних системах, які є вразливі до атак. Ця вразливість в основному обумовлена поведінкою та практикою користувачів і не пов’язані із самою системою. Такі ризики спричиняють інші проблеми, пов'язані з такими факторами, як повторне використання, спільний доступ та вибір слабких паролів

Ці проблеми добре відомі і дослідники їх називають «проблемами людського фактору». Більшість людей насправді усвідомлюють важливість,

вибираючи надійні паролі для захисту своєї інформації. Однак більшості бракує порад щодо створення паролів або вказівок, які б допомогли або мотивували їх до створення надійних паролів. Прийнято стверджувати, що пароль повинен містити поєднання символів клавіатури та не повинен містити значущих слів зі словників. На жаль, незважаючи на ці вказівки та поради, більшість користувачів не застосовують належні способи безпеки та, вибирають слабкі паролі.

Щоб збільшити стійкість вибраних паролів, користувачі мають дотримуватись набору правил, відомих як рекомендації щодо паролів, під час їх створення. Наприклад, пароль повинен містити принаймні вісім символів, включаючи принаймні одне число або одну велику буква, і він не повинен містити ім’я користувача. Згідно з дослідженням, користувач в середньому використовує вісім паролів на день.

Відповідно до оновлених рекомендацій, як згадувалось раніше, паролі, генеровані користувачами повинні мати принаймні 8 символів. Механізм автентифікації повинен дозволяти користувачам використовувати паролі та перефразовані вирази (довжиною до 64 символів) і заборонити паролі в чорному списку, які були скомпрометовані раніше. Існує більше нововведень, таких як можливість копіювання та відображення функції паролів, заборона на підказки щодо пароля та усунення терміну дії пароля без причини.

Політика обмеження паролів - це ряд правил, що визначають зміст і формат прийнятих паролів в системі автентифікації. Ці політики використовуються групою адміністраторів для підвищення рівня комп'ютерної безпеки шляхом керівництва користувачами для створення більш безпечних паролів. У 2006 році Національний інститут стандартів і технологій (NIST) оновив «Керівні принципи електронної автентифікації» для використання адміністраторами систем безпеки впровадження електронної автентифікації. Ці принципи забезпечують евристику для вимірювання сили та ефективності політики пароля з урахуванням бітів ентропії для визначення значення пароля. Командурі та ін. провели велике онлайн-дослідження для порівняння чотирьох

різних політик обмеження паролів. Вони дослідили, що користувачам склало менше труднощів у створюванні 16-ти символьних паролів у порівнянні з 8-символьними, за винятком словникових слів або подальших обмежень. Крім того, паролі принаймні з 16 символів надають найкращу безпеку. Вони також виміряли надійність пароля використовуючи ентропію розрахунку і таким чином, показали деякі помилкові уявлення про вплив політики обмеження для паролів. З їхніх висновків випливає, що додавання цифр значно збільшило ентропію паролів, але без урахування словникових слів ентропія збільшилася менше ніж це було очікувано. Також висновки показали, що створені паролі користувачами ледве забезпечує мінімальні вимоги. Результати показують, що важко створювати та запам'ятовувати паролі користувачам, коли їх змушують застосовувати суворі та складні політики щодо паролів. Щоб впоратися із запам’ятовуванням складних паролів, користувачі зазвичай застосовують небезпечні стратегії щодо паролів. Ці правила також допомагають зловмисникам вгадувати паролі більш ефективно, оскільки вони можуть зменшити кількість можливих паролів на основі політики обмежень.

Принципи NIST були оновлені в 2017 році, наголошуючи на непотрібності суворих правил політики щодо створення надійних паролів. Тому було рекомендовано організаціям не вимагати від користувачів застосування цих правил. Звідси виходить, що довжина пароля впливає на надійність найбільше, тому пропонується використовувати довгі паролів та парольні фрази. Нова настанова має на меті спонукати користувачів створювати криптографічно надійні та пам’ятні паролі.

Більшість систем, які вводять обмеження щодо паролів, пропонують свої поради користувачеві щодо створення паролів. Метою порад щодо створення паролів є прийняття легшої політики правил, а також, одночасно спонукають користувачів створювати надійніші паролі. На більшості веб-сайтів поради щодо паролів були визнані неоднозначними та непотрібними користувачам. Також, було виявлено, що існують суттєві розбіжності між порадами, що використовуються в різних середовищах.

Намагаючись заохотити користувачів створювати легко запам'ятовувані паролі, мнемонічні паролі на основі фраз були вперше запропоновані Бартонами. Мнемонічні паролі походять від фрази, де користувачі зазвичай використовують букву кожного слова в реченні. Хоча існує багато досліджень щодо паролів стосуються мнемонічних паролів, їх рідко рекомендується використовувати на практиці .Багато досліджень представлені, де використовуються два мнемонічні методи генерації паролів в дослідженні користувачів, де усі користувачі обирають своє речення. Оскільки паролі, створені методом мнемонічних рядків, зазвичай мають більше символів, тому їх вважали більш безпечними. Однак автори виявили незначну різницю в часі створення паролів та часі входу і згадування частоти помилок між двома методами. Було виявлено, що паролі, які містять більше символів, більш стійкі до злому.

Поради щодо паролів також можна представити за допомогою інструменту, що вимірює надійність пароля та надає користувачам числові дані результату або фрази, такі як «слабкий», «сильний» та «дуже сильний». Ці інструменти називаються "вимірювачами сили пароля", які зазвичай ілюструють силу поточно вибраного пароля, коли користувач реєструє свій обліковий запис. Сервісом зазвичай користуються популярними веб-сайтами (Gmail, PayPal та eBay). В іншому дослідженні новий метод називається адаптивним паролем вимірювачі стійкості були запропоновані для вимірювання стійкості пароля. Використовуються адаптивні лічильники надійності пароля для вимірювання надійності як колективної ймовірності кожного символу, що слідує за попередніми символами в паролі. Ці ймовірності обчислюється на основі навчального набору паролів. Така модель краща, ніж будь-яка інша запропонована для надійності пароля на сьогоднішній день, оскільки вона може підібрати паролі ближче до «ідеального». Існує багато різних калькуляторів для оцінки кількості припущень, необхідних для злому пароля, використовуючи певний алгоритм злому. Вони підраховують відсоток паролів, які можна зламати за допомогою заданих рядів припущень.

Очевидно, що короткі або осмислені паролі легко запам'ятовуються людиною, але вони набагато простіші для розкриття. Використання довгих і безглуздих паролів безумовно краще з погляду криптостійкості, але людина звичайно не може їх запам'ятати і записує на папірці, що потім або губиться, або попадає в руки зловмисникові. Саме з того, що недосвідчені користувачі звичайно вибирають або короткі, або осмислені паролі, існують два методи їхнього розкриття: атака повним перебором і атака по словнику.

У зв'язку з різким ростом обчислювальних потужностей атаки повним перебором мають набагато більше шансів на успіх, ніж раніше. Крім того, активно використовуються розподілені обчислення, тобто рівномірний розподіл задачі на велику кількість машин, що працюють паралельно. Це дозволяє багаторазово скоротити час злому.

Тому при використанні аутентифікації на основі паролів захищеною системою повинні дотримуватися наступні правила:

- не дозволяються паролі менше 8 символів;
- паролі повинні перевірятися відповідними контролерами;
- символи пароля при їхньому введенні не повинні з'являтися в явному виді;
- після введення правильного пароля видається інформація про останній вхід у систему;
- обмежується кількість спроб уведення пароля;
- вводиться затримка часу при неправильному паролі;
- при передачі по каналах зв'язку паролі повинні шифруватися;
- паролі повинні зберігатися в пам'яті тільки в зашифрованому вигляді у файлах, недоступних користувачам;
- користувач повинний мати можливість самому змінювати пароль;
- адміністратор не повинний знати паролі користувачів, хоча може їх змінювати;
- паролі повинні періодично мінятися;

- установлюються терміни дії паролів, після закінчення яких треба зв'язатися з адміністратором.

## 1.2. Методи побудови захищеного пароля для корпоративного програмного забезпечення

Конфіденційність є ключовим елементом інформаційної безпеки, а головним механізмом виступає автентифікація користувачів. Процедури автентифікації традиційно розділяються на два різні етапи. Ідентифікація користувача (User ID) спочатку підтверджує людину, яка взаємодіє з системою. Оскільки це лише засіб уточнення - цей ідентифікатор не повинен бути захищений. На другому етапі - автентифікації користувача - користувач повинен бути підтверджений як законний власник ідентифікатора, а пароль використовуваний як засіб автентифікації повинен бути секретним.

Спочатку паролі генерувалися системою, щоб гарантувати, що користувачі використовують «безпечні» комбінації символів. Однак більшості користувачів важко запам’ятати ці паролі, і тому вони, як правило, записують їх. Крім того, при генерації були виявлені ризики безпеки пароля. Обидві ці причини призвели до того, що паролі, створені користувачами, є найбільш широко використовуваними в процесі авторизації . Окрім паролів з одним словом, існує ряд інших механізмів автентифікації, які зараз використовуються:

- Парольні фрази (потрібна фраза замість слова).
- Когнітивні паролі (ряд запитань та відповідей щодо особистих даних).
- Асоціативні паролі (ряд слів та асоціацій).
- Персональні ідентифікаційні номери (PIN-коди).

Рівень безпеки, який забезпечується цими механізмами, може сильно відрізнятися залежно від рівня користувача, його досвіду у створенні паролів та обізнаності щодо безпеки. Федеральні стандарти обробки інформації США (FIPS, 1985) припускають, що існує кілька критеріїв, які слід використовувати для забезпечення різних рівнів захисту паролем.

Безпека визначається як зменшення несанкціонованого доступу до інформації або системи, а зручність використання пароля визначається з точки зору запам'ятовуваності та сприйняття користувачем.

Неодноразово встановлено, що 50% людей, які використовують більше одного пароля, створили метод побудови "пов'язаних" паролів, тобто всі або більшість їхніх паролів мали спільну тему або корінь. Вміст пароля повинен розглядатись з точки зору його запам'ятовуваності та безпеки. Це спонукає користувачів створювати правила та судження щодо стратегій проектування паролів. Користувачі хочуть створити безпечний пароль, а створюють небезпечний. Такі суперечності ускладнюють встановлення взаємозв'язку між чинниками, які впливають на поведінку користувачів. Суперечливі твердження можуть бути спричинені тим, що користувачі не впевнені у своїх власних діях. Приклад очевидного протиріччя наведено на рисунку 1.1.

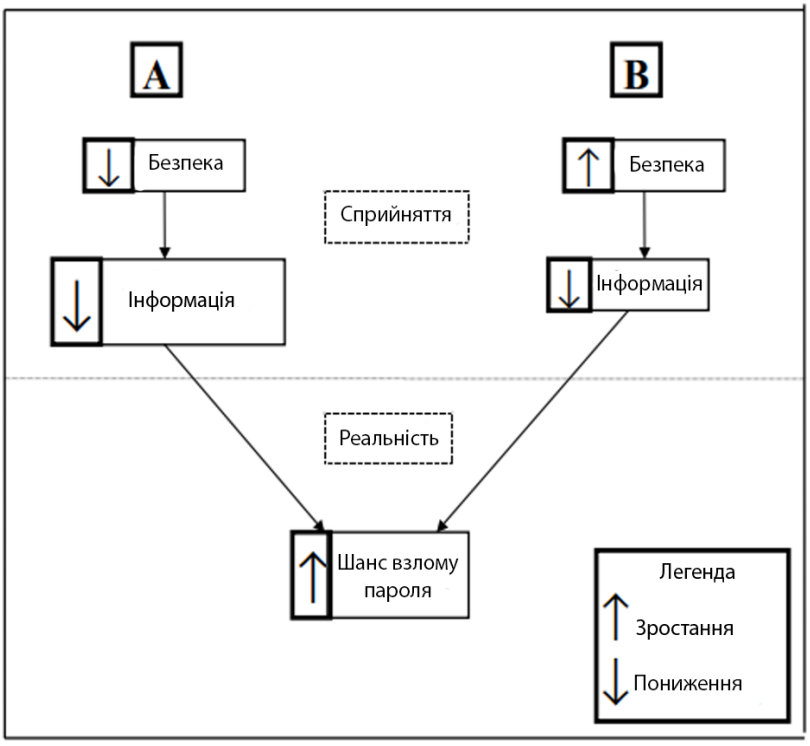



Рисунок 1.1 – Приклад протиріччя користувачів

Якщо користувачі сприймають загальний рівень безпеки організації як низький (знижений), це знижує їх рівень сприйняття того, наскільки чутливою є захищена інформація. Це, в свою чергу, збільшує можливість розкриття пароля. Якщо користувачі сприймають загальний рівень безпеки організації як високий (підвищений), це знижується загальне сприйняття ними загроз інформації. Це, в свою чергу, також збільшує невпевненість у роботі та

можливість розкриття пароля. Сумісність між робочою практикою та процедурою створення пароля є дуже важливою. Організація змушує користувачів мати індивідуальні паролі для групової роботи.

Основною доктриною безпеки паролів, прийнятою організаціями, є принцип знання. Припущення полягає в тому, що чим більше відомо про механізм безпеки, тим легше атакувати. Отже, частина захисту полягає у передачі інформації лише тим, хто «повинен знати». Такий підхід до безпеки застосовується багатьма діловими організаціями. Багато служб безпеки розглядають користувачів як «небезпечні», що створює тенденцію повідомляти користувачам якомога менше. Небезпечна поведінка користувачів часто спричинена нерозумінням. У багатьох організаціях системні паролі замінюються на створені користувачами. Це означає, що відповідальність за створення захищених паролів перекладена на користувачів, але політика «необхідності знати» багатьох підрозділів безпеки означає, що правила створення безпечного пароля рідко передаються користувачам. Користувачі сприймають загрози як низькі через відсутність видимих відгуків про ризики. У побудові захищених паролів для організації можуть допомогти рекомендації, наведені нижче.

• «Зміст пароля». Проведення онлайн інструктажів та тренінгів щодо побудови зручних та захищених паролів. Це покаже користувачам, що їм не потрібно обходити механізми безпеки для побудови не забутніх паролів. Це буде інформація користувачам про те, чого не можна використовувати в їхніх майбутніх паролях.

• «Зв'язок між паролем та кількома паролями». Кілька паролів знижують загальну безпеку та запам'ятовування, що, в свою чергу, збільшує ризик додаткових витрат. Якщо потрібно більше паролів, користувачі повинні бути проінформовані про те, що вони можуть створювати нові паролі без спільних коренів та слів

• «Сприйняття безпеки». Періодично інформувати користувачів про важливість безпеки паролів. Надсилання повідомлення користувачам про

можливі загрози для системи та інформації організації підвищить уявлення користувачів про сприйняті загрози.

• «Практичне застосування». Створення примусового використання паролів буде означати, що користувачі матимуть самі, наскільки безпека пароля відповідає їхнім очікуванням. Якщо очікування виявляється кращими - користувачі повинні мати можливість використати механізм побудови паролів.

## 1.3. Порівняльний аналіз існуючих менеджерів паролів

У вигляді поширення кібератак на злом пароля користувача приходиться використовувати складні паролі для аккаунтів, що створює складність їх запам’ятовувати для різних сайтів. Анонімізація, або редагування даних – процес видалення або приховування персональних даних з метою їх подальшого використання [1]. Необхідно аналізувати менеджери паролів для визначення ситуацій за захистом прихованої інформації .

За рекомендаціями пошуку Google для аналізу були вибрані 5 кращих менеджерів паролів:

- LastPass;
- Dashlane;
- Keeper;
- F-Secure Key Password Manager;
- 1Pasword.

*LastPass*. Рішення від розробників популярного розширення для десктопних браузерів, які раніше займалися зберіганням і синхронізацією паролів. Програмне забезпечення вміє зберігати в зашифроване відео паролі та інші дані (у платній версії), синхронізувати збережені дані між різними пристроями і браузерами, та має версії для всіх основних ОС та браузерів. Захист даних здійснюється на основі шифрування AES-256, всі дані зберігаються на сервері розробників та передачі даних на потрібні пристрої здійснюється після

запрошення на доступ до нього, що створює небезпеку перехоплення даних зловмисником.

Використовується однофакторна аутентифікація для входу в акаунт використовується мастер-пароль, кількість спроб введення пароля не обмежено, що створює небезпеку для здійснення грубої силу для атаки. Для додаткового захисту є можливість використовувати відпечаток пальця для входу в аккаунт, але його можна знайти і використовувати тільки, якщо є мастер-пароль.

*Dashlane*. Пропонує всі необхідні функції захисту даних. Програма підтримує автозаповнення паролів на веб-сайтах та в додатках, але не вміє відрізняти поля вводу пароля на різних піддоменах. Dashlane дозволяє віддалено змінити паролі у підтримуваних сайтах. Є можливість зробити резервну копію даних в хмарному сховищі. Доступ до зберігання здійснюється через додатки для Windows, MacOS, Android та iOS. Відсутня українська мова в інтерфейсі. Платна версія. Захист даних здійснюється на основі шифрування AES-256, зашифровані дані зберігаються на пристрої. Використовується однофакторна аутентифікація для входу в акаунт з використанням мастер-пароля, кількість спроб введення пароля не обмежено. Для додаткового захисту є можливості використовувати відпечаток пальця для входу в аккаунт, але його можна знайти і використовувати тільки при наявності мастер-пароля.

*Keeper*. Менеджер паролів з більшим числом функцій. Приклад охоплює основні функції, включаючи в себе автоматичне заповнення в різних додатках та веб-сайти. Наряду з паролями, також включає в себе відео та фото зберігання, де можливо зберігати конфіденційне зображення або відео. Присутня синхронізація додатків між пристроями та хмарним сховищем. Також можливо зберігати дані у сховищі. Захист даних здійснюється на основі шифрування AES-256, зашифровані дані можна зберігати на використаному пристрої або в сховищі, що створює небезпеку перехвату даних. Є можливість скидання секретного пароля. Відсутній захист бази даних, що представляє загрозу ін'єкції даних в базі без якої-небудь аутентифікації.

F – Secure KEY Password Manager. Пропонує всі необхідні функції захисту даних. У безкоштовній версії відсутня синхронізація між пристроями та хмарним сховищем. Відсутня українська мова в інтерфейсі. Використовується однофакторна аутентифікація для входу в акаунт. Використовується мастер-пароль, кількість спроб введення пароля не обмежено. Також мастер-пароль зберігається у форматі відкритого тексту в локальній директорії пристрою.

*1Password*. Флагман на ринку зберігання паролів та іншої конфіденційної інформації. Вся компанія працює лише над одним продуктом і цей продукт вважається одним із еталонів захисту паролів. Підтримує MacOS, Windows, Android, iOS. Має інтуїтивний і зрозумілий інтерфейс, синхронізацію та потужний пошук серед продуктів даних. Підписка на сервіс коштує 7 доларів за рік, безкоштовна версія відсутня. Є велика ймовірність того, що при находженні вразливості в програмному коді технології, вона не буде виправлена оперативно і зловмисники не зможуть використовувати їх. Використовується однофакторна аутентифікація для входу в акаунт та мастер-пароль, кількість спроб введення пароля не обмежена. Для додаткового захисту аккаунту можна використовувати відпечаток пальця.

На основі аналізу менеджерів паролів можливо визначити недоліки в захищеній системі для зберігання інформації:

1. Присутня синхронізація додатків між пристроями і хмарним зберіганням даних, що створює небезпеку перехватити даних
2. Використовується однофакторна аутентифікація для входу в акаунт, кількість спроб введення пароля не обмежено;
3. Можливості безкоштовних версій дуже обмежені, повний доступно всіх можливостей можна отримати тільки в платних версіях;
4. Кілька менеджерів паролів зберігають мастер-пароль у форматі відкритого тексту в локальних каталогах, що створює небезпечність виявлення пароля;
5. У менеджері пароля є можливість скинути секретний пароль і отримати всю подальшу аутентифікацію;

6. В менеджері паролів відсутній захист бази даних, що представляє загрозу ін’єкції даних в базі даних без будь-якої аутентифікації.

### 1.4 Висновки до першого розділу

Хоча використання паролів як методу автентифікації було широко вивчено в минулому, немає емпіричних досліджень, які перевіряли б ефективність методів створення паролів. Таким чином, зобов’язувати користувачів дотримуватися суворих правил політики щодо паролів, мотивуючи і спрямовуючи їх на створення надійних і незабутніх паролів є найбільш ефективним та придатним для використання способом. Настанову щодо правил створення пароля можна завжди вдосконалити, додавши різні елементи до неї. Часовий тиск - ще один фактор, який слід враховувати [2]. Слід змінювати паролі щонайменше один раз на місяць.

# РОЗДІЛ 2. ДОСЛІДЖЕННЯ ВРАЗЛИВОСТЕЙ МЕНЕДЖЕРІВ ПАРОЛІВ ДЛЯ ВИБОРУ АЛГОРИТМУ ШИФРУВАННЯ

## 2.1. Дослідження варіативних вразливостей менеджерів паролів

Безпечний менеджер паролів може автоматично генерувати паролі для веб-сайтів, звільняючи користувачів від когнітивного тягаря запам’ятовування. Додаючи хмарну синхронізацію між пристроями, і менеджерами паролів, розробники обіцяють величезні переваги безпеки та зручності при мінімальній можливості розгортання та витрат. Ідеалізовані менеджери паролів, що надають масу переваг, можуть мати недоліки, що перекреслять всі переваги нахваленого в мережі користувачами та рекламою програмного забезпечення. Широке поширення небезпечних менеджерів паролів може погіршити ситуацію безпеки користувачів в цілому. При оцінці готовності вразливості, через яку може реалізуватися загроза, враховуються зручність (можливість) використання вразливості джерелом загроз, складність використання, необхідні кошти, можливість застосування неспеціалізованої апаратури [3]. Зрештою, вразливість в таких менеджерах може дозволити зловмиснику викрасти всі паролі користувача одиноким махом. Мільйони користувачів довіряють цим вразливим менеджерам паролів та безпечно зберігати їхню секретну інформацію для доступу до різних джерел.

По суті, менеджер паролів це база даних, яка зберігає паролі та імена користувачів на різних ресурсах. Програмне забезпечення контролює доступ до цієї бази даних через головне ім'я користувача чи пароль. Менеджер паролів має базу даних облікових даних користувача в різних веб-програмах. Веб-програма - це веб-сайт, який автентифікує своїх користувачів, запитуючи комбінацію ім'я користувача та пароль. "Точка входу" веб-додатку - це сторінка, на якій користувач може вводити свої дані. Людина може зберігати кілька облікових даних для одного і того ж сайту. Рисунок 2 ілюструє загальну схему того, як користувач використовує менеджер паролів для того, щоб

пройти аутентифікацію системи. Спершу людина входить в менеджер паролів, використовуючи її ім’я та пароль. Потім користувач отримує його дані для потрібного акаунту. Після того він використовує свої ключі для входу. Оскільки отримання та надсилання облікових даних вручну є не найшвидшим процесом, менеджери паролів можуть також автоматизувати процес вибору відповідних облікових даних та вхід у відкриту веб-програму. Це може включати навігація веб-браузера для точки входу, заповнення деяких текстових полів з іменем користувача та паролем для відправлення форми логіну. Оскільки ці завдання передбачають виконання коду всередині веб-програми, менеджери паролів часто покладаються на привілейоване розширення браузера.

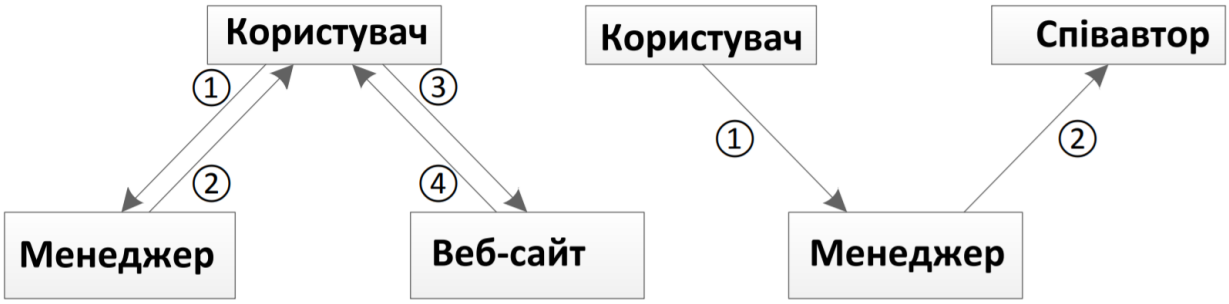



Рисунок 2.1 – Схема використання менеджера пароля

Сучасні менеджери паролів включають можливість ділитися паролями зі співавтором. Рисунок 2.1 показує загальну схему того, як користувач ділиться своїми повноваженнями зі співавтором. Користувач просить спільного доступу менеджера паролів до облікових даних для співавтора. Програмне забезпечення пересилає облікові дані співавтору, коли він просить це.

Через особливо інформацію, яка циркулює в менеджерах паролів, розробники прагнуть мінімізувати кількість коду та персоналу з доступом до облікових даних програмного забезпечення. Одним із поширених методів є шифрування облікових записів в базі даних, таким чином запобігаючи злом на стороні сервера та доступ до відкритого вигляду тексту. Шифрування облікових даних відбувється із використанням функції введення ключа.

Наприклад, LastPass використовує JavaScript для дешифрування / шифрування облікових даних користувача і база даних отримує ключ від головного імені користувача та пароля користувача.

Висока кваліфікація кіберзлочинців дозволяє створювати та використовувати унікальні, ще не відомі IT-індустрії шкідливі програми, знаходити нові вразливості в програмних продуктах і використовувати їх для проведення комплексних кібератак [4]. Тому, щоб оцінити безпеку сучасних менеджерів паролів, було обрано вибірку із п'яти кращих менеджерів паролів, що підтримують різноманітну суміш функцій. На рисунку 2.2, наведено огляд їх особливостей. Стовпці «Розширення» та «Закладка» вказують на підтримку автоматизації входу через певний механізм; «Веб-сайт» означає наявність веб-сторінки. Для менеджерів паролів, що підтримують шифрування облікових даних також перелічено їх виведення ключів.

| | Закладки | Розширення | Вебсайт | Шифрування | | Співвласники |
|---|---|---|---|---|---|---|
| | | | | Мастер-пароль | Зашифровані поля | |
| LastPass | ✓ | ✓ | ✓ | KDF($mp$,$mu$,5000,32) | логін та пароль | ✓ |
| RoboForm | ✓ | ✓ | ✓ | × | | × |
| My1login | ✓ | × | ✓ | MD5($ph_{even}$)+MD5($ph_{odd}$) | логін та пароль | ✓ |
| PasswordBox | × | ✓ | × | KDF($mp$,$mu$,10000,32) | тільки пароль | ✓ |
| NeedMyPassword | × | × | ✓ | × | | × |

Рисунок 2.2 – Особливості менеджерів паролів

Головна модель загрози - веб-зловмисник. Зазвичай, веб-зловмисник контролює один або кілька веб-серверів і DNS доменів та може змусити жертву відвідати контрольовані домени створені зловмисником. Це ключова модель загроз для веб-менеджерів паролів, які часто працюють у браузері. Користувач може створити обліковий запис у веб-програмі зловмисника і використовувати менеджер паролів для управління своїм обліковим записом. Зловмисник також можете створювати облікові записи і робити запити безпосередньо до менеджера паролів. На вищому рівні менеджер паролів має лише один ключ

безпеки, що забезпечує доступ до збереженого паролю лише авторизованим користувачам. Безпека мастер ключа - це цілісність головного облікового запису. Зловмиснику повинно бути неможливо пройти автентифікацію користувача у менеджері паролів.

Основна задача менеджера паролів - надійне зберігання списку облікових даних користувача. Програмному забезпеченню потрібно відповідати за конфіденційність, цілісність, і доступність — бази даних. Зловмисник, не повинен бути в змозі дізнатися облікові дані, що дозволило б скомпроментувати користувача. Для забезпечення безпеки інформаційних систем застосовують системи захисту інформації, які є комплексом організаційно - технологічних, програмно-технічних засобів і правових норм, направлених на протидію джерелам загроз безпеці інформації [5].

Ключова відмінність між веб-менеджерами паролів та «локальними» менеджерами паролів полягає в їх необхідності роботи у веб-браузерах. Веб-менеджери паролів. Можна виділити чотири ключові проблеми для сучасних веб-менеджерів паролів: вразливості до закладок, класичні веб-вразливості, вразливості авторизації та вразливості інтерфейсу користувача.

JavaScript - це динамічна, розширювана мова з глибокою підтримкою мета-програмування. Код закладки, запущений у контексті JavaScript-коду зловмисника не може бути ідентифікований жодним з API, доступних для типового програмного забезпечення, тому зловмисник скористуватись цим. Щоб заповнити пароль на будь-якому ресурсі, менеджер паролів повинен успішно аутентифікувати користувача, завантажити (можливо зашифровані) облікові дані, розшифрувати їх, автентифікувати веб-програму та нарешті, виконати вхід. Робиться все це в ненадійному середовище сценаріїв веб-сайту. На даний момент і Firefox, і Chrome замість цього надають власні або ізольовані API для розширень браузера. На жаль, популярний мобільний браузери, включаючи Safari на iOS, Chrome на Android / iPhone та стандартний браузер Android, не підтримують розширення. Як результат, веб-менеджери паролів часто покладаються на закладки.

Розробники веб-менеджера паролів повинні розуміти модель безпеки Інтернету[6]. Наприклад, браузери використовують спільні маркери автентифікації, такі як файли cookie, між програмами (включаючи програми та розширення), що призводить до атак, таких як між-сайтовий запит (CSRF).

Головною перевагою менеджерів паролів є їх здатність протистояти фішинг-атакам. Користувачі фактично не запам'ятовують пароль веб-сайтів; натомість вони покладайтеся на менеджер паролів, який може виявити, що відкрито в браузері, і ввести правильний пароль. Така перевага спірна, якщо менеджер паролів сам по собі вразливий до фішингових атак. Ще гірше, у випадку менеджерів паролів - одна фішинг-атака може видати всі облікові дані користувача.

Вразливості CSRF та XSS є загальними в всіх веб-додатки [7]. XSS у веб-сайту дозволяє повністю переглянути обліковий запис, а вразливості CSRF у дозволяють зловмиснику оновлення, видалення та додавання довільних облікових даних користувача в база даних програмного забезпечення.

Одноразовий пароль (OTP) - це особливість LastPass, яка дозволяє користувачеві генерувати код автентифікації для скидання пароля, що дійсний лише для одного використання. Користувач може використовувати одноразовий пароль, щоб запобігти зловмиснику отримати доступу до чужого облікового запису LastPass. На рисунку 2.3 показано, як користувач створює OTP.

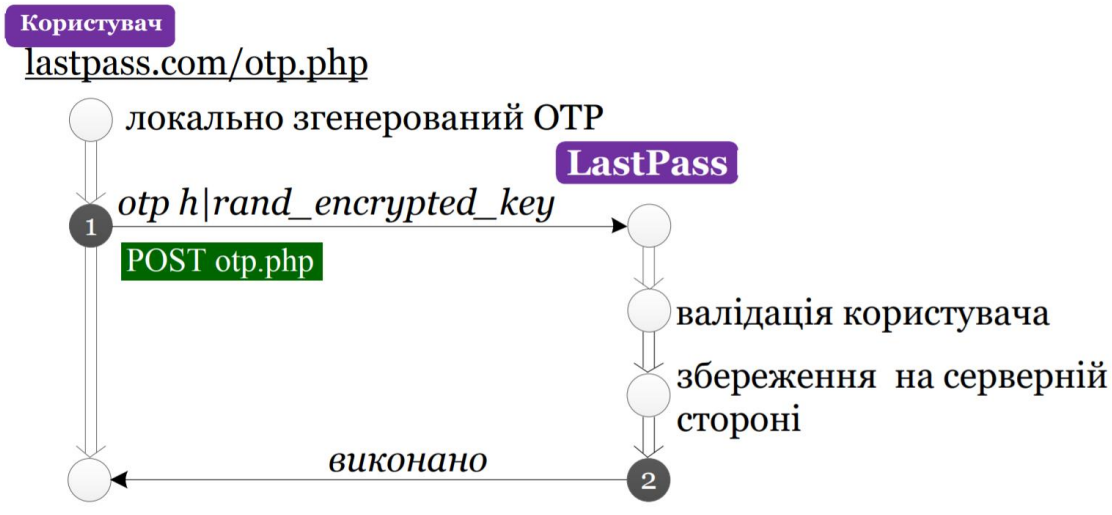


Рисунок 2.3 – Створення OTP

По-перше, LastPass зберігає список точок входу веб-додатків у незашифрованому вигляді, і зловмисник зможе прочитати список. Це є порушенням конфіденційності.

По-друге, зловмисник отримує зашифровану базу даних паролів ддля відгадування в ключів режимі офлайн. LastPass використовує ключ, отриманий із головного пароля користувача, який він повинен запам’ятати. На відміну від паролів, випадково сформованих LastPass, цей головний пароль, ймовірно, вразливий для здогадок.

І нарешті, атака призводить до заперечення сервісної атаки[8]. Зловмисник, зареєстрований як користувач, може видалити будь-які дані в базі даних, незважаючи на неможливість для розшифровки паролів. Оскільки ім'я користувача є частиною його посвідчення, відновлення всіх цих даних буде важким, а в деяких випадках неможливим.

Низка «класичних» вразливостей веб-додатків [9] у менеджерах паролів створює ризики для даних, якими циркулюють менеджери паролів XSS - це добре вивчена проблема, тому, відсутність суворої політики CSP в менеджері паролів повинна викликати тривогу у користувачів. У вище згаданих програмах, лише LastPass використовує заголовок Content-Security-Policy, хоча і з небезпечною політикою, яка дозволяє eval та inline сценарії.

Однією з проблем жетонів CSRF є необхідність створення і підтримувати стану користувача на стороні сервера. Це може бути громіздкою задачею для менеджерів паролів, які надають інтерфейс через розширення браузера. Натомість ці програми можуть покладатися на спеціальні заголовки (наприклад, X-CSRFToken) для захисту CSRF. Модель веб-безпеки забороняє evil.com встановлювати заголовки для перехресного походження.

Вразливості веб-додатків, про які йшлося вище, можна класифікувати за двома широкими категоріями. Перша категорія - недостатня безпека авторизації, що створює ризик вразливостей, а другою категорією є

передбачувані ідентифікатори. Заплутана автентифікація з авторизацією - це класична уразливість системи безпеки. Виходячи з трьох менеджерів паролів, які підтримують співпрацю, було виявлено вразливості авторизації у двох з них. Використання передбачуваного ідентифікатора повинні бути рідкісними, а будь-яке використання має бути причиною для перевірки безпеки.

SQL ін'єкція (SQL Injection) - один з поширених способів злому сайтів і програм, які працюють з базами даних, заснований на впровадженні в запит довільного SQL-коду [10]. Впровадження SQL, в залежності від типу використовуваної СУБД, може дати можливість атакуючому виконати довільний запит до бази даних (наприклад, прочитати вміст будь-яких таблиць, видалити, змінити або додати дані), отримати можливість читання і / або запису локальних файлів і виконання довільних команд на атакованому сервері. Припустимо, що на сайті є сторінка показу інформації користувача. Ідентифікатор цього користувача передається на посиланні як параметрі запиту: /user.php?usr_id= <ID>, де ID- це первинний ключ облікового запису. В PHP-сценарії використовується цей параметр для підстановки в SQL запит:

```
$user_id = $_GET['city_id'];
$res = mysqli_query($link, "SELECT * FROM users WHERE id = " . $user_id);
```

Якщо на сервері переданий параметр user_id рівний 10 (/users.php?user_id=10), то виконається SQL-запит:

```
SELECT * FROM users WHERE user_id = 10
```

Але якщо зловмисник передасть в якості параметра user_ id рядок -1 OR 1 = 1, то виконається запит:

```
SELECT * FROM users WHERE user_id = -1 OR 1=1
```

Додавання у вхідні параметри конструкцій мови SQL (замість простих значень) змінює логіку виконання всього SQL запиту!

У цьому прикладі замість показу даних по одному користувачеві, будуть отримані дані по всіх користувачах, тому що вираз 1 = 1 завжди істинно. Замість виразу SELECT ... могло бути вираз на оновлення даних, і тоді наслідки

були б ще серйозніші. Відсутність належної обробки параметрів SQL-запиту - це одна з найсерйозніших вразливостей.

Сенс атаки «Людина посередині (від англ. Man-in-the-middle, MITM) полягає в тому, що зловмисник «пропускає» веб-трафік жертви (Можливо, шляхом зміни параметрів DNS-сервера або файлу hosts) «через себе» [11]. У той час поки жертва вважає, що працює безпосередньо, наприклад, з веб-сайтом свого менеджера паролів. Трафік проходить через проміжний вузол зловмисника, який таким чином отримує все і відправляються користувачем дані (логін, пароль, ПІН-код і т. п.).. Багато менеджерів паролів є вразливими до атак перехоплення даних, які передаються між буфером обміну і додатком Спосіб боротьби для даної уразливості використовувати захищеної канал HTTPS.

Кейлоггер - програмне забезпечення або апаратний пристрій, що реєструє різні дії користувача – натискання клавіш на клавіатурі комп'ютера, руху і натиснення клавіш миші і т.д. Існує велика різноманітність варіантів кейлогерів, але основний поділ проводиться на програмні і апаратні. Найчастіше застосовується програмний кейлоггер, який є частиною шкідливої програми, такої як троян або руткіт. Як правило, це і є найпростіший варіант отримання доступу до системи без фізичного втручання. Один з найпоширеніших типів програмних кейлогерів вміє розгортати на цільовій машині готовий API, що записує кожне натискання клавіш [12].Як відомо, менеджери паролів не зберігають майстер-пароль і кожен раз при використанні менеджера користувач повинен

вводити його вручну. Для перехоплення таких паролів, як правило, використовують кейлоггери, плюс запис руху курсора і знімки екрану, якщо мала місце віртуальна клавіатура. Для боротьби з цією вразливість використовують антивіруси.

## 2.2. Дослідження алгоритму шифрування інформації в програмному забезпеченні

Чотири основні функції, що складають AES алгоритм - це додати ключ, замінити байт, зсунути рядки та перемішати стовпці. Блок даних, який повинен бути оброблений, розділений на масив байтів, що називається стеком. Додаючи ключ, перший крок перетворення, виконує побітове XOR блоку і матриці раундових ключів. Ця трансформація є зворотньою. Далі відбувається перетворення підбайта , що є нелінійною операцією заміни байтів, тобто складається з двох під-перетворень: мультиплікативної та афінне перетворення. У типовому програмному забезпеченні, ці два підкроки є об'єднані в єдину таблицю під назвою «box» або «S-box». Функція ShiftRows є процес лінійної дифузії, який діє окремо на останніх трьох рядках матриці стану. Просте транспонування байтів циклічно зміщується вліво у рядках із зміщенням, що змінюється від одного до трьох. Елементи другого, третього та чотирьох рядків є зміщені на один байт, два байти і три байти вліво, відповідно. Для повного шифрування дані передаються через Nr раунди (Nr = 10, 12, 14), які регулюються чотирма перетвореннями, як показано на рисунку 2.4.

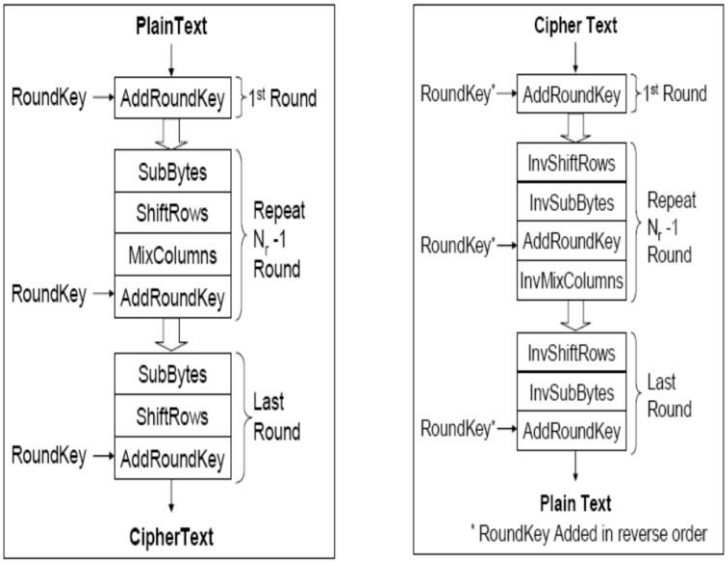



Рисунок 2.4 – Перетворення AES

Існує два методи, які зазвичай використовуються для того, щоб розрізняти критично важливі операції та не критичні для часу виконання. Перший метод - на основі аналізу перетворень AES на арифметичний або математичний рівень, тоді як другий метод заснований на аналізі перетворень та обсязі виконаних інструкцій. Другий спосіб буде виконаний з використанням програми, що дасть детальний опис інформація про виконані інструкції. Кожен AddRoundkey є реалізований з 8Nb байт-AND і 4Nb байт-OR, де Nb = довжина блоку / 32. Кожна операція SubByte містить 3Nb байт-AND і 2Nb байт-OR. Rожен ShiftRows складається з 3Nb зсувних байтів і 3Nb байт-OR. AES для одного блоку даних - це функція розміру блоку, розміру ключа, кількість раундів шифрування (Nr) і кількість циклів обробки, необхідних для виконання базових операції байт-AND, байт-OR, і побайтовий зсув (Ts), що виражено як:

$$TAES\text{-}ENCRYPT = (46Nb\ Nr - 30Nb)Ta + [31Nb\ Nr + 12(Nr - 1) - 20Nb\ ]To + [64Nb\ Nr + 96(Nr - 1) - 61Nb]Ts.$$

Значне обмеження AES пов'язане з його дешифрування. Код розшифрування має операцію Inverse MixColumns, яка використовує перетворення з іншим багаточленом, 0Bx3 + 0Dx2 + 09x + 0E.

Перетворення MixColumn є найбільш критичним по часу, тому що MixColumn і обернене перетворення MixColumn міститять більшість виконуваних інструкцій, які пов'язані з цілочисельними обчисленнями. Тому можна зробити висновок, що MixColumn та зворотні перетворення MixColumn є критичними за часом операції, але доцільними зі сторони безпеки.

Ланцюговий блок шифрування (CBC) є дуже популярним блочним режимом шифрування, де кожен відкритий текст виконує XOR з попереднім блоком зашифрованого тексту. Отже, кожен зашифрований текст є залежним на всіх блоках відкритого тексту раунду. Відкрите текстове повідомлення M поділяється на t n-бітові блоки $M_i$ та зашифрований текст $C_i$ подається як:

$$C_i = E_k\ (M_i\ XOR\ C_{i-1}),\ i = 1,2\ldots t$$

У режимі CBC значення $C_{i-1}$ використовується для рандомізації відкритого тексту шляхом комбінування з блоками даних $M_i$. CBC модель зображена на рисунку 2.5. Режим CBC є безпечний проти стандартних атак.

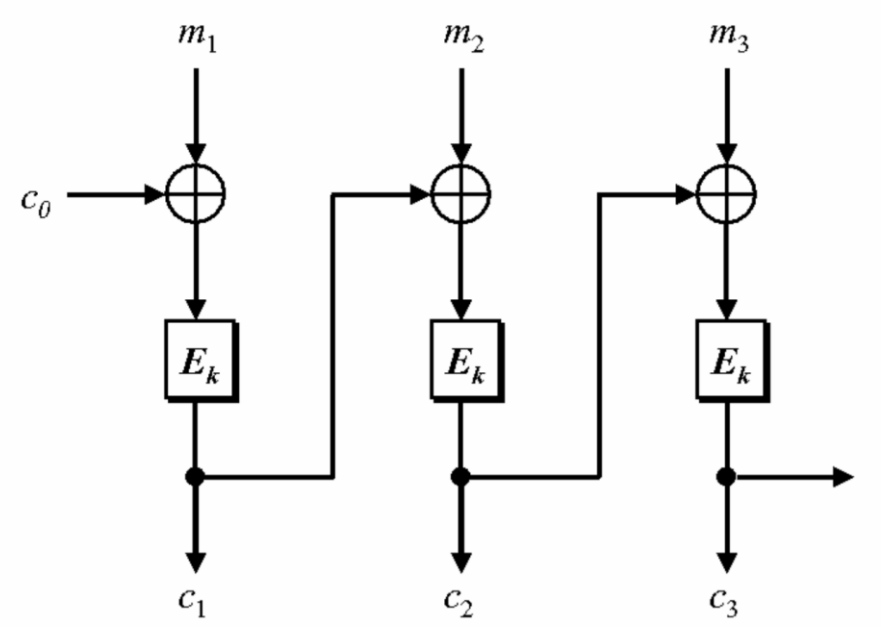



Рисунок 2.5 – Модель режиму CBC

Особливості:

- Наявність механізму поширення помилки: якщо при передачі відбудеться зміна одного біта шифротекста, дана помилка поширюється і на наступний блок. Однак на наступні блоки (через один) похибка не поширюється, тому режим CBC також називають самовідновлюватися.
- Нестійкий до помилок, пов'язаних з втратою або вставкою бітів, якщо не використовується додатковий механізм вирівнювання блоків.
- Зловмисник має можливість додати блоки до кінця зашифрованого повідомлення, доповнюючи тим самим відкритий текст.
- Для дуже великих повідомлень все-таки можливе застосування атак, заснованих на структурні особливості відкритого тексту.
- Вимагає доповнення повідомлень до довжини кратної довжині блоку.
- Вразливий для атаки оракула доповнень). У разі, якщо атакуючий може відправляти зашифровані повідомлення на розшифрування необмежену

кількість разів, він може, змінюючи певні байти, вгадати коректне доповнення. Це призводить до можливості розшифрувати повідомлення за винятком першого блоку, створити коректне довільне зашифроване повідомлення за винятком першого блоку без знання ключа.

## 2.3. Дослідження стратегії безпечного управління інформацією в базі даних.

Безпека баз даних включає широкий спектр тем безпеки, незважаючи на фізичну безпеку, мережеву безпеку, шифрування та автентифікацію. Захист баз даних будується на основі трьох принципів: конфіденційність, цілісність та доступність. Конфіденційність або таємниця стосується захисту даних від несанкціонованого розголошення, цілісність відноситься до запобігання несанкціонованому та неналежному зміненню даних, доступність стосується запобігання та відновлення помилок апаратного та програмного забезпечення, а також зловмисного доступу до даних. Безпека бази даних у будь-якому повинна охоплювати механізми контролю доступу, доступу до програм, вразливості та механізмів аудиту.

Основним методом захисту даних є обмеження доступу до даних. Це можна зробити за допомогою автентифікації, авторизації та контролю доступу. Ці три механізми різняться але зазвичай використовується в поєднанні з акцентом на контролі доступу для деталізації при передачі прав до конкретних об'єктів та користувачів. Наприклад, більшість систем баз даних використовують певну форму автентифікації, таку як ім’я користувача та пароль, для обмеження доступу до системи. Крім того, більшість користувачів отримують дозвіл або присвоєння визначених привілеїв конкретним ресурсам. Контроль доступу додатково вдосконалює шляхи присвоєння прав та привілеїв конкретним об’єктам даних та наборам даних. У межах бази даних ці об’єкти зазвичай включають таблиці, подання, рядки та стовпці.

Контроль доступу - це основна концепція безпеки [13]. Контроль доступу обмежує дії над об’єктами конкретних користувачів. У безпеці бази даних об'єкти відносяться до таких даних, як таблиці та стовпці, а також SQL-об'єкти, такі як подання та збережені процедури. Дії даних включають читання (вибір), вставку, оновлення та видалення або виконання для збережених процедур.

Як правило, контроль доступу визначається трьома способами: обов’язковий контроль доступу (MAC), дискреційний контроль доступу (DAC) та рольовий контроль доступу (RBAC). MAC та DAC забезпечують привілеї для вказаних користувачів або груп, до яких призначені користувачі. Правила MAC застосовуються системно і вважається статичним і більш безпечним.

Правила DAC надаються користувачем, вважаються динамічними та орієнтованими на вміст. MAC та DAC забезпечують потужні інструменти, але рольовий контроль доступу виявляється особливо ефективним для систем баз даних. Ролі аналогічні робочим функціям. Щодо ролей, основна увага приділяється ідентифікації операцій та об'єктам, до яких ці операції потребують доступу. Користувачі, отримуючи ролі, автоматично отримують і пов'язані привілеї. Наприклад, доктор Сміт може бути призначений на роль факультету. Визначення та визначення ролей та правильне надання прав доступу до дій та об’єктів, а потім належне присвоєння користувачам цих ролей є суть процесу. Після створення ролі формат реалізації RBAC дотримується шаблону:

- GRANT privilege_name
- ON object_name
- TO role_name;

Privilege_name визначає права, що надаються. Сюди входять такі права, як вибір даних, модифікація даних або маніпулювання структурою бази даних. ON визначає об'єкти бази даних та TO визначає ролі, до яких застосовуються ці привілеї.

Керування доступом до таблиць або стовпців бази даних часто потрібне і може бути введено в дію просто наданням пільг одному з цих об’єктів. Обмеження доступу до даних, що містяться в окремих записах, вимагає додаткових кроків. Наприклад, користувач повинен мати можливість лише переглядати або змінити рядок або рядки даних, які конкретно відповідають йому чи їй. Однак реалізація захисту на рівні рядків не може здійснюватися так само, як контроль доступу \ до об'єктів бази даних, таких як таблиці. Це тому, що вибір рядка базується на оцінці конкретного значення даних. Тому

поширеним способом реалізації захисту на рівні рядків є використання переглядів SQL. Можна побудувати подання, яке виконує оператор select, який повертає задані рядки даних, що обчислюються за певним значенням, таким як поточний користувач.

Більшість користувачів не отримують доступ до бази даних, безпосередньо входячи в систему баз даних. Натомість вони отримують доступ до бази даних через прикладну програму. Простий інструмент, відомий CRUD можна використовувати для чіткого визначення необхідних прав доступу, необхідних прикладній програмі. Зокрема, матриця безпеки забезпечує візуальне зображення кореляції між операціями або дозволами, необхідними для об'єктів бази даних та джерел вводу / виводу, таких як форми та звіти. Операції, зображені в матриці безпеки, включають вибір, створення (вставка), оновлення та видалення. У верхньому рядку матриці перелічені об’єкти таблиці бази даних.

Оскільки все більше і більше баз даних стають доступними через Інтернет та веб-додатки, їх вплив на загрози безпеці зростатиме. Мета програміста - зменшити сприйнятливість до цих загроз. Мабуть, найбільш розрекламованою вразливістю програми баз даних була ін’єкція SQL. Ін’єкції SQL - це чудові приклади для обговорення безпеки, оскільки вони втілюють одне з найважливіших питань безпеки баз даних - властиві ризики до неперевіреного введення користувачем. Ін'єкції SQL можуть відбуватися, коли оператори SQL динамічно створений за допомогою вводу користувача. Загроза виникає, коли користувачі вводять шкідливий код, який «обманює» базу даних на виконання ненавмисних команд. Насамперед через особливості мови SQL, які дозволяють вбудовувати коментарі за допомогою подвійних дефісів (- -), об’єднання операторів SQL, розділених крапками з комою, та можливість запиту метаданих зі словників бази даних. Рішенням для зупинки введення SQL є перевірка вхідних даних. Наприклад, модель SQL-коду може бути:

```
SELECT Count(*) FROM UsersTable
 WHERE UserName = ‘contents of username textbox’
```

```
 AND Password = 'contents of password textbox';
```

When a user enters a valid username, such as 'Mary' and a password

Коли користувач вводить дійсне ім'я користувача, наприклад, «Mary» та пароль «qwerty», запит SQL стає

```
SELECT Count(*) FROM UsersTable
 WHERE UserName='Mary'
 AND Password='qwerty';
```

Однак, якщо користувач вводить в якості імені користувача наступне: «АБО 1 = 1 - запит SQL стає:

```
SELECT Count(*) FROM UsersTable
 WHERE UserName='' OR 1=1 - -'
 AND Password='';
```

Вразливості введення SQL виникають внаслідок динамічного створення запитів SQL у прикладних програмах, які отримують доступ до системи баз даних. Запити SQL будуються з урахуванням вводу користувача та передається системі баз даних у вигляді рядкової змінної. Для вирішення проблеми валідації рядків запитів зазвичай використовують три підходи: використання чорного списку, використання білого списку або реалізація параметризованих запитів. Чорний список аналізує введені даних, де порівнює кожен символ із заздалегідь визначеним списком заборонених символів. Недолік використання чорного списку означає, що багато спеціальних символів можуть бути законними, але їх буде відхилено таким підходом. Підхід до білого списку подібний, за винятком того, що кожен символ порівнюється зі списком дозволених символів. Параметризовані запити використовують внутрішньо визначені параметри для заповнення в раніше підготовлений оператор SQL.

Тонкою вразливістю, виявленою в технологіях баз даних, є висновок або можливість отримувати невідому інформацію на основі отриманої інформації. Проблема полягає в тому, що відсутні ідеальні рішення. Єдині рекомендовані рішення включають засоби контролю, пов'язані з запитами або елементами керування, що стосуються окремих елементів у базі даних. Іншими словами,

конфіденційні дані, запитувані в запиті, або не надаються, або відповіді наведені близько, але не точно, заважаючи користувачеві отримати достатньо інформації для висновків. Ні з того, ні з іншого це не дасть ідеальні рішення, оскільки вони мають обмежувальний характер. Однак коритсувачам важливо розуміти ризики висновку та те, як це може статися. Халепа також може статися, коли користувачі можуть з'ясувати інформацію з даних, доступних для них на рівні їхньої безпеки, навіть незважаючи на те, що ця конкретна інформація захищається на більш високому рівні безпеки та доступу.

## 2.4. Висновки до другого розділу

Дослідження безпеки п'яти веб-менеджерів паролів показало критичні вразливості, із-за яких зловмисник може вкрасти довільні дані з облікового запису користувача. Широкий спектр виявлених вразливостей робить необхідною розробку безпечного веб-менеджера паролів, що тягне за собою систематичний і поглиблений підхід до захисту інформації.

Багато менеджерів паролів є вразливими до атак перехоплення даних, які передаються між буфером обміну і додатком, а також до відновлення залишкової інформації. У деяких менеджерах мастер-пароль зберігається у відкритому вигляді або ключі шифрування містяться в коді веб-сайту. Іноді дістати доступ до облікової інформації, що зберігається в менеджерах паролів, можна шляхом зараження пристрою жертви шкідливими програмами. Кіберзлочинці можуть скористатися вразливостями на сайтах, не маючи при цьому root-прав. Дані уразливості ставлять під сумнів надійність аналізованих менеджерів паролів. У зв'язку з цим постає необхідність розробки надійного менеджера паролів.

Потреба в захисті комп'ютерних систем є актуальною. Все більший обсяг конфіденційних даних зберігається в базах даних і більшість із цих баз даних робляться доступними через Інтернет. Основними цілями безпеки баз даних є запобігання несанкціонованому доступу до даних, запобігання

несанкціонованому втручанню або модифікації даних, а також забезпечити, щоб вони залишались доступними у разі потреби. Поняття, що стосуються безпеки баз даних, багатогранні.

# РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ СПЕЦІАЛІЗОВАНОЇ СИСТЕМИ КЕРУВАННЯ ПАРОЛЯМИ

## 3.1. Основні відомості про веб-розробку

Веб-розробка – це робота, пов'язана з розробкою веб-сайту для Інтернету (всесвітньої павутини) або інтрамережі (приватна мережа). Веб-розробка може варіюватися від розробки простої єдиної статичної сторінки простого тексту до складних Інтернет-додатків (веб-додатків) для електронного бізнесу та послуг соціальної мережі. Більш повний перелік завдань, до яких часто відноситься веб-розробка, може включати також веб-інжиніринг, веб-дизайн, розробку веб-контенту, взаємодію з клієнтами, сценарії на стороні клієнта / сервер, веб-сервер та конфігурація безпеки мережі та розвиток електронної комерції. Серед веб-професіоналів, "веб-розробка", як правило, стосується основних аспектів, не пов'язаних із розробкою веб-сайтів: написання розмітки та кодування. Останнім часом веб-розробка мала на увазі створення систем управління контентом (CMS). Ці CMS можуть бути створені з нуля, закритим або з відкритим вихідним кодом. Загалом, CMS діє як проміжне програмне забезпечення між базою даних та користувачем через браузер. Основним достоїнством CMS є те, що це дозволяє нетехнічним людям внести зміни до свого веб-сайту без технічних знань.

Для великих організацій та підприємств, команди веб-розробників можуть складатися із сотень людей (веб-розробників) і виконувати стандартні методи, такі як Agile методології, при розробці веб-сайтів. Менші організації можуть вимагати лише одного постійного або контрагента розробника або вторинного розпорядження пов'язаними робочими місцями, такими як графічний дизайнер або технік інформаційних систем. Веб-розробка може бути спільним зусиллям між відділами, а не доменом призначеного відділу. На даний час існує три види спеціалізації, які характеризують веб-розробників: інтерфейсний розробник, розробник програмного забезпечення та розробник

повного стека. Розробники інтерфейсу мають справу з макетом і візуальними зображеннями веб-сайту, тоді як розробники заднього виду вирішують функціональність веб-сайту.

Бізнес-програми зазвичай розбиваються на логічні фрагменти, які називаються "рівнями". Для кожного рівня існують певні функції. Веб-програми, як правило, багатогранні за своєю природою, найпоширенішою структурою є трирівнева архітектура. Веб-браузер - це перший рівень, який є рівнем презентації. Використовуючи деякі динамічні технології веб-вмісту, такі як ASP, ASP.NET, CGI, ColdFusion, JSP / Java, PHP, Perl, Python, Ruby on Rails або Struts2 середнього рівня будуть відповідати за логіку програми. База даних - сховище третього рівня. Перед початком розробки, моделювання сайтів з використанням блок-схеми, макети екранів є загальним явищем. Існує ряд переваг використання стандартів для розробки веб-додатків включаючи:

- послідовний стиль дизайну на веб-сайтах в організації,
- стандартизацію документа про вимоги до веб-додатків
- вказівки щодо тестування веб-сайтів .

На рівні презентації веб-сторінка відображаються в браузері. Традиційно веб-сторінки містять лише HTML-код. Сьогодні, презентаційний рівень веб-додатків забезпечує майже такий самий функціонал користування, як і програми. Такі інтерфейси використовують група технологій, спільно названі Rich Internet Application (RIA). В інтерфейсах RIA на рівні сценаріїв використовуються такі мови, як JavaScript (та його розширення AJAX та jQuery) та CSS. Такий код виконується на стороні клієнта або інтерпретується у формі платформи браузера. Сценарії на стороні клієнта, як правило, можуть переглядати будь-хто.

Існує два способи побудови рівня програми - за допомогою мов сценаріїв на стороні сервера або складені бізнес-об'єкти. До мов сценаріїв на стороні сервера належать ASP, PHP, JSP, ColdFusion, Perl, Ruby, WebObjects та Python. Сценарії на стороні сервера зазвичай інтерпретуються та виконуються веб-

сервером і не є видимими для клієнтів. PHP - дуже поширена серверна мова сценаріїв.

### 3.2. Вибір засобів для створення програмного забезпечення

В якості основної операційної системи для роботи сервера використовується Linux. Для роботи клієнтської частини можна також використовувати ОС Android, Windows, MacOS або iOS. Створення та проектування прикладного ПЗ велось на ОС Windows 10 Корпоративна LTSC. Основою для сайта є створена система управління контентом за допомогою PHP-фреймворку Laravel для розробки з використанням архітектурної моделі MVC. Обов'язковими є також до використання HTML, CSS, JavaScript, Ajax для створення веб-сторінок та мова SQL для оброки даних із СКБД HeidiSQL. Ajax є необхідною технологією з використанням JavaScript для обробки даних без перезавантаження веб-сторінок.

До сервісних програм, які будуть використовуватись з боку користувачів, можна віднести антивірусне програмне забезпечення, що буде забезпечувати безпеку даних користувачів наряду із шифруванням різноманітними алгоритмами.

Як інструментальне програмне забезпечення необхідно залучити інтерпретатор мови PHP у складі веб-серверу Apache як з боку розробки, так і зі сторони подальшої роботи розроблюваної системи на віддаленому сервері.

В якості системи контролю версій для розробки було використано Git із завантаженням подальших даних в репозиторій на GitHub.

*Прикладне програмне забезпечення.*

До пакетів прикладних програм, що необхідні для створення веб-платформи відносяться таке програмне забезпечення як PHPStorm, Blade, Composer, GitBash, SublimeText, HeidiSQL.

PHPStorm є комерційним крос-платформним середовищем для розробки на мові PHP. Розробляється компанією JetBrains. За допомогою нього виконується весь процес написання та модифікації програмного таких

компонентів як контролери, моделі та вигляди. Також надає можливість безпосередньо вносити зміни до таблиць БД.

Blade є шаблонізатором, що сумісний із PHP-фреймворком Laravel, і дозволяє здійснювати за допомогою різноманітних директив структуризації та коректне відображення контенту, що формується у виглядах для веб-сторінок.

Composer - це пакетний менеджер рівня додатків для мови програмування PHP, який надає засоби по керуванню залежностями в PHP-додатку. Працює через інтерфейс командного рядка та дозволяє керувати такими важливими процесами як встановлення Laravel та його модифікація, створення моделей та контролерів та внесення змін у залежності модулів.

GitBash – сервісна утиліта системи контролю версій Git для фіксації комітів та відправки/отримання даних в/із віддаленого репозиторію, створеному заздалегідь для проекту.

SublimeText – пропрієтарний текстовий редактор для внесення змін у у файли формату HTML та CSS.

HeidiSQL є влаштованим клієнтом для взаємодії із створеною базою даних типу MySQL та додавання нових сесій їх обробці за допомогою різноманітних програмних розробок, в тому числі з використанням PHPStorm.

До програмних розробок відносяться такі основні компоненти функціонування системи як моделі, вигляди та контролери (model, views, controllers).

Моделі надає дані і реагує на команди контролера, змінюючи свій стан. Модель створюється у відповідності до таблиці з усіма зв’язками в базі даних.

Вигляд або представлення відповідає за відображення даних моделі для користувача, реагує на зміни моделі. Дані у нашому випадку подаються у вигляді веб-сторінок.

Контролери інтерпретують дії користувача оповіщаючи модель про необхідність змін. Тобто вони забезпечують зв’язок між користувачем та системою і відправляють дані від користувача до системи і навпаки.

## 3.3. Специфікації вимог до програмного забезпечення

1. Введення
   1.1. Призначення

   Програмне забезпечення для керування паролями користувача та організації.

   1.2. Узгодження, прийняті у документах

   - Програмне забезпечення може шифрувати паролі двома способами
   - Функціонал повинен включати керування організаціями
   - Можливість категоріювання ресурсів
   - Наявність генератора паролів
   - Наявність генератора ключів

   1.3. Аудиторія та рекомендації по змісту

   Користувачі Інтернету та власники організацій

   1.4. Границі проекту

   Програмне забезпечення слугує як звичайний веб-сервіс. Для використання, потрібно зареєструватись і увійти в особистий кабінет. Користувач може створювати, редагувати, групувати да видаляти ресурси. Може створювати, редагувати, керувати організаціями

   1.5. Посилання

   http://password-manager.isitlab.com/

2. Загальний опис
   2.1. Особливості продукту

   - Швидкий доступ
   - Генератор паролів
   - Генератор ключів
   - Функціонал категорій
   - Функціонал для організацій
   - Керування користувачами в підрозділах

2.2.Класи та характеристики користувачів

- Клієнт – власник особистого кабінету в сервісі. Здійснює повний контроль на своїми записами та ресурсами

2.3.Операційне середовище

LAMP, географічне розташування Тернопіль, Україна

2.4.Обмеження дизайну та реалізації

Режим адміністрування – Bootstrap 4, JQuery

3. Функції системи

3.1.Функція системи X

1. Швидкий доступ до ресурсів
2. Управління обліковими записами
3. Управління організаціями
4. Управління категоріями

3.2.Опис та приорітетність

1. Швидкий доступ до ресурсів

   Після авторизації, користувач завжди першим повинен побачити його ресурси

   Пріоритет - низький

2. Управління обліковими записами

   В системі, можна виконувати такі дії над обліковим записом: додавання, редагування, групування, видалення, копіювання даних.

   Пріоритет - високий

3. Управління організаціями

   Користувач може створювати безліч організацій. В кожній із них – підрозділи. В підрозділ можливо додавати посилання та користувачів, які можуть використовувати ресурси. Керівник може призначати помічників по підрозділу.

   Пріоритет - високий

4. Управління категоріями

Система дозволяє здійснювати управління категоріями. Можна створювати, редагувати, додавати посилання та видаляти їх, а також видаляти категорії.. При видаленні категорії – посилання не видаляют Пріоритет - високий

3.3.Послідовності «дія-реакція»

1. Швидкий доступ до ресурсів
    I. Пройти авторизацію
    II. Здійснювати керування ресурсами
2. Управління обліковими записами
    I. Натиснути на кнопку «Додати»
    II. Заповнити всі поля, згенерувати пароль та вибрати метод шифрування
    III. Зберегти обліковий запис
    IV. Вибір категорії здійснити при редагуванні, створені
3. Управління організаціями
    I. Створити організацію
    II. Створити підрозділи
    III. Додати облікові записи в кожен підрозділ
    IV. Додати існуючих користувачів в підрозділ, шляхом введення електронної пошти
    V. Призначити замісника в підрозділі
4. Управління категоріями
    I. Перейти до створення категорій
    II. Ввести назву та натиснути кнопку «Зберегти»
    III. Перейти до ресурсів та вказати потрібну категорію

3.4.Функціональні вимоги

1. Швидкий доступ
    I. Можливість пошуку за назвою
    II. Можливість пошуку за посиланням
2. Управління обліковими записами

I. Можливість додавання, редагування, групування та видалення
II. Функціонал генератора паролів
III. Оцінка безпеки пароля
IV. Вибір режиму шифрування пароля
V. Можливість копіювання пароля
VI. Швидкий перехід на редагування
VII. Видалення без перезавантаження сторінки

3.5. Вимоги безпеки

- Під час створення ресурсу, пароль має бути маскований
- Пароль в базі даних не повинен зберігатись у відкритому вигляді
- Запит повинен валідуватись зі сторони сервера
- API запит повинен використовувати особистий токен користувача в заголовках запиту

4. Вимоги до зовнішнього інтерфейсу

4.1.Інтерфейси адміністративної частини

Меню на всіх сторінках. Вибраний пункт меню підсвічується. Кольорова схема складається з двох кольорів:

- Фіолетовий (#886ab5)
- Білий (#ffffff)

Шрифт для заголовків - Helvetica Neue:

Шрифт для іншого тексту - Roboto

В адміністративній частині, присутній шаблон повідомлення виконаної дії.

**Розробка захищеної системи керування паролями**

Результати проектування та створення спеціалізованої системи керування паролями з метою підвищення ефективності функціонування та безпеки підприємств цілком задовольняє поставлену мету магістерської роботи.

Для того щоб розпочати роботу, потрібно здійснити авторизацію в системі. На рис.3.1 можна побачити, що користувач повинен ввести свій логін та пароль. Якщо ж немає облікового запису – створити його.

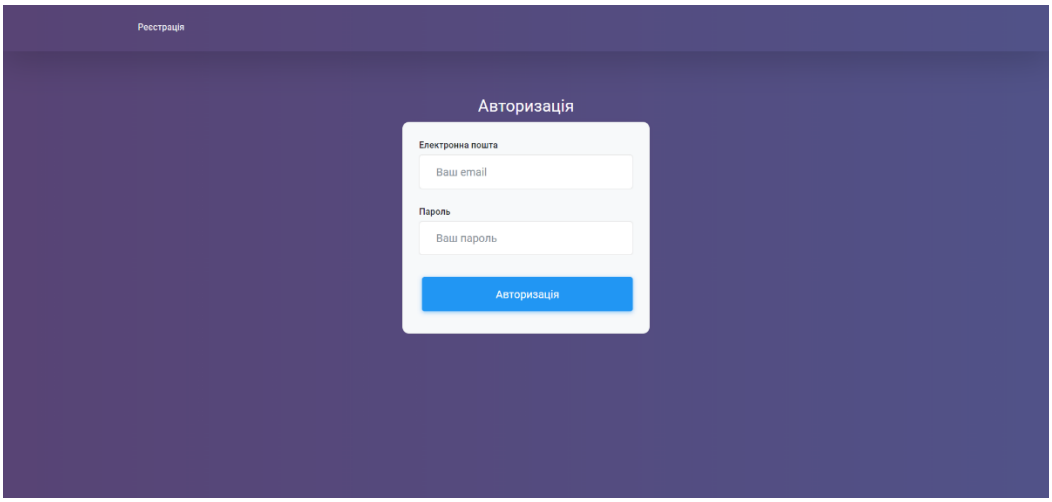



Рисунок 3.1 – Сторінка авторизації користувача

Під час створення користувача, система автоматично присвоює api_token – набір символів, кількістю в 255 та генерується стандартний ключ для першого (простішого) методу шифрування паролів. Токен потрібний для аутентифікації користувача, під час API-запитів. Пароль зберігається в захешованому вигляді, що можна побачити на рис.3.2.

| id | email | email_verified_at | password |
|---|---|---|---|
| 1 | revo0708@gmail.com | <null> | $2y$10$atpBVa4iKMA3Jxt0kT//1.zEoLNA.JKl. |
| 2 | nhaley@hotmail.com | <null> | $2y$10$ybJjW/cYlgiBvoBnFFhgwew0Ci34a9qv. |
| 3 | ylockman@schaden.com | <null> | $2y$10$yxNttYq5Aokl97DkOBC6denTsOzdOQsC. |
| 4 | arnold.strosin@koch.com | <null> | $2y$10$Lo.mv/4kpEy3Phnu6lCXtuHujJQApdBk. |
| 5 | sanford.mossie@hotmail.com | <null> | $2y$10$d6eqL6ECJOhdyNWj.fk/zOrwFIBHTxHI. |
| 6 | jay.damore@hotmail.com | <null> | $2y$10$dDy3/s7bhQxjMiYBxX0PAeCt2icZQRDU. |
| 7 | mthiel@yahoo.com | <null> | $2y$10$CzaNJUOGH3JRPThpVR1JPerhZM75JUAa. |
| 8 | marks.mabel@gleason.com | <null> | $2y$10$Abp3FhcP99wPmqw5uTzWFebWkG26942t. |
| 9 | oboyer@nitzsche.org | <null> | $2y$10$rJQKaatNpz3S7YRA6tuwceyGe4SFbYcB. |
| 10 | ylockman@hotmail.com | <null> | $2y$10$mU6GpAL6P1vJzeOFzWtIFuHdpIYY3Qn6. |
| 11 | ehilpert@yahoo.com | <null> | $2y$10$oXYxRhKmJhxS0CxuQcsdjeQLZjo/5fpZ. |

Рисунок 3.2 – Приклад хешування паролів в базі даних

Після проходження авторизації, користувач попадає на вкладку «Ресурси», де відображені його категорії ресурсів та самі посилання, що можна побачити на рис.3.3

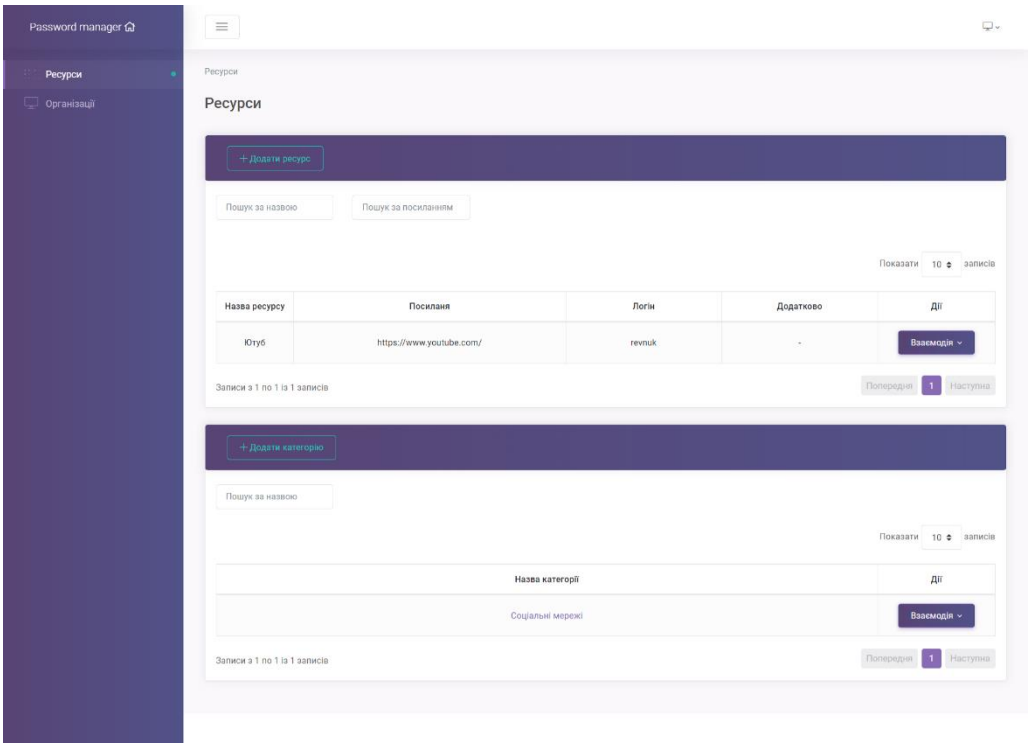



Рисунок 3.3 – Ресурси та категорії користувача

Для того щоб додати ресурс – потрібно натиснути на відповідну кнопку зверху. На рис. 3.4 можна побачити, що э декілька текстових полів, поле для пароля та метод шифрування на вибір

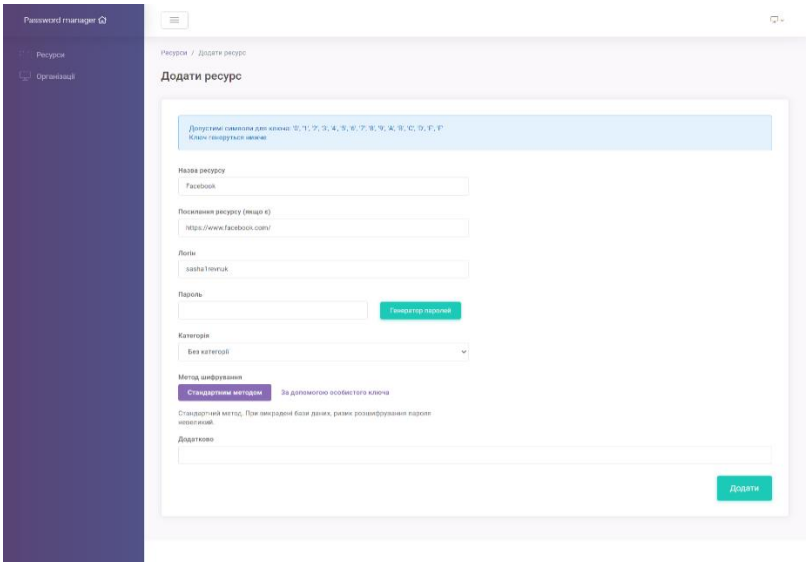


Рисунок 3.4 – Форма створення\редагування ресурса

«Стандартний метод» - пароль шифрується за допомогою ключа, який додається до запису користувача в базі даних при створенні. «За допомогою особистого ключа» - метод який дає можливість зашифрувати пароль за допомогою власного ключа. Якщо його немає – можна скористуватись функціоналом генерації ключа. Слід пам’ятати, що відновити пароль при втраті ключа – неможливо. На рис.3.5 можна побачити приклад генерації ключа для шифрування.

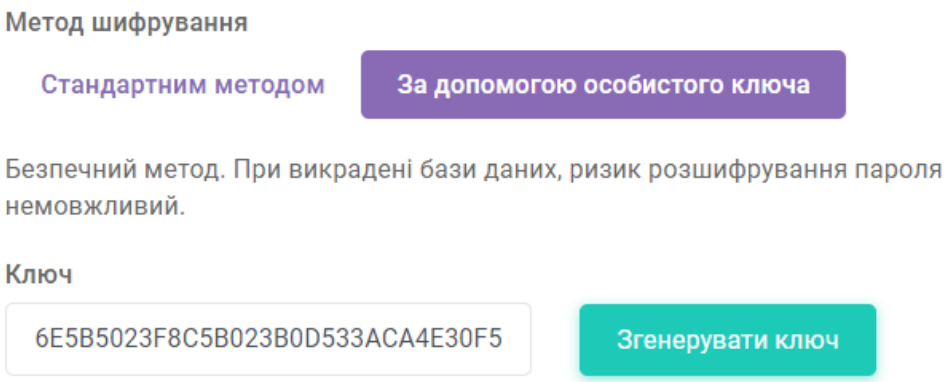



Рисунок 3.5 – Приклад генерованого ключа

Пароль можна ввести як свій, так і згенерувати за допомогою генератора. Останній можна побачити на рис.3.6. Генератор дає можливість вибрати які варіанти символів, з яких буде генеруватись пароль, а саме: малі літери, великі літери, спеціальні символи та цифри. Також можна вибрати довжину потрібного пароля. Знизу користувач побачить полосу, яка відобразить міцність згенерованого пароля.

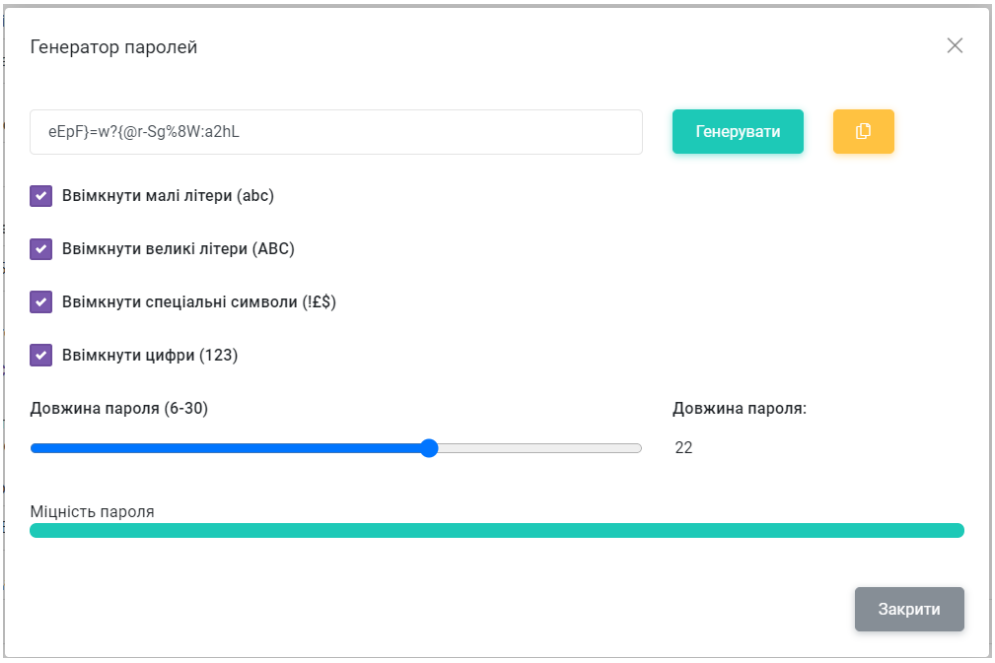



Рисунок 3.6 – Генерація нового пароля

Під час створення, дуже важливу роль Request-файли. Це правила, якими валідується отримана інформація від користувача системою. Правила валідації можна побачити в додатку А. Після створення ресурсу, можна редагувати, видаляти та копіювати пароль. Якщо пароль зашифрований першим методом, т опри копіюванні пароля – користувач отримає відразу результат. В іншому випадку – клієнт побачить форму, яка запропонує ввести ключ, для того щоб розшифрувати пароль та скопіювати його. Форму зображена на рис. 3.7.

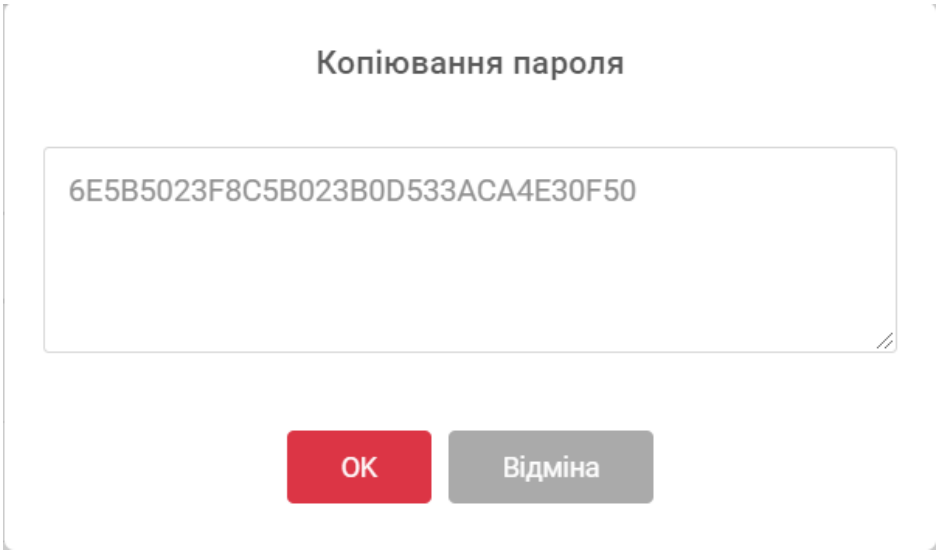



Рисунок 3.7 – Форма для введення ключа розшифрування пароля

Таблиця дає можливість для моментального пошуку ресурсів по назві та посиланні.

Для того щоб створити категорію потрібно перейти в відповідний розділ, натиснувши на кнопку, та ввести назву категорії. Форму можна побачити на рис.3.8.

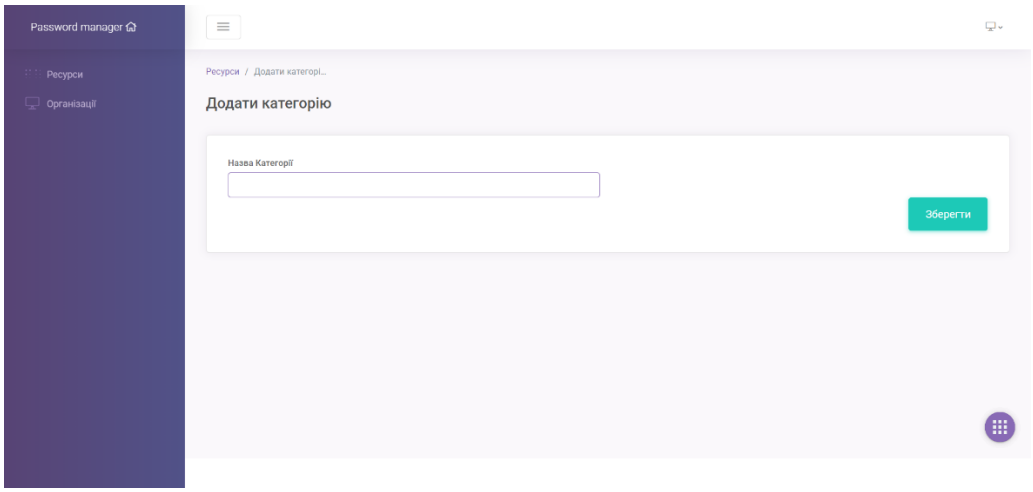



Рисунок 3.8 – Форма додавання категорії

При переході в відповідну категорію – користувач побачить ресурси, що належать до категорії

Для того щоб керувати своїми організаціями – потрібно перейти на відповідний пункт меню справа. Тут користувач побачить таблицю зі своїми корпораціями, а також можливість переглянути ті, де він працівник в одному із відділів. На рис.3.9 можна побачити таблицю з організаціями

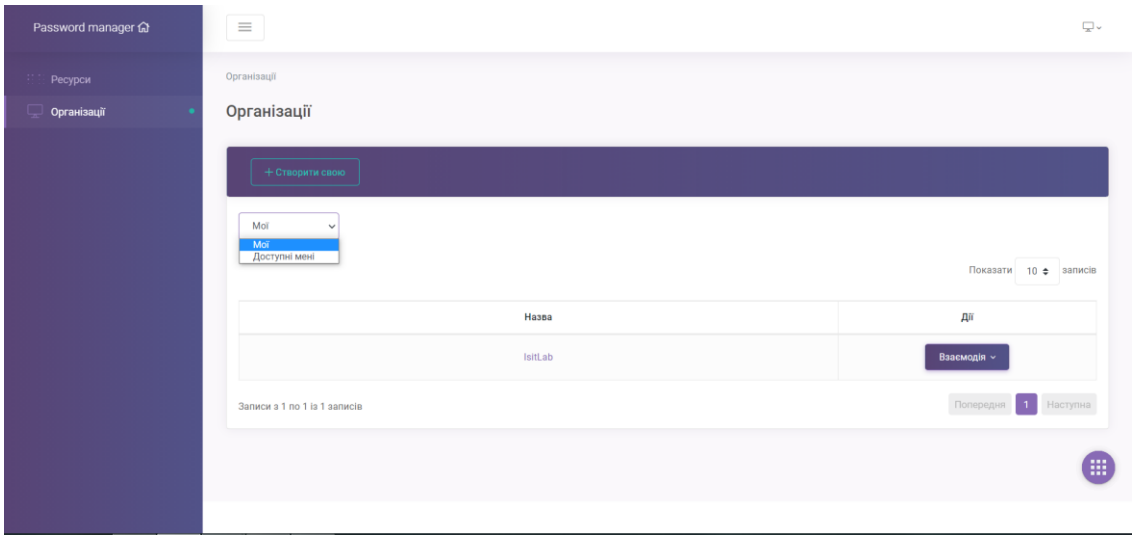


Рисунок 3.9 – Організації користувача

Форма створення та редагування організації і відділів аналогічна як в категоріях. Після того, як користувач створив організацію і відділи – він може переглядати, додавати та видаляти ресурси, а також здійснювати керування працівниками відділу. Власник може призначати адміністратора відділу. Останній – має всі права як у власника, але не може керувати ним та іншими адміністраторами. На рис.3.10-3.11 можна побачити вигляд сторінки організації адміністратором та звичайним працівником.

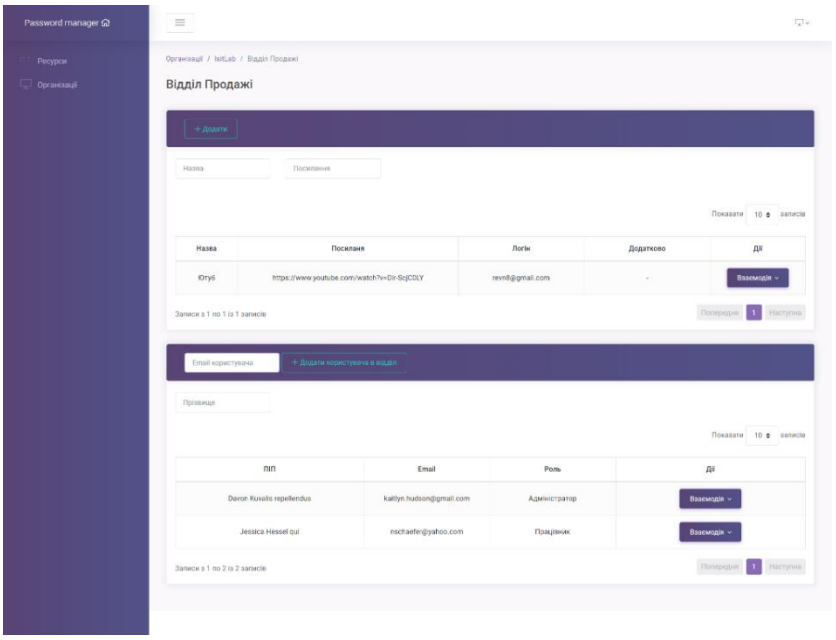



Рисунок 3.10 – Вигляд сторінки для адміністратора та власника відділу

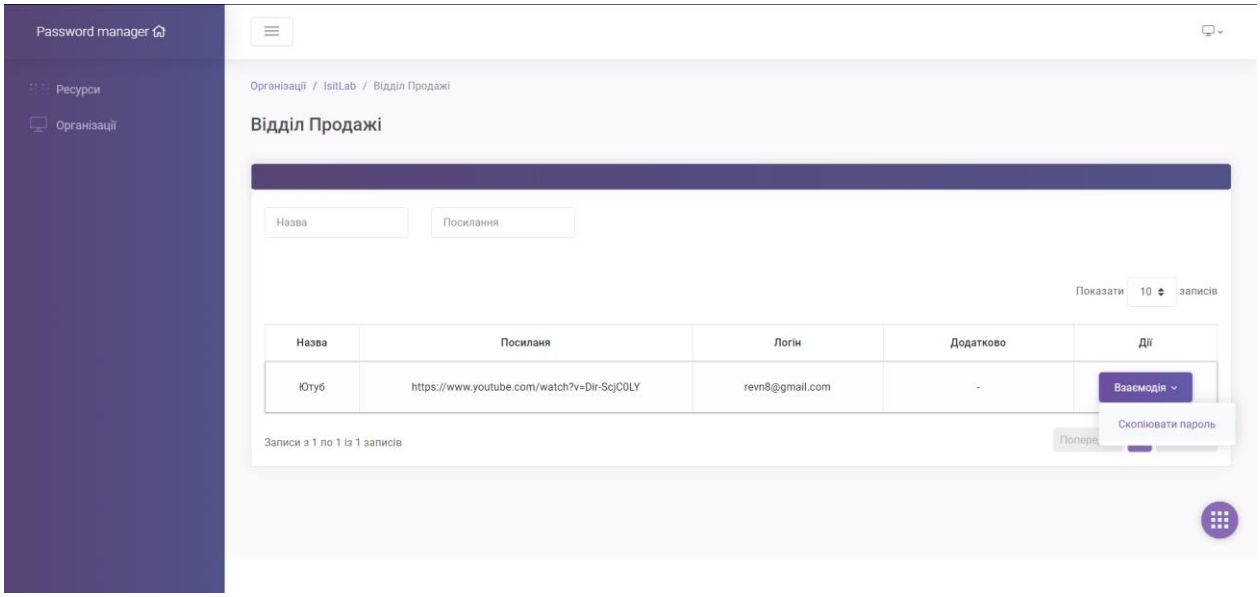



Рисунок 3.11 – Вигляд сторінки для працівника відділу

Отже, працівники можуть тільки копіювати пароль в відділі, до якого його додав або влсник, або адміністратор.

## 3.4. Висновки до третього розділу

В даному розділі було зроблено огляд та аналіз основних технологій, які використовуються в WEB-розробці. Було розроблено специфікацію вимог до програмного забезпечення та обрані відповідні інструменти для розробки платформи. Було розроблено спеціалізовану системи керування паролями з метою підвищення ефективності функціонування та безпеки підприємств. Система функціонує в реальному часі і виконує всі поставлені завдання.

# РОЗДІЛ 4. ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

## 4.1. Охорона праці

Промислова безпека, що її розглядає охорона праці, має велике значення для працюючих, оскільки саме вона контролює фізичний стан працівника, що не може не позначитись на його житті, здоров'ї, а також продуктивності праці в тому числі і у галузі розробки програмного забезпечення.

Незадовільний стан охорони праці може викликати соціально- економічні проблеми працюючих та їх родин. Саме тому соціально- економічне значення охорони праці полягає в наступному: зростанні продуктивності праці, збільшенні валового внутрішнього продукту, зменшенні витрат на оплату лікарняних і виплат компенсацій за шкідливі умови праці та інше.

В цьому розділі проводиться аналіз небезпечних та шкідливих для людини і навколишнього середовища чинників, які виникають під час проведення роботи зі спеціалізованою системою керування паролями на підприємстві, тобто безпосередньо з ЕОМ. Тут висвітлюються, зокрема, технічні рішення з гігієни праці та виробничої санітарії та технічні рішення з безпеки при проведенні дослідження.

Технічні рішення з гігієни праці та виробничої санітарії

Мікроклімат та склад повітря робочої зони

Під мікрокліматом виробничих приміщень розуміють клімат внутрішнього середовища цих приміщень, який визначається діючими на організм людини поєднаннями температури, вологості та швидкості руху повітря, а також інтенсивності теплового випромінювання.

Якщо з технічних чи економічних міркувань оптимальні норми не забезпечуються, то встановлюються допустимі величини показників мікроклімату [14].

Вибираємо для приміщення для проведення роботи зі спеціалізованою системою керування паролями на підприємстві, категорію важкості робіт за фізичним навантаженням – легка Іб.

Повітря робочої зони не повинно містити шкідливих речовин з концентраціями вище гранично допустимих концентрацій (ГДК) в повітрі робочої зони та підлягає систематичному контролю з метою запобігання можливості перевищення ГДК.[15]

При використанні ЕОМ джерелом забруднення повітря є також іонізація молекул речовин, що знаходяться в повітрі.

Для встановлення необхідних за нормативами параметрів мікроклімату і складу повітря робочої зони передбачено такі заходи:

1) у приміщенні повинна бути розміщена система кондиціонування для теплого і опалення для холодного періодів року;

2) припливно-витяжна система вентиляції, а при несприятливих погодних умовах кондиціонування.

Виробниче освітлення

Для забезпечення раціональних гігієнічних умов на робочих місцях значні вимоги висуваються до якісних та кількісних параметрів освітлення.

З точки зору задач зорової роботи в приміщенні, в якому проводяться роботи зі спеціалізованою системою керування паролями на підприємстві, знаходимо, що вони відповідають IV розряду зорових робіт. Вибираємо контраст об’єкта з фоном – середній та характеристику фону – середню, яким відповідає підрозряд зорових робіт в.

З метою забезпечення нормованих значень параметрів освітлення передбачено такі заходи:

1) за недостатнього природного освітлення у світлу пору доби доповнення штучним за допомогою люмінесцентних ламп з утворенням системи суміщеного освітлення;

2) застосування загального штучного освітлення в темну пору доби.

Виробничі віброакустичні коливання

Зважаючи на те, що під час експлуатації пристроїв крім усього іншого обладнання використовується устаткування, робота якого генерує шум, потрібно передбачити захист від шуму.

Визначено, що приміщення, де проводиться робота із спеціалізованою системою керування паролями на підприємстві може мати робочі місця із шумом, що спричиняється вентиляторами блоку живлення ЕОМ і кулерами мікропроцесора та відеокарти.

Для встановлення нормованих показників шуму в приміщенні передбачено такі заходи:

1) оздоблення стін спеціальними перфорованими плитами, панелями з метою шумопоглинання;

2) контроль рівня шуму не менше 1 разу на рік.

Технічні рішення щодо безпеки при проведенні дослідження

На теперішньому етапі розвитку техніки, автоматизації розробок та досліджень широкого використання на робочому місці набули ЕОМ. Велика кількість прикладних програм перетворює ЕОМ на основне знаряддя праці робітника на підприємстві.

Безпека щодо організації робочих місць

Розміщення робочих місць, оснащених ЕОМ виконується в приміщеннях з однобічним розміщенням вікон, що обов'язково мають бути оснащені сонцезахисним засобами: шторами та жалюзями [16].

При розміщенні робочих місць у приміщеннях з джерелами шкідливих та небезпечних виробничих чинників, вони зобов’язані розміщатись в повністю ізольованих кабінетах з природним освітленням та організованою вентиляцією. Площа, на якій розташовується одне робоче місце для обслуговуючого персоналу, має складати не менше 6,0 м2, об'єм – не менше ніж 20 м3, а висота – не менше 3,2 м.

Робочі місця з відеодисплейним терміналом зобов’язані розміщатися на віддалі не менше як 1,5 м від стіни з віконними прорізами, від інших стін – на відстані 1 м, одне від одного на відстані не менше ніж 1,5 м. У випадку

розміщення робочих місць необхідно виключити можливість прямого засвічування екрану джерелом природного освітлення. Робоче місце раціонально розташовувати так, щоб природне світло падало на нього збоку, переважно з лівого.

Розташовувати відеодисплейний термінал на робочому місці необхідно так, щоб поверхня екрана повинна знаходитись на віддалі 400-700 мм від органів зору користувача. Висота робочої поверхні столу при виконанні роботи сидячи повинна налаштовуватись в межах 680-800 мм. Робочий стіл повинен мати простір для ніг висотою не менше 600 мм, шириною не менше як 500 мм, глибиною на рівні колін не менше 450 мм та на рівні витягнутої ноги не менше як 650 мм.

Поверхня підлоги повинна бути гладкою, без вибоїн, не слизькою, мати антистатичні властивості, зручною для вологого прибирання. Не дозволяється використовувати для оздоблення інтер'єру полімерні матеріали, що виділяють у повітря шкідливі хімічні речовини.

## 4.2. Безпека в надзвичайних ситуаціях

Від ефективності розроблення та впровадження в життя заходів із запобігання та ліквідації надзвичайної ситуації в разі її виникнення залежатиме життя та здоров’я персоналу та відвідувачів цих підприємств і розміри заподіяної шкоди.

Відповідно до Кодексу цивільного захисту України, підготовка персоналу на підприємствах незалежно від форм власності до дій у надзвичайних ситуаціях здійснюється за спеціально розробленою схемою заходів захисту населення та територій.

Для великих і малих підприємств система заходів захисту від надзвичайних ситуацій включає:

- планування та здійснення необхідних заходів для захисту своїх працівників, об’єктів господарювання;

- розроблення планів локалізації та ліквідації аварій з подальшим погодженням з Державною службою України з надзвичайних ситуацій;
- підтримання у готовності до застосування сил і засобів із запобігання виникненню та ліквідації наслідків надзвичайних ситуацій;
- створення та підтримання матеріальних резервів для попередження та ліквідації надзвичайних ситуацій;
- забезпечення своєчасного оповіщення своїх працівників про загрозу виникнення або при виникненні надзвичайної ситуації.

Основною особливістю дій малих підприємств при загрозі або виникненні надзвичайних ситуацій є в першу чергу захист персоналу та відвідувачів.

Виходячи з цього, ст. 130 Кодексу цивільного захисту України передбачає, що на підприємствах з чисельністю персоналу 50 осіб і менше розробляються та затверджуються інструкції щодо дій при загрозі або виникненні надзвичайних ситуацій.

Розроблена інструкція не повинна суперечити положенням та вимогам Кодексу цивільного захисту України.

Інструкція розробляється та підписується посадовою особою підприємства з питань цивільного захисту, затверджується керівником підприємства та доводиться до всіх працівників під підпис.

Крім Інструкції, на малому підприємстві розробляється План евакуації при пожежі або загрозі вибуху. Особливо це важливо для тих об’єктів, на території яких може знаходитись значна кількість відвідувачів.

Деякі конкретні заходи, не відображені в нормативних документах підприємства, потребують внесення до посадових інструкцій працівників. Крім того, на малому підприємстві необхідно розробляти й доводити до всіх працівників Порядок цілодобового оповіщення керівництва та працівників у випадку загрози або виникнення надзвичайної ситуації.

Всі працівники підприємства повинні бути навчені діям, чітко знати свої обов’язки та неухильно їх виконувати. Це також стосується адміністрації

малого підприємства, яка в екстремальній обстановці не може приймати помилкові рішення або віддавати необґрунтовані розпорядження.

# ВИСНОВКИ

У роботі дано опис сучасного процесу регулювання даної сфери і позначено основні проблеми, які мають місце бути. Як виявилось, менеджери паролів є сприйнятливими до атак перехоплення даних, а також до відновлення інформації. У деяких менеджерах мастер-пароль зберігається у відкритому вигляді або ключі шифрування містяться в коді програм. Деколи заполучити доступ до інформації, можна за допомогою зараження пристроїв жертви шкідливими програмами. Дані вразливості ставлять під сумнів на рахунок надійності аналізованих менеджерів паролів. У всіх аналізованих менеджерах паролів була відсутня можливість корпоративного використання. Таким чином, була необхідність розробки власного рішення, а саме надійного менеджера паролів з можливістю управління організаціями. Також пропонується структура та реалізація програмного забезпечення для керування паролями.

Веб-сервіс був повністю створений у прогресивному середовищі PHPStorm. Було використано FTP-менеджер, віртуальний хостинг та система курування базами даних для повноцінного функціонування менеджера паролів. Для захисту від несанкціонованого доступу та різноманітних маніпуляцій з відправленими даними користувача, було створено та використано декілька middleware-технологій та валідація усіх запитів. Створена web-платформа дозволяє спростити збереження паролів, доступ до паролів організації працівникам  та всебічно захистити їх від впливів зловмисників. Два способи генерації ключів додадуть користувачеві впевненості в захисті своїх облікових записів.

До переваг такого адаптивного ресурсу можна віднести:

- можливість використання з будь-яких пристроїв;
- можливість керування всіма процесами з адміністративної панелі;
- швидке отримання результатів та перехід на сторінку ресурсу
- відсутність потреби у створені паролів. Це можна зробити за допомогою генератора.

- автоматичне генерування ключів
- функціонал управління організаціями
- функціонал управління підрозділами

В перспективі передбачається створення модульного тестування для покращення організації роботи з розширенням проекту, а саме – його функціоналу. Також заплановано вдосконалення методу зберігання та формування ключа шифрування

# СПИСОК ЛІТЕРАТУРИ

1. Загородна Н. В. Захист персональної інформації в задачах аналізу та обробки великих даних [Електронний ресурс] / Н. В. Загородна, Т. Сачик // Тернопільський національний технічний університет імені Івана Пулюя. – 2019. – Режим доступу до ресурсу: http://elartu.tntu.edu.ua/bitstream/lib/30457/2/IMST_2019_Sachyk_T-Protection_of_personal_information_95.pdf.

2. Барильська С. А. Фактори, які впливають на ефективність тестування програмного забезпечення [Електронний ресурс] / С. А. Барильська, Н. В. Загородна // ТНТУ. – 2017. – Режим доступу до ресурсу: http://elartu.tntu.edu.ua/bitstream/lib/23002/2/CAZST_2017v2_Barylska_S_A-The_influence_factors_on_18.pdf.

3. Обертинюк І. Л. Технології оцінки ризиків інформаційної безпеки відповідно до вітчизняних нормативних документів та міжнародних стандартів [Електронний ресурс] / І. Л. Обертинюк, О. В. Кареліна // ТНТУ. – 2018. – Режим доступу до ресурсу: http://elartu.tntu.edu.ua/bitstream/lib/27141/2/VII_MNTK_2018v2_Obertuniyk_I_L-Technologies_of_assessment_132-133.pdf

4. Загородна Н. В. Аналіз сучасних методів виявлення аномалій та можливих вторгнень в роботі комп'ютерної системи [Електронний ресурс] / Н. В. Загородна, Б. О. Паперовський // ТНТУ. – 2018. – Режим доступу до ресурсу: http://elartu.tntu.edu.ua/bitstream/lib/27008/2/VII_MNTK_2018v2_Paperovskyi_B_H-Analysis_of_modern_methods_144.pdf.

5. Мельник О. С. Метод пошукова оптимізація веб-сайту [Електронний ресурс] / О. С. Мельник, Н. В. Загородна // ТНТУ. – 2015. – Режим доступу до ресурсу: http://elartu.tntu.edu.ua/bitstream/123456789/11047/2/ConfATMT_2015v2_Melnyk_O_S-Search_engine_optimization_34.pdf.

6. Komanduri, S., Shay, R., Kelly, P.G., Mazurek, M.L., Bauer, L., Christin, N., Egelman, S.: Of passwords and people: Measuring the effect of

password-composition policies. In: Proceedings of the Human Factors and Computing Systems, pp. 2595–2604. ACM

7. Florencio, D., Herley, C.: Where do security policies come from? New York, NY, USA (2010).

8. Barton, B.F., Barton, M.S.: User-friendly password methods for computer-mediated information systems. Comput. Secur. 3(3), 186–195 (1984)

9. D. Akhawe, A. Barth, P. E. Lam, J. Mitchell, and D. Song. Towards a formal foundation of web security. In Proceedings of the 23rd IEEE Computer Security Foundations Symposium, 2010.

10. K. Bhargavan and A. Delignat-Lavaud. Web-based attacks on host-proof encrypted storage. In Proc. of WOOT, 2012.

11. M. Blanchou and P. Youn. Password managers: Exposing passwords everywhere, Nov 2013

12. M. Rochkind. Security, forms, and error handling. In Expert PHP and MySQL, pages 191–247. Springer, 2013.

13. Christina W. Master Passwords At Risk in LastPass Security Breach [ALERT] [Електронний ресурс] / Warren Christina. – 2011. – Режим доступу до ресурсу: https://mashable.com/2011/05/05/last-pass-breach/.

14. Охорона праці та безпека в надзвичайних ситуаціях [Електронний ресурс] – Режим доступу до ресурсу: http://inmad.vntu.edu.ua/portal/static/60D6D00C-3419-4907-9009-0343E953423B.pdf.

15. НАКАЗ Про затвердження Методичних вказівок "Критерії обґрунтування необхідності і визначення черговості розробки гігієнічних нормативів шкідливих речовин у повітрі робочої зони, атмосферному повітрі населених м [Електронний ресурс]. – 2004. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/v0369282-04#Text.

16. НАКАЗ Про затвердження Загальних вимог стосовно забезпечення роботодавцями охорони праці працівників [Електронний ресурс]. – 2012. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/z0226-12#Text.

17. Як діяти персоналу підприємства в надзвичайній ситуації [Електронний ресурс]. – 2013. – Режим доступу до ресурсу: https://oppb.com.ua/content/yak-diyati-personalu-pidpriiemstva-v-nadzvichayniy-situaciyi

# ДОДАТКИ

# Додаток А

## Правила валідації даних, відправлених користувачами на сервер

### Правила для форм організацій, категорій, відділів

```
public function rules()
    {
        $rules = [
            'name' => ['required', 'string', 'max:255'],
        ];
        return $rules;
    }
```

### Правила для форми створення ресурсів

```
public function rules()
    {
        $rules = [
            'name' => ['required', 'string', 'max:255'],
            'login' => ['nullable', 'string', 'max:255'],
            'url' => ['nullable', 'string', 'max:255'],
            'password' => ['required', 'string', 'max:255'],
            'category_id' => ['nullable', 'integer'],
            'secret' =>['nullable', 'string', 'max:255'],
            'addition' => ['nullable', 'string', 'max:255'],
        ];

        if(request()->get('secret')) {
            $rules['secret'] = [new CheckSecret(), 'string', 'max:32', 'min:32'];
        }
        return $rules;
    }
```

### Правила для форми редагування ресурсів

```
public function rules()
    {
        $rules = [
```

```
            'name' => ['required', 'string', 'max:255'],
            'login' => ['nullable', 'string', 'max:255'],
            'url' => ['nullable', 'string', 'max:255'],
            'password' => ['nullable', 'string', 'max:255'],
            'category_id' => ['nullable', 'integer'],
            'secret' =>['nullable', 'string', 'max:255'],
            'addition' => ['nullable', 'string', 'max:255'],

        ];

        if(request()->get('secret')) {
            $rules['secret'] = [new CheckSecret(), 'string', 'max:32', 'min:32'];
        }
        return $rules;
    }
```

# Додаток Б

## Лістинг коду для управління ресурсами

```
<?php

namespace App\Http\Controllers;

use App\Enumerators\LinksEnumerator;
use App\Enumerators\UsersEnumerator;
use App\Http\Requests\PasswordsAdd;
use App\Http\Requests\PasswordsUpdate;
use App\Models\Department;
use App\Models\Link;
use App\Models\LinkCategory;
use App\Models\User;
use App\Services\CheckerService;
use App\Services\TableService;
use App\Services\PasswordServices;
use Hash;
use Illuminate\Http\Request;

class PasswordsController extends Controller
{
    /**
     * @return \Illuminate\Contracts\View\Factory|\Illuminate\Contracts\View\View
     */
    public function index()
    {
        $data = [
            'meta' => [
                'pageTitle' => 'Ресурси'
            ],
            'breadcrumb' => [
                'Ресурси' => ''
            ],
        ];
```

```
        return view('site.passwords.index', $data);
    }

    /**
     * @return \Illuminate\Contracts\View\Factory|\Illuminate\Contracts\View\View
     */
    public function addForm(Department $department)
    {
        $data = [
            'meta' => [
                'pageTitle' => 'Додати ресурс'
            ],
            'breadcrumb' => [
                'Ресурси' => route('passwords.index'),
                'Додати ресурс' => ''
            ],
            'categories' => auth()->user()->linkCategories,
            'department' => $department->id ? $department : null,
        ];
        return view('site.passwords.add', $data);
    }

    public function create(PasswordsAdd $request, Link $link, Department $department,
PasswordServices $passwordServices) {
        $link->name = $request->get('name');
        $link->login = $request->get('login') ?? null;
        $link->url = $request->get('url');
        $link->type = LinksEnumerator::TYPE_KEY;
        $link->addition = $request->get('addition') ?? null;

        $key = auth()->user()->standart_key;
        if($request->get('secret')) {
            $link->type = LinksEnumerator::TYPE_SECRET;
            $key = $request->get('secret');
        }
        $link->password = $passwordServices->encrypt($request->get('password'), $key);
        if(!$department->id) {
```

```
            if($request->get('category_id')) {
                $link->category_id = (int)$request->get('category_id');
            }
        }



        $link->user_id = auth()->user()->id;


        $link->save();
        $routeData  = ['link' => $link];
        if($department->id) {
            $link->departments()->attach($department->id);
            $routeData  = ['link' => $link, 'department' => $department];
        }


        return redirect()->route('passwords.editForm', $routeData);
    }


    public function editForm(Request $request, Link $link, Department $department,
PasswordServices $passwordServices)
    {
        if($department->id) {
            $breadcrumb = [
                'Організації' => route('bussinesses.index'),
                $department->bussiness->name => route('bussinesses.show', ['bussiness'
=> $department->bussiness]),
                'Відділ ' . $department->name => route('departments.show',
['department' => $department, 'bussiness' => $department->bussiness]),
                'Редагувати ресурс ' . $link->name => ''
            ];
        } else {
            $breadcrumb = [
                'Ресурси' => route('passwords.index'),
                'Редагувати ресурс ' . $link->name => ''
            ];
        }
        $data = [
            'meta' => [
```

```
                'pageTitle' => 'Редагувати ресурс ' . $link->name
            ],
            'breadcrumb' => $breadcrumb,
            'categories' => auth()->user()->linkCategories,
            'linkCategory' => $link->category ?? '',
            'link' => $link,
            'department' => $department->id ? $department : null,

        ];
        return view('site.passwords.edit', $data);
    }

    public function update(PasswordsUpdate $request, Link $link, Department
$department, PasswordServices $passwordServices) {
        $link->name = $request->get('name');
        $link->url = $request->get('url');
        $link->login = $request->get('login') ?? null;

        $link->addition = $request->get('addition') ?? null;
        if($request->get('password')) {
            $key = auth()->user()->standart_key;
            $link->type = LinksEnumerator::TYPE_KEY;
            if($request->get('secret')) {
                $link->type = LinksEnumerator::TYPE_SECRET;
                $key = $request->get('secret');
            }

            $link->password = $passwordServices->encrypt($request->get('password'),
$key);
        }

        if($request->get('category_id')) {
            $link->category_id = (int)$request->get('category_id');
        } else {
            $link->category_id = null;

        }
```

```
        $routeData  = ['link' => $link];
        if($department->id) {
            $routeData  = ['link' => $link, 'department' => $department];
        }
        $link->save();
        return redirect()->route('passwords.editForm', $routeData);

    }

    public function getDecrypted(Request $request, Link $link, PasswordServices
$passwordServices, CheckerService $checkerService)
    {

        if($link->user->id != auth()->user()->id) {
            return response()->json(['error' => 'Невірні дані']);
        }

        $key = auth()->user()->standart_key;
        if($link->type == LinksEnumerator::TYPE_SECRET) {
            if(!$request->get('secret')) {
                return response()->json(['error' => 'Не вказаний ключ']);
            }
            $key = $request->get('secret');
        }

        if($checkerService->checkSecret($key) == false) {
            return response()->json(['error' => 'Не вірний ключ']);

        }

        $password = $passwordServices->decrypt($link->password, $key);
        return response()->json(['success' => $password]);
    }

    public function getClear(Request $request, Department $department, TableService
$getTableButtons)
    {
```

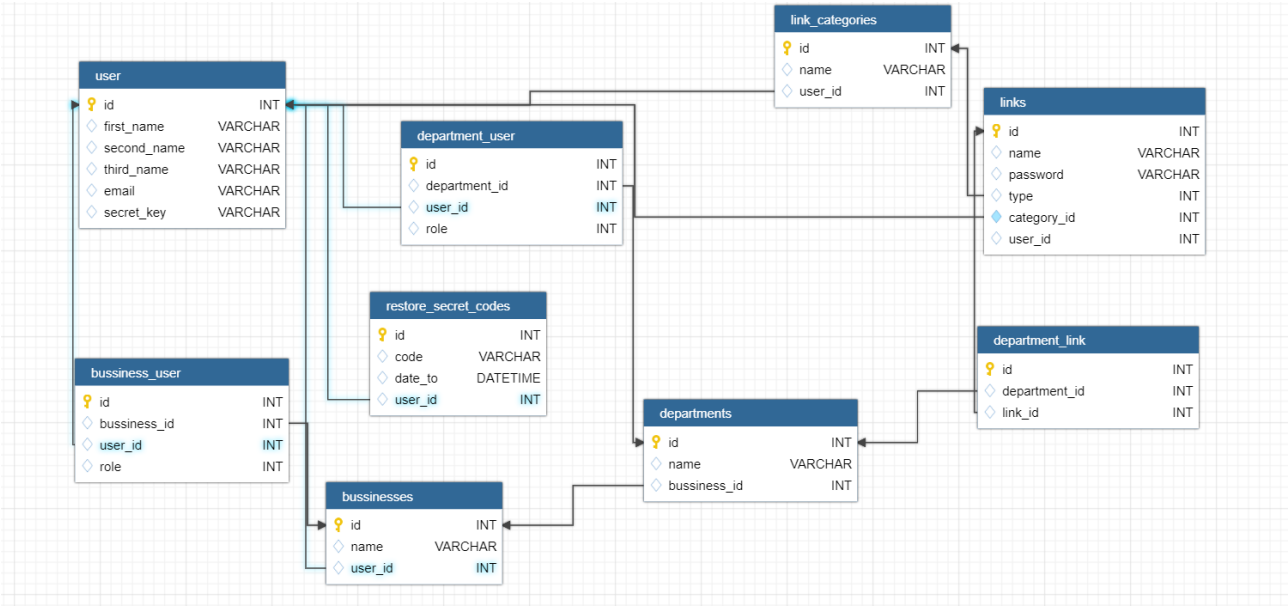
```
        if($department->id) {
            $links = $department->links();
        } else {
            $links = auth()->user()->links()->withoutCategory()->withoutBussiness();
        }
        if($request->get('name')) {
            $links = $links->where('name', 'LIKE', '%'. $request->get('name') . '%');
        }

        if($request->get('url')) {
            $links = $links->where('url', 'LIKE', '%'. $request->get('url') . '%');
        }
        $linksCount = $links->count();

        $links = $links->orderBy('id')->offset($request->get('start'))->limit($request->get('length'))->get();
        $data = [];

        foreach ($links as $link) {
            $dataArray = [
                $link->name,
                $link->url,
                $link->login,
                $link->addition ?? '-',
            ];

            if($department->id) {
                $role =  auth()->user()->departments()->where('department_id', $department->id)->first()->pivot->role;
                if($role == UsersEnumerator::USER_ROLE_ROOT || $role == UsersEnumerator::USER_ROLE_ADMINISTRATOR) {
                    $buttons = [
                        [
                            'type' => 1,
                            'title' => 'Редагувати',
                            'href' => route('passwords.editForm', ['link' => $link, 'department' => $department]),
                        ],[
```

```
                'title' => 'Видалити',
                'id' => $link->id,
                'class' => 'deleteLink',
                'data-attribute-name' => 'data-link',
            ],[
                'title' => 'Скопіювати пароль',
                'id' => $link->id,
                'class' => 'copyPassword',
                'data-attribute-name' => 'data-link',
                'addition-data-types' => [
                    'data-type' => $link->type
                ]
            ]
        ];
    } else {
        $buttons = [
           [
                'title' => 'Скопіювати пароль',
                'id' => $link->id,
                'class' => 'copyPassword',
                'data-attribute-name' => 'data-link',
                'addition-data-types' => [
                    'data-type' => $link->type
                ]
            ]
        ];
    }

} else {
    $buttons = [
        [
            'type' => 1,
            'title' => 'Редагувати',
            'href' => route('passwords.editForm', ['link' => $link->id]),
        ],[
```

```
                    'title' => 'Видалити',
                    'id' => $link->id,
                    'class' => 'deleteLink',
                    'data-attribute-name' => 'data-link',
                ],[
                    'title' => 'Скопіювати пароль',
                    'id' => $link->id,
                    'class' => 'copyPassword',
                    'data-attribute-name' => 'data-link',
                    'addition-data-types' => [
                        'data-type' => $link->type
                    ]
                ]
            ];
        }


        $dataArray[] = $getTableButtons->getButtons($buttons);
        //dd($dataArray);
        $data[] = $dataArray;
    }

    $responseData = [
        'data' => $data,
        'recordsTotal' => $linksCount,
        'recordsFiltered' => $linksCount,
    ];
    return response()->json($responseData);
}

public function delete(Link $link)
{
    $link->delete();

    return response()->json(true);
}
}
```

# Додаток В

## Даталогічна модель бази даних



# Додаток Г

## Лістинг коду контроллерів системи

### Контроллер організацій

```
<?php

namespace App\Http\Controllers;

use App\Enumerators\BussinessesEnumerator;
use App\Enumerators\UsersEnumerator;
use App\Http\Requests\BussinessesRequest;
use App\Models\Bussiness;
use App\Services\CheckerService;
use App\Services\DepartmentService;
use App\Services\TableService;
use Illuminate\Http\Request;

class BussinessesController extends Controller
{
    /**
     * @return \Illuminate\Contracts\View\Factory|\Illuminate\Contracts\View\View
     */
    public function index()
    {
        $data = [
            'meta' => [
                'pageTitle' => 'Організації'
            ],
            'breadcrumb' => [
                'Організації' => ''
            ],
        ];
        return view('site.bussinesses.index', $data);
    }

    /**
     * @return \Illuminate\Contracts\View\Factory|\Illuminate\Contracts\View\View
     */
    public function addForm()
    {
        $data = [
            'meta' => [
                'pageTitle' => 'Додати організацаю'
            ],
            'breadcrumb' => [
                'Організації' => route('bussinesses.index'),
                'Додати організацаю' => ''
            ],
            'bussiness' => null,
        ];
        return view('site.bussinesses.form', $data);
    }

    public function store(BussinessesRequest $request, Bussiness $bussiness) {
        $bussiness->name = $request->get('name');
        if(!$bussiness->id) {
            $bussiness->user_id = auth()->user()->id;
        }
```

```
        $bussiness->save();
        return redirect()->route('bussinesses.editForm', ['bussiness' => $bussiness]);
    }

    public function editForm(Request $request, Bussiness $bussiness)
    {
        $data = [
            'meta' => [
                'pageTitle' => 'Редагувати організацію ' . $bussiness->name
            ],
            'breadcrumb' => [
                'Організації' => route('bussinesses.index'),
                'Редагувати організацію ' . $bussiness->name => ''
            ],
            'bussiness' => $bussiness,
            ];
        return view('site.bussinesses.form', $data);
    }

    public function getApi(Request $request, TableService $getTableButtons)
    {
        if((int)$request->get('type') == BussinessesEnumerator::TYPE_MY) {
            $bussinesses = auth()->user()->bussinesses();

        } else {
            $bussinesses = Bussiness::query()
                ->whereHas('departments', function ($q){
                    $q->whereHas('users', function ($u){
                        $u->where('user_id', auth()->user()->id)->where('role', '<>',
UsersEnumerator::USER_ROLE_ROOT);
                    });
                });
        }
        $bussinessesCount = $bussinesses->count();
        $bussinesses = $bussinesses->orderBy('id')->offset($request->get('start'))-
>limit($request->get('length'))->get();
        $data = [];

        foreach ($bussinesses as $bussiness) {
            $dataArray = [
                $getTableButtons->getUrl($bussiness->name, route('bussinesses.show',
['bussiness' => $bussiness])),
            ];
            if((int)$request->get('type') == BussinessesEnumerator::TYPE_MY) {
                $buttons = [
                    [
                        'type' => 1,
                        'title' => 'Редагувати',
                        'href' => route('bussinesses.editForm', ['bussiness' =>
$bussiness->id]),
                    ],[

                        'title' => 'Видалити',
                        'id' => $bussiness->id,
                        'class' => 'deleteBussiness',
                        'data-attribute-name' => 'data-bussiness',
                    ]
                ];

            } else {
```

```
                $buttons = null;
            }


            $dataArray[] = $buttons ? $getTableButtons->getButtons($buttons) : '-';
            $data[] = $dataArray;
        }

        $responseData = [
            'data' => $data,
            'recordsTotal' => $bussinessesCount,
            'recordsFiltered' => $bussinessesCount,
        ];
        return response()->json($responseData);
    }

    public function delete(Bussiness $bussiness, DepartmentService $departmentService)
    {
        foreach ($bussiness->departments as $department) {
            $departmentService->delete($department);
        }
        $bussiness->delete();

        return response()->json(true);
    }

    public function show(Bussiness $bussiness, CheckerService $checkerService)
    {
        $data = [
            'meta' => [
                'pageTitle' => 'Відділи ' . $bussiness->name
            ],
            'breadcrumb' => [
                'Організації' => route('bussinesses.index'),
                'Відділи ' . $bussiness->name => ''
            ],
            'bussiness' => $bussiness,
            'isOwner' => $checkerService->checkOwnerBussiness(auth()->user()->id,
$bussiness->user_id),
        ];
        return view('site.bussinesses.show', $data);
    }
}
```

## Контроллер відділів

```
<?php

namespace App\Http\Controllers;

use App\Enumerators\BussinessesEnumerator;
use App\Enumerators\UsersEnumerator;
use App\Http\Requests\BussinessesRequest;
use App\Http\Requests\DepartmentsRequest;
use App\Models\Bussiness;
use App\Models\Department;
use App\Models\User;
use App\Services\CheckerService;
use App\Services\DepartmentService;
use App\Services\TableService;
```

```
use Illuminate\Database\Eloquent\Relations\Pivot;
use Illuminate\Http\Request;

class DepartmentsController extends Controller
{
    /**
     * @return \Illuminate\Contracts\View\Factory|\Illuminate\Contracts\View\View
     */
    public function addForm(Bussiness $bussiness)
    {
        $data = [
            'meta' => [
                'pageTitle' => 'Додати відділ'
            ],
            'breadcrumb' => [
                'Організації' => route('bussinesses.index'),
                $bussiness->name => route('bussinesses.show', ['bussiness' =>
$bussiness]),
                'Додати відділ' => ''
            ],
            'bussiness' => $bussiness,
            'department' => null,
        ];
        return view('site.departments.form', $data);
    }

    public function store(DepartmentsRequest $request,
                          Bussiness $bussiness,
                          Department $department,
                          CheckerService $checkerService) {
        $department->name = $request->get('name');
        $create = false;

        if(!$department->id) {
            $create = true;

            if($checkerService->checkOwnerBussiness(auth()->user()->id, $bussiness-
>user_id)) {
                $department->bussiness_id = $bussiness->id;
            }
        }

        $department->save();

        if($create) {
            $department->users()->attach($bussiness->user_id, ['role' =>
UsersEnumerator::USER_ROLE_ROOT]);
        }
        return redirect()->route('departments.editForm', ['bussiness' => $bussiness,
'department' => $department]);
    }

    public function editForm(Request $request, Bussiness $bussiness, Department
$department)
    {
        $data = [
            'meta' => [
                'pageTitle' => 'Редагувати відділ ' . $department->name
            ],
            'breadcrumb' => [
                'Організації' => route('bussinesses.index'),
```

```
                $bussiness->name => route('bussinesses.show', ['bussiness' =>
$bussiness]),
                'Редагувати відділ ' . $department->name => ''
        ],
            'bussiness' => $bussiness,
            'department' => $department,
        ];
        return view('site.departments.form', $data);
    }

    public function getApi(Request $request, Bussiness $bussiness, TableService
$getTableButtons, CheckerService $checkerService)
    {
        $departments = $bussiness->departments()->whereHas('users', function ($q) {
            $q->where('user_id', auth()->user()->id);
        });
        if($request->get('name')) {
            $departments = $departments->where('name', 'LIKE', '%'.$request-
>get('name').'%');

        }

        $departments = $departments->get();
        $departmentsCount = $departments->count();
        $data = [];

        foreach ($departments as $department) {
            $dataArray = [
                $getTableButtons->getUrl($department->name, route('departments.show',
['bussiness' => $bussiness, 'department' => $department])),
            ];
            if($checkerService->checkOwnerBussiness(auth()->user()->id, $bussiness-
>user_id)) {
                $buttons = [
                    [
                        'type' => 1,
                        'title' => 'Редагувати',
                        'href' => route('departments.editForm', ['bussiness' =>
$bussiness, 'department' => $department]),
                    ],[

                        'title' => 'Видалити',
                        'id' => $department->id,
                        'class' => 'deleteDepartment',
                        'data-attribute-name' => 'data-department',
                        'addition-data-types' => ['data-bussiness' => $bussiness->id],
                    ]
                ];

            } else {
                $buttons = null;
            }


            $dataArray[] = $buttons ? $getTableButtons->getButtons($buttons) : '-';
            $data[] = $dataArray;
        }

        $responseData = [
            'data' => $data,
            'recordsTotal' => $departmentsCount,
```

```
            'recordsFiltered' => $departmentsCount,
        ];
        return response()->json($responseData);
    }

    public function delete(Bussiness $bussiness, Department $department,
DepartmentService $departmentService, CheckerService $checkerService)
    {
        if($checkerService->checkOwnerBussiness(auth()->user()->id, $bussiness-
>user_id)) {
            $departmentService->delete($department);
        }
        return response()->json(true);
    }

    public function show(Bussiness $bussiness, Department $department, CheckerService
$checkerService)
    {
        $data = [
            'meta' => [
                'pageTitle' => 'Відділ ' . $department->name
            ],
            'breadcrumb' => [
                'Організації' => route('bussinesses.index'),
                $bussiness->name => route('bussinesses.show', ['bussiness' =>
$bussiness]),
                'Відділ ' . $department->name => ''
            ],
            'bussiness' => $bussiness,
            'department' => $department,
            'role' => $department->users()->where('user_id', auth()->user()->id)-
>first()->pivot->role
        ];
        return view('site.departments.show', $data);
    }

    public function getUsersApi(Request $request, Bussiness $bussiness, TableService
$getTableButtons, Department $department,  CheckerService $checkerService)
    {
        $role = $checkerService->getRole($department, auth()->user());
        if($role == UsersEnumerator::USER_ROLE_ROOT) {
            $users = $department->users();
        } else if($role == UsersEnumerator::USER_ROLE_ADMINISTRATOR) {
            $users = $department->users()->whereHas('departments', function($q) {
                $q->where('department_user.role', UsersEnumerator::USER_ROLE_EMPLOYER);
            });
        }

        $users = $users->where('user_id', '<>', auth()->user()->id);
        if($request->get('name')) {
            $users = $users->where(function ($q) use ($request){
                $q->where('first_name', 'LIKE', '%' . $request->get('name') . '%');
            });
        }
        $users = $users->orderBy('id')->offset($request->get('start'))->limit($request-
>get('length'));
        $usersCount = $users->count();
        $users = $users->get();
        $data = [];

        foreach ($users as $user) {
```

```
            $dataArray = [
                $user->fullName,
                $user->email,
                UsersEnumerator::getTypes()[$user->pivot->role],
            ];
                $buttons = [
                    [
                        'title' => 'Забрати з відділу',
                        'id' => $user->id,
                        'class' => 'deleteUser',
                        'data-attribute-name' => 'data-user',
                        'addition-data-types' => ['data-bussiness' => $bussiness->id,
'data-department' => $department->id],
                    ]
                ];

            if($role == UsersEnumerator::USER_ROLE_ROOT) {
                $buttons[] = [
                    'title' => 'Роль: ' .  UsersEnumerator::getTypes()[$user-
>getRevertRole($user->pivot->role)],
                    'id' => $user->id,
                    'class' => 'changeRole',
                    'data-attribute-name' => 'data-user',
                    'addition-data-types' => ['data-bussiness' => $bussiness->id,
'data-department' => $department->id],
                ];
            }
                $dataArray[] = $buttons ? $getTableButtons->getButtons($buttons) : '-';
            $data[] = $dataArray;
        }

        $responseData = [
            'data' => $data,
            'recordsTotal' => $usersCount,
            'recordsFiltered' => $usersCount,
        ];
        return response()->json($responseData);
    }

    public function deleteUserApi(Bussiness $bussiness, Department $department, User
$user, CheckerService $checkerService)
    {

        $role = $checkerService->getRole($department, $user);

        if($role == UsersEnumerator::USER_ROLE_ADMINISTRATOR) {
            $roleAdmin = $checkerService->getRole($department, auth()->user());
            if($roleAdmin != UsersEnumerator::USER_ROLE_ROOT) {
                return response()->json(false);
            }
        } else if ($role == UsersEnumerator::USER_ROLE_EMPLOYER) {
            $roleAdmin = $checkerService->getRole($department, auth()->user());
            if($roleAdmin == UsersEnumerator::USER_ROLE_EMPLOYER) {
                return response()->json(false);
            }
        } else {
            return response()->json(false);

        }

        $department->users()->detach($user);
```

```
        return response()->json(true);

    }

    public function changeUserRoleApi(Bussiness $bussiness, Department $department,
User $user, CheckerService $checkerService)
    {
        $role = $checkerService->getRole($department, $user);
        $adminRole = $checkerService->getRole($department, auth()->user());
        if($adminRole != UsersEnumerator::USER_ROLE_ROOT) {
            return response()->json(false);
        }

        $pivotUser = $department->users()->where('user_id', $user->id)->first()->pivot;
        if($role == UsersEnumerator::USER_ROLE_EMPLOYER) {
            $pivotUser->role = UsersEnumerator::USER_ROLE_ADMINISTRATOR;
        } else {
            $pivotUser->role =  UsersEnumerator::USER_ROLE_EMPLOYER;
        }

        $pivotUser->save();
        return response()->json(true);
    }

    public function addUserApi(Request $request, Bussiness $bussiness, Department
$department, CheckerService $checkerService) {

        $user = User::where('email', $request->get('email'))->first();
        if(!$user) {
            return response()->json(['error' => 'Користувача не знайдено']);
        }

        $adminRole = $checkerService->getRole($department, auth()->user());
        if($adminRole == UsersEnumerator::USER_ROLE_EMPLOYER) {
            return response()->json(false);
        }

        $pivotUser = $department->users()->where('user_id', $user->id)->first();
        if($pivotUser) {
            return response()->json(['error' => 'Користувача в відділі, ПІП: ' .
$pivotUser->fullName]);
        }

        $department->users()->attach($user->id, ['role' =>
UsersEnumerator::USER_ROLE_EMPLOYER]);

        return response()->json(['success' => 'Користувача добавлено']);

    }
}
```

## Контроллер категорій

```
<?php

namespace App\Http\Controllers;

use App\Enumerators\LinksEnumerator;
use App\Http\Requests\CategoriesRequest;
use App\Http\Requests\PasswordsAdd;
```

```
use App\Models\Bussiness;
use App\Models\Link;
use App\Models\LinkCategory;
use App\Services\LinkService;
use App\Services\PasswordServices;
use App\Services\TableService;
use Illuminate\Http\Request;

class LinkCategoriesController extends Controller
{
    /**
     * @return
\Illuminate\Contracts\Foundation\Application|\Illuminate\Contracts\View\Factory|\Illumi
nate\Contracts\View\View
     */
    public function addForm()
    {
        $data = [
            'meta' => [
                'pageTitle' => 'Додати категорію'
            ],
            'breadcrumb' => [
                'Ресурси' => route('passwords.index'),
                'Додати категорію' => ''
            ],
            'linkCategory' => null,
        ];
        return view('site.categories.form', $data);
    }

    /**
     * @param CategoriesRequest $request
     * @param LinkCategory $linkCategory
     * @return \Illuminate\Http\RedirectResponse
     */
    public function store(CategoriesRequest $request, LinkCategory $linkCategory) {

        $linkCategory->name = $request->get('name');

        if(!$linkCategory->id) {
            $linkCategory->user_id = auth()->user()->id;
        }
        $linkCategory->save();

        return redirect()->route('categories.editForm', ['linkCategory' =>
$linkCategory]);
    }

    /**
     * @param LinkCategory $linkCategory
     * @return
\Illuminate\Contracts\Foundation\Application|\Illuminate\Contracts\View\Factory|\Illumi
nate\Contracts\View\View
     */
    public function editForm(LinkCategory $linkCategory)
    {
        $data = [
            'meta' => [
                'pageTitle' => 'Редагувати категорію ' . $linkCategory->name
            ],
            'breadcrumb' => [
```

```
                'Ресурси' => route('passwords.index'),
                'Редагувати категорію ' . $linkCategory->name => ''
            ],
            'linkCategory' => $linkCategory,
        ];
        return view('site.categories.form', $data);
    }

    /**
     * @param Request $request
     * @param TableService $getTableButtons
     * @return \Illuminate\Http\JsonResponse
     */
    public function getApi(Request $request, TableService $getTableButtons)
    {
        $linkCategories = auth()->user()->linkCategories();

        if($request->get('name')) {
            $linkCategories = $linkCategories->where('name', 'LIKE', '%'. $request-
>get('name') . '%');
        }

        $linkCategoriesCount = $linkCategories->count();

        $linkCategories = $linkCategories->orderBy('id')->offset($request-
>get('start'))->limit($request->get('length'))->get();
        $data = [];

        foreach ($linkCategories as $linkCategory) {
            $dataArray = [
                $getTableButtons->getUrl($linkCategory->name, route('categories.show',
['linkCategory' => $linkCategory])),
            ];
            $buttons = [
                [
                    'type' => 1,
                    'title' => 'Редагувати',
                    'href' => route('categories.editForm', ['linkCategory' =>
$linkCategory->id]),
                ],[

                    'title' => 'Видалити',
                    'id' => $linkCategory->id,
                    'class' => 'deleteLinkCategory',
                    'data-attribute-name' => 'data-linkCategory',
                ]
            ];

            $dataArray[] = $buttons ? $getTableButtons->getButtons($buttons) : '-';
            $data[] = $dataArray;
        }

        $responseData = [
            'data' => $data,
            'recordsTotal' => $linkCategoriesCount,
            'recordsFiltered' => $linkCategoriesCount,
        ];
        return response()->json($responseData);
    }

    /**
```

```
     * @param Request $request
     * @param LinkCategory $linkCategory
     * @param TableService $getTableButtons
     * @return \Illuminate\Http\JsonResponse
     */
    public function getLinksApi(Request $request, LinkCategory $linkCategory,
TableService $getTableButtons)
    {
        $links = $linkCategory->links();

        if($request->get('name')) {
            $links = $links->where('name', 'LIKE', '%'. $request->get('name') . '%');
        }

        if($request->get('url')) {
            $links = $links->where('url', 'LIKE', '%'. $request->get('url') . '%');
        }

        $linksCount = $links->count();

        $links = $links->orderBy('id')->offset($request->get('start'))->limit($request-
>get('length'))->get();
        $data = [];

        foreach ($links as $link) {
            $dataArray = [
                $link->name,
                $link->url,
                $link->login,
                $link->addition ?? '-',
            ];
            $buttons = [
                [
                    'type' => 1,
                    'title' => 'Редагувати',
                    'href' => route('passwords.editForm', ['link' => $link->id]),
                ],[

                    'title' => "Від'єднати",
                    'id' => $link->id,
                    'class' => 'detachLinkFromCategory',
                    'data-attribute-name' => 'data-detachLinkCategory',
                ]
            ];

            $dataArray[] = $buttons ? $getTableButtons->getButtons($buttons) : '-';
            $data[] = $dataArray;
        }

        $responseData = [
            'data' => $data,
            'recordsTotal' => $linksCount,
            'recordsFiltered' => $linksCount,
        ];
        return response()->json($responseData);
    }

    /**
     * @param LinkCategory $linkCategory
```

```
     * @return
\Illuminate\Contracts\Foundation\Application|\Illuminate\Contracts\View\Factory|\Illumi
nate\Contracts\View\View
     */
    public function show(LinkCategory $linkCategory)
    {
        $data = [
            'meta' => [
                'pageTitle' => 'Категорія ' . $linkCategory->name
            ],
            'breadcrumb' => [
                'Категорії' => route('passwords.index'),
            ],
            'linkCategory' => $linkCategory,
        ];
        return view('site.categories.show', $data);
    }

    /**
     * @param LinkCategory $linkCategory
     * @param LinkService $linkService
     * @return \Illuminate\Http\JsonResponse
     * @throws \Exception
     */
    public function delete(LinkCategory $linkCategory, LinkService $linkService)
    {
        foreach ($linkCategory->links as $link) {

            $linkService->detachLinkFromCategory($link);
        }
        $linkCategory->delete();

        return response()->json(true);
    }

    /**
     * @param Link $link
     * @param LinkService $linkService
     * @return \Illuminate\Http\JsonResponse
     */
    public function detachCategory(Link $link, LinkService $linkService)
    {

        $linkService->detachLinkFromCategory($link);

        return response()->json(true);
    }
}
```

## Контроллер користувача

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class UserController extends Controller
{
    /**
     * @param Request $request
```

```
     * @return \Illuminate\Http\JsonResponse
     */

    public function changeName(Request $request)
    {
        $user = auth()->user();
        if(!$user) {
            return response()->json(['error' => 'Помилка']);
        }

        $firstName = $request->get('first_name') ?? null;
        $secondName = $request->get('second_name') ?? null;
        $thirdName = $request->get('third_name') ?? null;
        if(!$firstName) {
            return response()->json(['error' => 'Введіть прізвище адміністратора']);
        }

        if(!$thirdName) {
            return response()->json(['error' => 'Введіть по батькові адміністратора']);
        }

        if(!$secondName) {
            return response()->json(['error' => 'Введіть ім\'я адміністратора']);
        }

        $user->first_name = $firstName;
        $user->second_name = $secondName;
        $user->third_name = $thirdName;


        if($user->save()) {
            return response()->json(['success' => 'Дані змінено', 'first_name' =>
$firstName, 'second_name' => $secondName,  'third_name' => $thirdName]);
        }

    }
}
```