Автор:
Магда Матвій Олексійович
студент 21 КНм групи
Науковий керівник:
кандидат педагогічних наук, доцент
доцент кафедри комп'ютерної та
програмної інженерії
Галицький Олександр Вадимович

РЕАЛІЗАЦІЯ ПРОГРАМНОЇ СИСТЕМИ СЕКРЕТНИХ ДАНИХ

Анотація: Останні роки загроза розкриття конфіденційних відомостей для звичайних користувачів та компаній стає все більш актуальною. Основна небезпека полягає в витоку даних облікових записів та ключів або паролів, проте й інші секретні дані теж стають об'єктом потенційної загрози. З метою забезпечення безпеки конфіденційних відомостей для широкого кола користувачів в умовах кібербезпеки, необхідно інтегрувати ефективні рішення та інструменти. Важливим аспектом цього процесу є використання соціальних мереж та месенджерів для зручної адаптації та впровадження цих засобів безпеки.

Ключові слова: відомості, конфіденційні відомості, ключ, пароль, програма, менеджмент, кібербезпека, інтеграція.

Вступ. Рівні загроз у сфері кібербезпеки постійно змінюються, а разом з ними розвиваються відповідні заходи та інструменти безпеки.

Однією з ключових проблем у цьому контексті є недотримання правил та рекомендацій, пов'язаних із створенням, зберіганням та управлінням критично важливою конфіденційними відомостями. Це дані облікових записів та паролі, ключі доступу до ресурсів, сертифікати валідації та інші конфіденційні відомості. Ця проблема досить актуальна як для індивідуальних користувачів, так і для корпорацій та урядових структур. Ігнорування цих правил може призвести до втрати критично важливих даних, призводячи до значних фінансових збитків та безповоротної втрати інформації.

Для часткового вирішення цієї проблеми достатньо дотримуватись простих рекомендацій. Але для повного захисту рекомендується використовувати спеціально призначені для цього сервіси, так звані менеджери паролів або менеджери секретів [5].

Мета роботи полягає в наступному: створити програмну систему для створення, безпечного зберігання та керування обліковими даними. Створити програмний інтерфейс [2] для можливості інтегрування системи із будь-якими соціальними мережами (месенджерами). Інтегрувати розроблену систему у популярний месенджер Telegram для залучення побутових користувачів.

Виклад основного матеріалу. Розроблена система складається із шести проектів, що знаходяться під одним програмним рішенням (Solution). Створено рішення, що називається PasswordManager. Під цим рішенням створено наступні проекти:

- ✓ *PasswordManager.Web* – проєкт-мікросервіс, що отримує запити із-зовні, та в подальшому адресує їх відповідним мікросервісам. Проєкт створено на основі шаблону ASP NET Core AWS Lambda для .NET;
- ✓ *PasswordManager.Common* – проєкт, що має у собі набір загальних утиліт. Проєкт створено на основі шаблону Class Library;
- ✓ *PasswordManager.Core* – проєкт, що має у собі моделі сутностей бази даних та абстракції репозиторів для роботи з ними. Проект створено на основі шаблону Class Library;
- ✓ *PasswordManager.Infrastructure* – проєкт, що має у собі реалізацію репозиторіїв з проєкту PasswordManager.Core на основі використання Entity Framework Core. Проєкт створено на основі шаблону Class Library;

✓ *PasswordManager.Application* – проєкт-мікросервіс, що має у собі абстракції та реалізацію бізнес-шару програми, тобто прошарку між користувацьким інтерфейсом або користувацькими запитами та базою даних (із використанням абстракцій проекту PasswordManager.Core). Проект створено на основі шаблону AWS Lambda для .NET;

✓ *PasswordManager.Bot* – проєкт-мікросервіс, що має у собі реалізацію користувацького інтерфейсу через Telegram Bot API. Проект створено на основі шаблону AWS Lambda для .NET.

Для реалізації функціоналу використані такі основні бібліотеки:

✓ *BouncyCastle.NetCore* – бібліотека, із використанням якої реалізовано симетричне шифрування за алгоритмом шифрування AES-256.

✓ *Telegram.Bot* – бібліотека, із використанням якої реалізовано користувацький інтерфейс взаємодії із месенджером Telegram через Telegram Bot API.

✓ *Microsoft.EntityFrameworkCore* – бібліотека, що має у собі Entity Framework Core та інструменти для роботи з ним, наприклад драйвер бази даних MS SQL Servier.

✓ *Serilog* – бібліотека, що має у собі потужний функціонал логування подій.

Для локалізації програми було створено два текстові файли у форматі JSON із англомовним та україномовним інтерфейсом.

Розглянимо функції системи для *Користувача*. Користувач – це людина, яка користується програмою за допомогою інтерфейсу користувача на прикладі чат-інтерфейсу месенджеру Telegram. Користувачу доступні наступні функції: отримати довідку з навігації та користування системою (команда /help); додати обліковий запис (команда /add); відмінити виконання поточної дії (команда /cancel); видалити обліковий запис (кнопка Видалити у меню облікового запису); зашифрувати пароль (кнопка Зашифрувати із подальшим введенням ключу шифрування); розшифрувати пароль (введення ключу шифрування); згенерувати пароль або ключ (кнопка у сценарному меню при додаванні облікового запису або команда /generate або команда /key); шукати облікові записи (введення пошукового запиту при відсутності інших поточних дій стану); змінити мову інтерфейсу (команда /language або перший запуск програми); налаштувати параметри генератору паролю (команда /generator); показати усі облікові записи користувача (команда /all); переглянути дані облікового запису (обрати відповідний обліковий запис після пошуку); показати пароль (кнопка Пароль у меню облікового запису); оновити дані облікового запису (кнопка Редагувати у меню облікового запису).

Розглянемо функції системи для *Адміністратора*. Адміністратор – це людина, яка має усі права Користувача та ще додаткові права для адміністрування системи. Адміністратору доступні наступні функції: додати користувача у систему (команда /addUser); видалити користувача із системи (команда /deleteUser); показати список користувачів системи (команда /userList).

Проєкт програми було створено за архітектурою програмного проекту типу Onion. Onion-подібна архітектура складається з центральної доменної області (Domain model), що містить моделі доменних сутностей. Навколо центральної області знаходиться шар Domain Services, що містить абстракції репозиторіїв для роботи з доменними сутностями. Навколо цього шару знаходиться шар бізнес-логіки (Application services) що взаємодіє із абстракціями репозиторіїв створюючи відповідні обгортки та додаючи додатковий функціонал безпосередньо для функцій сервісу. А вже навколо усіх цих шарів паралельно знаходяться шари: інфраструктури програми, інтерфейсу користувача, сервісів третіх сторін та тестів. Інфраструктура програми в тому числі містить реалізацію абстракцій репозиторіїв для роботи із сутностями. Тобто при зміні однієї інфраструктури на іншу бізнес-логіка сервісу не зміниться, бо шар Application services взаємодіє лише із абстракціями репозиторіїв, використовуючи принцип SOLID: Dependency inversion. Таким чином шар за шаром архітектура нагадує шари цибулі, звідки і береться її назва.

Проєкт побудовано за допомогою мікросервісної архітектури. На відміну від монолітної архітектури, мікросервісна архітектура дозволяє гнучко налаштовувати не тільки

вертикальне масштабування сервісу (тобто додавання більшої потужності до серверу), а й горизонтальне масштабування – створення додаткових вузлів, кластерів, серверів одного й того самого сервісу за необхідності, що дозволить правильно розподіляти трафік сервісу при його великій навантаженності.

Також слід зазначити, що використання мікросервісної архітектури дозволяє гнучко підтримувати та розробляти проект, роблячи зміни лише у необхідних сервісах, не чіпаючи роботу стабільно працюючих. Так як, зазвичай мікросервіси спілкуються між собою стандартизованими запитами, наприклад через HTTP API або передаючи запити безпосередньо через інфраструктуру хмарного обчислювального середовища такого як AWS – вони можуть бути написані на різних платформах та мовах програмування. Мікросервісна архітектура дозволяє постійно додавати у проект нові модулі. Наприклад, у випадку даного проекту система може доповнюватись новими сервісами для інтерфейсу користувача: можна додати сервіс Web API для комунікації із десктопними та мобільними додатками, додати веб-сайт, що буде комунікувати безпосередньо із сервісом PasswordManager.Application та інші.

У проєкті використовується схема сутностей для бази даних, що складається із двох сутностей: Користувач (User) та обліковий запис (Account).

Користувач має наступні атрибути: Id (первинний ключ) – унікальний ідентифікатор; Language – мова інтерфейсу; PasswordGenerationPattern – налаштування генератору паролів; Action – поточна виконувана дія; KeyHint – підказка для ключа шифрування; OutdatedTime – проміжок часу, після якого пароль вважається застарілим за замовчуванням.

Обліковий запис має наступні атрибути: Id (первинний ключ) – унікальний ідентифікатор; UserId - ключ на відповідний запис у таблиці користувачів; AccountName – назва облікового запису; Link – посилання на сервіс; Login – обліковий логін; Password – пароль; PasswordUpdatedDate – час останнього оновлення паролю; Note – додаткова нотатка по запису; Encrypted – маркер на те чи пароль зашифровано; OutdatedTime – проміжок часу, після якого пароль вважається застарілим (індивідуально для облікового запису).

Увесь код системи протестовано модульними та інтеграційними тестами. Для розробки використовувався Test Driven Development (TDD) – підхід, за якого спочатку створюються неробочі тести, після чого пишеться код модулю, поки тести не почнуть проходити, потім пишуться нові тести і так далі.

Висновок. Для забезпечення безпеки своїх облікових даних необхідно використовувати рішення та інструменти для менеджменту секретними даними. Для інтегрування таких рішень у маси можна інтегрувати рішення у соціальні мережі, що дозволить побутовим користувачам простіше адаптуватися до них

Список літератури

1. Armstrong D. The Quarks of Object-Oriented Development / Deborah J Armstrong., 2006. – 245 с.
2. Lapinskyi V.V., Mykytenko P.V., Halytskyi O.V. (2021). Design of medical information systems user interface. *Information Technologies and Learning Tools, 85(5), 1-13*. https://doi.org/10.33407/itlt.v85i5.44072
3. Paul K. Microsoft open-sources Entity Framework / Krill Paul, 2012., - 54 с.
4. SafetyDetectives. The 20 Most Hacked Passwords in the World: Is Yours Here?.SafetyDetectives. URL: https://www.safetydetectives.com/blog/the-most-hacked-passwords-in-the-world/
5. Магда М. (2023). Програмна система для менеджменту секретних даних. Інформаційно-комунікаційні технології в освіті, (10). вилучено із https://e journals.udu.edu.ua/index.php/ikt/article/view/1303.
6. Kogent Solutions Inc. ASP.NET 3.5 Black Book, 2009. - 134 с.