

27 MAJA 2025

PATRYK ZAWADZKI, KUBA ZAŁUSKA, KAROLINA TRZASKULSKA, KUBA ZALEWSKI, PATRYK WELKIER



MANAGER HASEŁ

Projekt z bezpieczeństw usług chmurowych

Spis treści

1	Wstęp.....	3
2	Opis projektu	4
2.1	Funkcjonalności:	4
2.2	Baza	7
2.3	Endpointy api:	11
3	Technologie:	20
4	Bezpieczeństwo	21
4.1	Proces wdrożenia.....	26
4.2	Rola postgres:.....	28
4.3	Rola node_app:.....	29
4.4	Rola domain:	29
4.5	Rola apparmor:.....	30
4.6	Frontend:	31
4.7	Stworzenie Blob Storage	33
5	Backup.....	35
6	Testy	37
7	Instrukcja użytkowania	45
7.1	Użycie rozszerzenia.....	57
8	Co można zmienić aby aplikacja nadawała się do środowiska produkcyjnego ...	60
7	Regulamin aplikacji.....	62
9	Polityka prywatności	64
10	Logi	66
11	Podsumowanie	67
12	Dokumentacja.....	68
13	Bibliografia.....	68

1 Wstęp

Celem projektu było zaprojektowanie i wdrożenie bezpiecznego menedżera haseł, umożliwiającego użytkownikom przechowywanie, zarządzanie oraz bezpieczne wykorzystywanie haseł w aplikacjach webowych. Aplikacja składa się z backendu opartego na Node.js, który dostarcza API REST do zarządzania hasłami, bazy danych PostgreSQL do bezpiecznego przechowywania zaszyfrowanych danych, frontendu napisanego w React z TypeScript oraz rozszerzenia dla przeglądarki Chrome (również w React z TypeScript), umożliwiającego logowanie do aplikacji webowych oraz edytowanie haseł. System został wdrożony w środowisku chmury Microsoft Azure, wykorzystując maszynę wirtualną do hostowania backendu i bazy danych oraz Azure Static Web Apps do hostowania frontendu, co zapewnia minimalizację kosztów przy zachowaniu wysokiej dostępności. Rozszerzenie Chrome jest instalowane przez użytkownika bezpośrednio z repozytorium. Projekt kładzie szczególny nacisk na bezpieczeństwo, stosując szyfrowanie danych (AES-256), bezpieczne protokoły komunikacyjne (TLS 1.2, zgodnie z normami ISO/IEC 27017 i 27018 oraz wymogami GDPR. W skład projektu wchodzi repozytoria:

<https://github.com/patrykzawadzki/ggw/SecureBox-frontend.git>

<https://github.com/JustNormalProgrammer/SecureBox.git>

<https://github.com/patrykzawadzki/ggw/securebox-ansible.git>

2 Opis projektu

2.1 Funkcjonalności:

- Backend (Node.js): API REST do zarządzania hasłami, uwierzytelnianie, szyfrowanie, przechowuje zaszyfrowane hasła w plikach tekstowych w folderach użytkowników, przechowuje dane w bazie danych
- Frontend (React): Interfejs użytkownika do dodawania, edycji i usuwania haseł, resetu hasła, eksportu i importu haseł, generowania hasła, monitorowania jakości haseł i aktywności użytkownika na poszczególnych użytkownikach.
- Rozszerzenie Chrome (React): Szybki dostęp do haseł z poziomu przeglądarki umożliwia użytkownikowi zalogowanie się do różnych aplikacji webowych, zapisanie haseł bądź ich aktualizacje oraz wygenerowanie silnego hasła

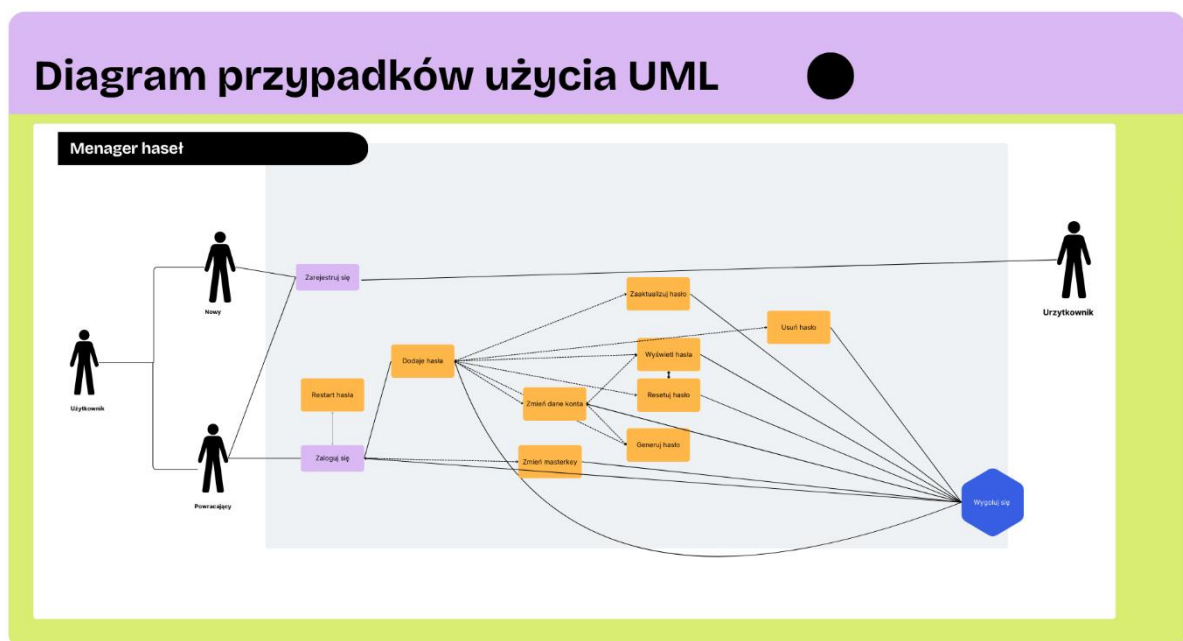


Diagram klas

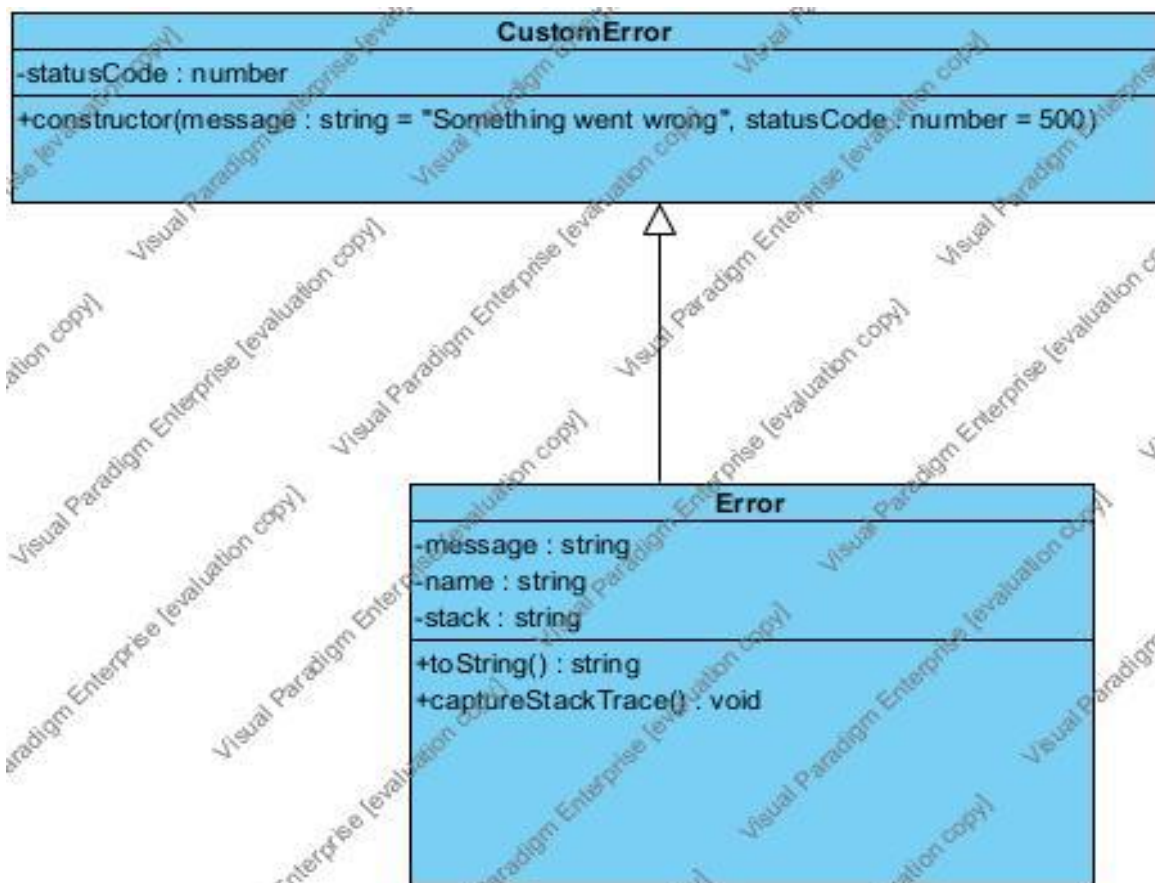
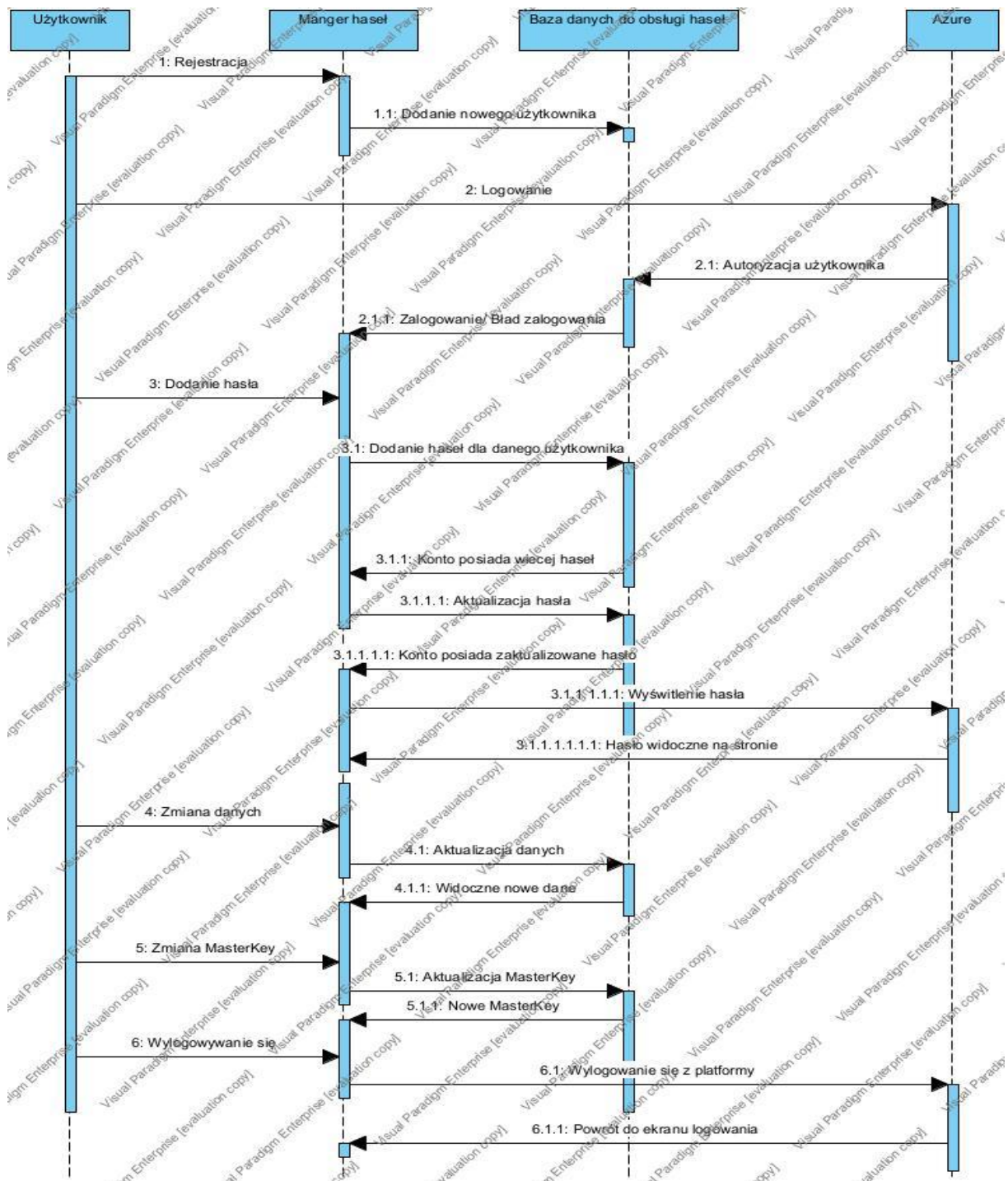
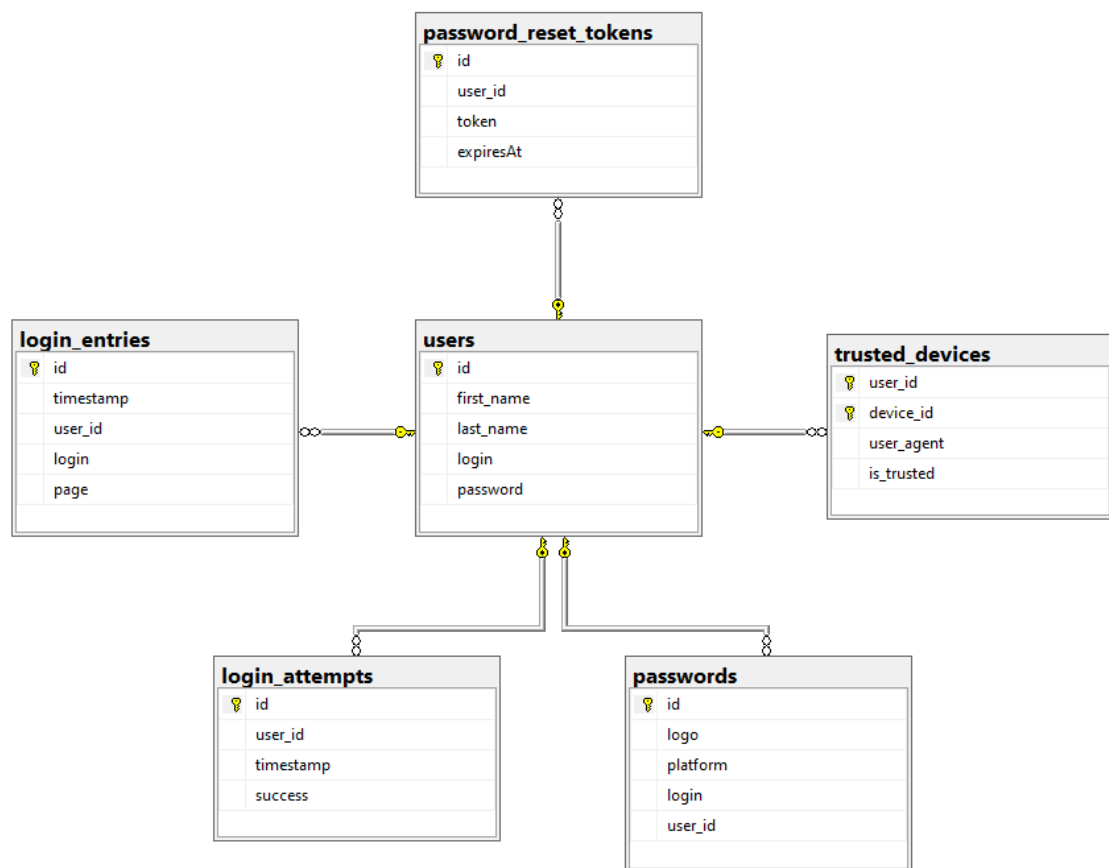


Diagram sekwencji



2.2 Baza

Baza danych aplikacji menedżera haseł została zaprojektowana w PostgreSQL i wdrożona na maszynie wirtualnej w Microsoft Azure przy użyciu skryptu Ansible. Schemat składa się z pięciu tabel: `users`, `passwords`, `login_entries`, `trusted_devices` oraz `password_reset_tokens`, z dodatkową tabelą `login_attempts`, które razem wspierają funkcjonalność aplikacji, zapewniając bezpieczne przechowywanie haseł, zarządzanie dostępem i audyt działań użytkowników. Struktura bazy danych została zaprojektowana z uwzględnieniem zasad bezpieczeństwa, minimalizacji danych i zgodności z regulacjami GDPR oraz normami ISO/IEC 27017 i 27018.



2.2.1 Tabela users

Cel: Przechowuje dane użytkowników aplikacji menedżera haseł.

Pola:

- id (text, klucz główny): Unikalny identyfikator użytkownika.
- first_name (text, niepusty): Imię użytkownika.
- last_name (text, niepusty): Nazwisko użytkownika.
- login (text, niepusty, unikalny): Unikalna nazwa logowania użytkownika.
- password (text, niepusty): Hasło użytkownika, zaszyfrowane (np. za pomocą SCRAM-SHA-256, skonfigurowane w roli postgres skryptu Ansible).

Zabezpieczenia:

- Unikalność pola login zapewnia ograniczenie UNIQUE.
- Hasła są szyfrowane zgodnie z konfiguracją PostgreSQL (password_encryption = scram-sha-256), co wspiera zgodność z GDPR.

Zastosowanie: Podstawa uwierzytelniania użytkowników w aplikacji i rozszerzeniu Chrome.

2.2.2 Tabela passwords

Cel: Przechowuje zaszyfrowane hasła użytkowników dla różnych platform.

Pola:

- id (text, klucz główny): Unikalny identyfikator hasła.
- logo (text, niepusty): Adres URL lub identyfikator logo platformy.
- platform (text, niepusty): Nazwa platformy (np. "www.google.com").
- login (text, niepusty): Login do platformy.
- user_id (text, niepusty): Klucz obcy odnoszący się do users(id).

Zabezpieczenia:

- Klucz obcy user_id z opcją ON DELETE CASCADE zapewnia usuwanie haseł przy usunięciu użytkownika, wspierając zgodność z GDPR (prawo do bycia zapomnianym).

Zastosowanie: Przechowywanie haseł dla frontendu React i rozszerzenia Chrome, z dostępem ograniczonym do uwierzytelnionego użytkownika.

2.2.3 Tabela login_entries

Cel: Rejestruje historię logowań użytkowników dla celów audytu.

Pola:

- id (integer, klucz główny, generowany automatycznie): Unikalny identyfikator wpisu logowania.
- timestamp (timestamp with time zone, niepusty): Czas logowania.
- user_id (text, niepusty): Klucz obcy odnoszący się do users(id).
- login (text, niepusty): Login użytkownika.
- page (text, niepusty): Strona, na której dokonano logowania (np. adres URL).

Zabezpieczenia:

- Klucz obcy user_id z opcją ON DELETE CASCADE.
- Pole timestamp z czasem w strefie czasowej wspiera precyzyjny audyt.

Zastosowanie: Umożliwia śledzenie aktywności logowania, wspierając zarządzanie incydentami bezpieczeństwa (ISO/IEC 27017).

2.2.4 Tabela trusted_devices

Cel: Przechowuje informacje o zaufanych urządzeniach użytkownika.

Pola:

- user_id (text, niepusty): Klucz obcy odnoszący się do users(id).
- device_id (text, niepusty): Unikalny identyfikator urządzenia.
- user_agent (text, niepusty): Informacje o przeglądarce/urządzeniu.
- is_trusted (integer, niepusty): Flaga wskazująca, czy urządzenie jest zaufane (np. 1 = zaufane, 0 = niezaufane).

Klucz główny: Kombinacja user_id i device_id.

Zabezpieczenia:

- Klucz obcy user_id z opcją ON DELETE CASCADE.
- Dane urządzenia są minimalizowane, co wspiera zasadę minimalizacji danych GDPR.

Zastosowanie: Umożliwia weryfikację urządzeń podczas logowania, zwiększając bezpieczeństwo aplikacji.

2.2.5 Tabela password_reset_tokens

Cel: Przechowuje tokeny do resetowania haseł.

Pola:

- id (text, klucz główny): Unikalny identyfikator tokenu.
- user_id (text, niepusty): Klucz obcy odnoszący się do users(id).
- token (text, niepusty): Token resetowania hasła.
- expiresAt (timestamp with time zone, niepusty): Data wygaśnięcia tokenu.

Zabezpieczenia:

- Klucz obcy user_id z opcją ON DELETE CASCADE.
- Ograniczenie czasowe tokenów (expiresAt) minimalizuje ryzyko nieautoryzowanego użycia.

Zastosowanie: Umożliwia bezpieczne resetowanie haseł przez użytkownika.

2.2.6 Tabela login_attempts

Cel: Rejestruje próby logowania (udane i nieudane) dla celów audytu i wykrywania ataków.

Pola:

- id (text, klucz główny): Unikalny identyfikator próby logowania.
- user_id (text, niepusty): Klucz obcy odnoszący się do users(id).
- timestamp (timestamp with time zone, niepusty): Czas próby logowania.
- success (boolean, niepusty): Flaga wskazująca, czy logowanie było udane.

Zabezpieczenia:

- Klucz obcy user_id z opcją ON DELETE CASCADE.
- Rejestracja prób logowania wspiera detekcję ataków brute-force (integracja z fail2ban w roli postgres).

Zastosowanie: Umożliwia monitorowanie i analizę prób logowania, wspierając zarządzanie incydentami bezpieczeństwa.

2.3 Endpointy api:

POST /login

Opis: Uwierzytelnia użytkownika i zwraca token JWT.

Żądanie:

Body:

login (string): Login użytkownika.

password (string): Hasło użytkownika.

Odpowiedź:

Body:

user (object):

id (string): ID użytkownika.

first_name (string): Imię użytkownika.

last_name (string): Nazwisko użytkownika.

login (string): Login użytkownika.

token (string): Token JWT.

2.3.1 User Management Endpoints

POST /users

Opis: Tworzy nowego użytkownika.

Żądanie:

Body:

first_name (string): Imię użytkownika.

last_name (string): Nazwisko użytkownika.

login (string): Login użytkownika.

password (string): Hasło użytkownika.

Odpowiedź:

Body:

id (string): ID nowo utworzonego użytkownika.

first_name (string): Imię użytkownika.

last_name (string): Nazwisko użytkownika.

login (string): Login użytkownika.

PATCH /users/:user_id

Opis: Aktualizuje dane użytkownika.

Żądanie:

Parametry:

user_id (string): ID użytkownika.

Body:

first_name (string, opcjonalne): Nowe imię użytkownika.

last_name (string, opcjonalne): Nowe nazwisko użytkownika.

login (string, opcjonalne): Nowy login użytkownika.

password (string, opcjonalne): Nowe hasło użytkownika.

Odpowiedź:

Body:

id (string): ID użytkownika.

first_name (string): Imię użytkownika.

last_name (string): Nazwisko użytkownika.

login (string): Login użytkownika.

GET /users/:user_id

Opis: Pobiera dane użytkownika.

Żądanie:

Parametry:

user_id (string): ID użytkownika.

Odpowiedź:

Body:

id (string): ID użytkownika.

first_name (string): Imię użytkownika.

last_name (string): Nazwisko użytkownika.

login (string): Login użytkownika.

GET /users/me/get

Opis: Pobiera dane zalogowanego użytkownika.

Żądanie: Brak.

Odpowiedź:

Body:

id (string): ID użytkownika.

first_name (string): Imię użytkownika.

last_name (string): Nazwisko użytkownika.

login (string): Login użytkownika.

GET /users/:user_id/logins

Opis: Pobiera historię logowań użytkownika.

Żądanie:

Parametry:

user_id (string): ID użytkownika.

Odpowiedź:

Body:

logins (array): Lista logowań użytkownika.

timestamp (string): Czas logowania.

user_id (string): ID użytkownika.

login (string): Login użytkownika.

page (string): Strona, na którą zalogowano.

POST /users/:user_id/logins

Opis: Dodaje wpis do historii logowań użytkownika.

Żądanie:

Parametry:

user_id (string): ID użytkownika.

Body:

login (string): Login użytkownika.

page (string): Strona, na którą zalogowano.

Odpowiedź:

Body:

timestamp (string): Czas logowania.

user_id (string): ID użytkownika.

login (string): Login użytkownika.

page (string): Strona, na którą zalogowano.

2.3.2 Password Management Endpoints

GET /passwords

Opis: Pobiera wszystkie hasła użytkownika.

Żądanie: Brak.

Odpowiedź:

Body:

passwords (array): Lista haseł użytkownika.

id (string): ID hasła.

passwordfile (string): Nazwa pliku z hasłem.

logo (string): Logo platformy.

platform (string): Nazwa platformy.

login (string): Login użytkownika.

user_id (string): ID użytkownika.

GET /passwords/:user_id/files

Opis: Pobiera wszystkie pliki z hasłami użytkownika w formacie ZIP.

Żądanie:

Parametry:

user_id (string): ID użytkownika.

Odpowiedź: Plik ZIP zawierający wszystkie pliki z hasłami użytkownika.

POST /passwords/:user_id/files

Opis: Dodaje nowe hasło użytkownika.

Żądanie:

Parametry:

user_id (string): ID użytkownika.

Body:

password (string): Hasło użytkownika.

platform (string): Nazwa platformy.

login (string): Login użytkownika.

logo (string): Logo platformy.

Odpowiedź:

Body:

id (string): ID hasła.

passwordfile (string): Nazwa pliku z hasłem.

logo (string): Logo platformy.

platform (string): Nazwa platformy.

login (string): Login użytkownika.

user_id (string): ID użytkownika.

PUT /passwords/:user_id/passwords/:platform/:login

Opis: Aktualizuje hasło użytkownika.

Żądanie:

Parametry:

user_id (string): ID użytkownika.

platform (string): Nazwa platformy.

login (string): Login użytkownika.

Body:

new_password (string): Nowe hasło użytkownika.

Odpowiedź:

Body:

id (string): ID hasła.

passwordfile (string): Nazwa pliku z hasłem.

logo (string): Logo platformy.

platform (string): Nazwa platformy.

login (string): Login użytkownika.

user_id (string): ID użytkownika.

DELETE /passwords/:user_id/passwords/:platform/:login

Opis: Usuwa hasło użytkownika.

Żądanie:

Parametry:

user_id (string): ID użytkownika.

platform (string): Nazwa platformy.

login (string): Login użytkownika.

Odpowiedź:

Body:

message (string): Wiadomość potwierdzająca usunięcie hasła.

PUT /passwords/:user_id/passwords

Opis: Aktualizuje wszystkie hasła użytkownika.

Żądanie:

Parametry:

user_id (string): ID użytkownika.

Body:

passwordsall (array): Lista haseł do zaktualizowania.

platform (string): Nazwa platformy.

login (string): Login użytkownika.

new_password (string): Nowe hasło użytkownika.

Odpowiedź:

Body:

updatedEntries (array): Lista zaktualizowanych haseł.

id (string): ID hasła.

passwordfile (string): Nazwa pliku z hasłem.

logo (string): Logo platformy.

platform (string): Nazwa platformy.

login (string): Login użytkownika.

user_id (string): ID użytkownika.

2.3.3 Trusted Device Management Endpoints

GET /users/:user_id/trusted-devices

Opis: Pobiera listę zaufanych urządzeń użytkownika.

Żądanie:

Parametry:

user_id (string): ID użytkownika.

Odpowiedź:

Body:

devices (array): Lista zaufanych urządzeń użytkownika.

user_id (string): ID użytkownika.

device_id (string): ID urządzenia.

user_agent (string): User agent urządzenia.

is_trusted (boolean): Czy urządzenie jest zaufane.

PATCH /users/:user_id/trusted-devices

Opis: Aktualizuje zaufane urządzenie użytkownika.

Żądanie:

Parametry:

user_id (string): ID użytkownika.

Body:

device_id (string): ID urządzenia.

user_agent (string): User agent urządzenia.

is_trusted (boolean): Czy urządzenie jest zaufane.

Odpowiedź:

Body:

user_id (string): ID użytkownika.

device_id (string): ID urządzenia.

user_agent (string): User agent urządzenia.

is_trusted (boolean): Czy urządzenie jest zaufane.

DELETE /users/:user_id/trusted-devices/:device_id

Opis: Usuwa zaufane urządzenie użytkownika.

Żądanie:

Parametry:

user_id (string): ID użytkownika.

device_id (string): ID urządzenia.

Odpowiedź:

Body:

message (string): Wiadomość potwierdzająca usunięcie urządzenia z listy zaufanych.

3 Technologie:

Backend: archiver, cookie-parser, cors, csrf-csrf, dotenv, drizzle-orm, express, express-async-handler, express-validator, helmet, jsonwebtoken, nodemailer, pg

Frontend: tailwindcss, axios, jszip, lucide-react, react, react-dom, react-google-recaptcha-v3, react-hook-form, react-router-dom, tailwindcss, zod, zxcvbn, typescript

Rozszerzenie chrome: tailwindcss, axios, jszip, lucide-react, react, react-dom, react-google-recaptcha-v3, tailwindcss, zod, zxcvbn, typescript

Środowisko chmurowe: Azure Virtual Machine, konfiguracja sieci, Azure Static Web Apps, Azure Blob Storage

Baza danych: PostgreSQL

4 Bezpieczeństwo

Początkowy stan bezpieczeństwa systemu menedżera haseł, przed wdrożeniem kompleksowych zabezpieczeń, charakteryzował się szeregiem krytycznych podatności, które zostały zidentyfikowane podczas skanowania aplikacji narzędziem Snyk Code Test. Poniżej przedstawiono szczegółową analizę wykrytych problemów, które stanowiły poważne zagrożenie dla poufności, integralności i dostępności danych użytkowników.

```
X [Low] Improper Type Validation
Path: routes/passwords.js, line 123
Info: The type of this object, coming from body and the value of its map property can be controlled by the user. An attacker may craft the properties of the object to crash the application or bypass its logic. Consider checking the type of the object.

X [Medium] Information Exposure
Path: app.js, line 7
Info: Disable X-Powered-By header for your Express app (consider using Helmet middleware), because it exposes information about the used framework to potential attackers.

X [High] Hardcoded Secret
Path: middleware/auth.js, line 3
Info: Hardcoded value is used as a cipher key (in jwt.verify). Generate the value with a cryptographically strong random number generator and do not hardcode it in source code.

X [High] Path Traversal
Path: routes/passwords.js, line 47
Info: Unsantized input from an HTTP parameter flows into fs.promises.writeFile, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to write to arbitrary files.

X [High] Path Traversal
Path: routes/passwords.js, line 74
Info: Unsantized input from an HTTP parameter flows into fs.promises.writeFile, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to write to arbitrary files.

X [High] Path Traversal
Path: routes/passwords.js, line 138
Info: Unsantized input from an HTTP parameter flows into fs.promises.writeFile, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to write to arbitrary files.

X [High] Path Traversal
Path: routes/passwords.js, line 183
Info: Unsantized input from an HTTP parameter flows into fs.promises.unlink, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to delete arbitrary files.
```

Wykryte podatności:

1. X [Low] Improper Type Validation:

- Ścieżka: *routes/passwords.js*, linia 123
- **Opis:** Podatność związana z niewłaściwą walidacją typu obiektu pochodzącego z ciała żądania HTTP. Oznacza to, że atakujący mógłby manipulować właściwościami obiektu oraz wartością jego właściwości map w celu doprowadzenia do awarii aplikacji (np. poprzez wstrzyknięcie nieoczekiwanych typów danych) lub ominięcia logiki biznesowej. Brak odpowiedniej kontroli typu danych wejściowych może prowadzić do nieprzewidzianych zachowań systemu, a w skrajnych przypadkach do przejęcia kontroli nad aplikacją.

2. X [Medium] Information Exposure:

- **Ścieżka:** *app.js*, linia 7
- **Opis:** Ta podatność dotyczy ujawniania informacji o technologii backendu, a konkretnie obecności nagłówka X-Powered-By w odpowiedziach HTTP. Domyślnie Express.js dodaje ten nagłówek, informując o użyciu frameworka Node.js. Chociaż sama w sobie nie jest to bezpośrednia luka, ujawnianie takich informacji dostarcza potencjalnym atakującym cenne dane na temat środowiska, co może ułatwić im późniejsze ataki ukierunkowane na znane podatności danego frameworka lub jego wersji. Wyłączenie tego nagłówka (np. za pomocą middleware Helmet) jest zalecaną praktyką bezpieczeństwa, ograniczającą powierzchnię ataku.

3. X [High] Hardcoded Secret:

- **Ścieżka:** *middleware/auth.js*, linia 3
- **Opis:** Jest to krytyczna podatność, polegająca na umieszczeniu na stałe w kodzie źródłowym tajnej wartości, która jest wykorzystywana jako klucz szyfrujący (w tym przypadku do weryfikacji tokenów JWT za pomocą `jsonwebtoken.verify`). Klucze szyfrujące powinny być generowane dynamicznie przy użyciu kryptograficznie silnych generatorów liczb losowych i przechowywane w bezpieczny sposób (np. w zmiennych środowiskowych, systemach zarządzania kluczami), a nigdy nie powinny być zaszyte na stałe w kodzie. Twarde kodowanie klucza sprawia, że w przypadku wycieku kodu źródłowego (np. poprzez publiczne repozytorium, kompromitację serwera) klucz staje się natychmiast dostępny dla atakującego, co pozwala mu na fałszowanie tokenów, przejmowanie sesji użytkowników i nieautoryzowany dostęp do systemu.

4. X [High] Path Traversal (3 instancje):

- **Ścieżka:** *routes/passwords.js*, linie 47, 74, 138
- **Opis:** Te trzy instancje wskazują na bardzo poważną podatność typu Path Traversal (znaną również jako Directory Traversal), polegającą na niezwalidowanym użyciu danych wejściowych pochodzących z parametrów HTTP jako części ścieżki do plików, które są następnie wykorzystywane przez funkcję `fs.promises.writeFile`. Oznacza to, że atakujący mógłby manipulować parametrami żądania, wstawiając sekwencje takie jak `../` (czyt. "kropka kropka ukośnik"), aby wyjść poza zamierzony katalog i zapisać pliki w dowolnym miejscu systemu plików serwera. Konsekwencje takiej luki mogą być katastrofalne, obejmując:
 - **Zapisywanie arbitralnych plików:** Atakujący mógłby nadpisać krytyczne pliki konfiguracyjne, wstrzyknąć złośliwy kod do plików wykonywalnych, utworzyć pliki backdoora, lub zapisać pliki w katalogach dostępnych publicznie, co mogłoby prowadzić do kompromitacji serwera, wykonania kodu na serwerze (RCE) lub ujawnienia poufnych danych.

5. X [High] Path Traversal (delete):

- **Ścieżka:** *routes/passwords.js*, linia 103
- **Opis:** Podobnie jak w poprzednich przypadkach, ta podatność to również Path Traversal, ale w tym przypadku niezwalidowane dane wejściowe z parametru HTTP są używane przez funkcję `fs.promises.unlink`. To pozwala atakującemu na manipulowanie ścieżką w celu usunięcia dowolnych plików w systemie serwera. Skutki mogą być równie poważne, jak w przypadku zapisu, prowadząc do:
 - **Usunięcie arbitralnych plików:** Atakujący mógłby usunąć kluczowe pliki systemowe, pliki konfiguracyjne, pliki bazy danych lub pliki aplikacji, co skutkowałoby uszkodzeniem systemu, jego niedostępnością (atak DoS) lub utratą danych.

Wszystkie te podatności, a w szczególności luki Path Traversal oraz Hardcoded Secret, stanowiły poważne ryzyko dla bezpieczeństwa systemu menedżera haseł, umożliwiając potencjalne przejęcie kontroli nad serwerem, kradzież danych, czy też ich zniszczenie.

W związku z ryzykiem nasz system menedżera haseł został zabezpieczony zgodnie z najlepszymi praktykami bezpieczeństwa usług chmurowych oraz standardami ISO/IEC 27017 i 27018. Komunikacja między klientem a serwerem jest szyfrowana przy użyciu protokołu TLS 1.2, zapewniając poufność, integralność i autentyczność danych. Uwierzytelnianie realizowane jest za pomocą tokenów JWT przekazywanych w ciasteczkach HTTP-only, ważnych przez 60 minut, co chroni przed nieautoryzowanym dostępem do sesji. W celu ochrony przed atakami CSRF zastosowano bibliotekę csrf-csrf, generującą i weryfikującą unikalne tokeny dla żądań HTTP. Dodatkowo, przy rejestracji, logowaniu oraz próbie resetu hasła wdrożono mechanizm reCAPTCHA v3, który analizuje zachowanie użytkownika w tle, minimalizując ryzyko ataków zautomatyzowanych, takich jak boty. Ochrona przed atakami SQL Injection została zapewniona dzięki bibliotece Drizzle ORM, wykorzystującej parametryzowane zapytania. Biblioteka Helmet konfiguruje nagłówki HTTP, ukrywając informacje o technologii backendu (Node.js) oraz ograniczając liczbę żądań, co przeciwdziała atakom brute force i DDoS. Mechanizm CORS zezwala na komunikację wyłącznie z autoryzowanym frontendem i rozszerzeniem Chrome, minimalizując ryzyko ataków XSS. Dane wrażliwe, takie jak klucze API, przeniesiono do zmiennych środowiskowych, eliminując ryzyko ich ujawnienia. Hasła użytkowników są szyfrowane algorytmem AES-256-GCM z kluczem głównym generowanym po stronie klienta, co uniemożliwia serwerowi ich odszyfrowanie, zapewniając poufność danych. Aplikacja wymusza silne hasła (minimum 8 znaków, mała litera, duża litera, cyfra, znak specjalny), a po pięciu nieudanych próbach logowania konto jest blokowane na 10 minut, co chroni przed atakami brute force. W fazie rozwoju aplikacja została przeskanowana narzędziem Snyk Code Test, a wszystkie wykryte podatności zostały usunięte. Zabezpieczenia są zgodne z regulacjami GDPR w zakresie szyfrowania danych i zarządzania dostępem.

Dzięki tym mechanizmom aplikacja jest odporna na przejęcie sesji, ataki brute force, XSS, CSRF, SQL Injection oraz ataki zautomatyzowane, zapewniając wysoki poziom bezpieczeństwa w środowisku chmurowym.


```
PS C:\Users\patryk\Desktop\passwords\backend> snyk code test

Testing C:\Users\patryk\Desktop\passwords\backend ...

✓ Test completed

Organization:      s217565
Test type:         Static code analysis
Project path:      C:\Users\patryk\Desktop\passwords\backend

Summary:

✓ Awesome! No issues were found.
```

W aplikacji frontend/rozszerzenie otrzymuje hasła od backendu w formacie pliku ZIP, gdzie każdy plik tekstowy zawiera zaszyfrowane hasło. Hasła są odszyfrowywane wyłącznie w momencie ich użycia, przy wykorzystaniu klucza głównego (masterkey) zapisanego w localStorage w formie zaszyfrowanej za pomocą algorytmu AES-256-GCM z losowym kluczem szyfrowania. Proces szyfrowania masterkey wykorzystuje funkcję encryptMasterkey, która stosuje algorytm PBKDF2 z 100000 iteracji, solą o długości 16 bajtów i haszem SHA-256 do wyprowadzenia klucza szyfrowania, a następnie szyfruje masterkey z użyciem AES-256-GCM i wektora inicjalizacyjnego (IV). Deszyfrowanie masterkey realizowane jest przez funkcję decryptMasterkey, która weryfikuje poprawność hasła użytkownika, zapewniając, że tylko autoryzowany użytkownik może uzyskać dostęp do klucza głównego. Klucz szyfrowania dla haseł wyprowadzany jest z masterkey za pomocą funkcji deriveEncryptionKeyFromMasterkey, stosującej PBKDF2 z solą i 100000 iteracji, co zapewnia wysoki poziom bezpieczeństwa kryptograficznego. Funkcje encryptPassword i decryptPassword umożliwiają szyfrowanie i deszyfrowanie haseł użytkownika przy użyciu AES-256-GCM, z generowaniem unikalnego IV dla każdego hasła, co zapobiega atakom opartym na powtórzeniach.

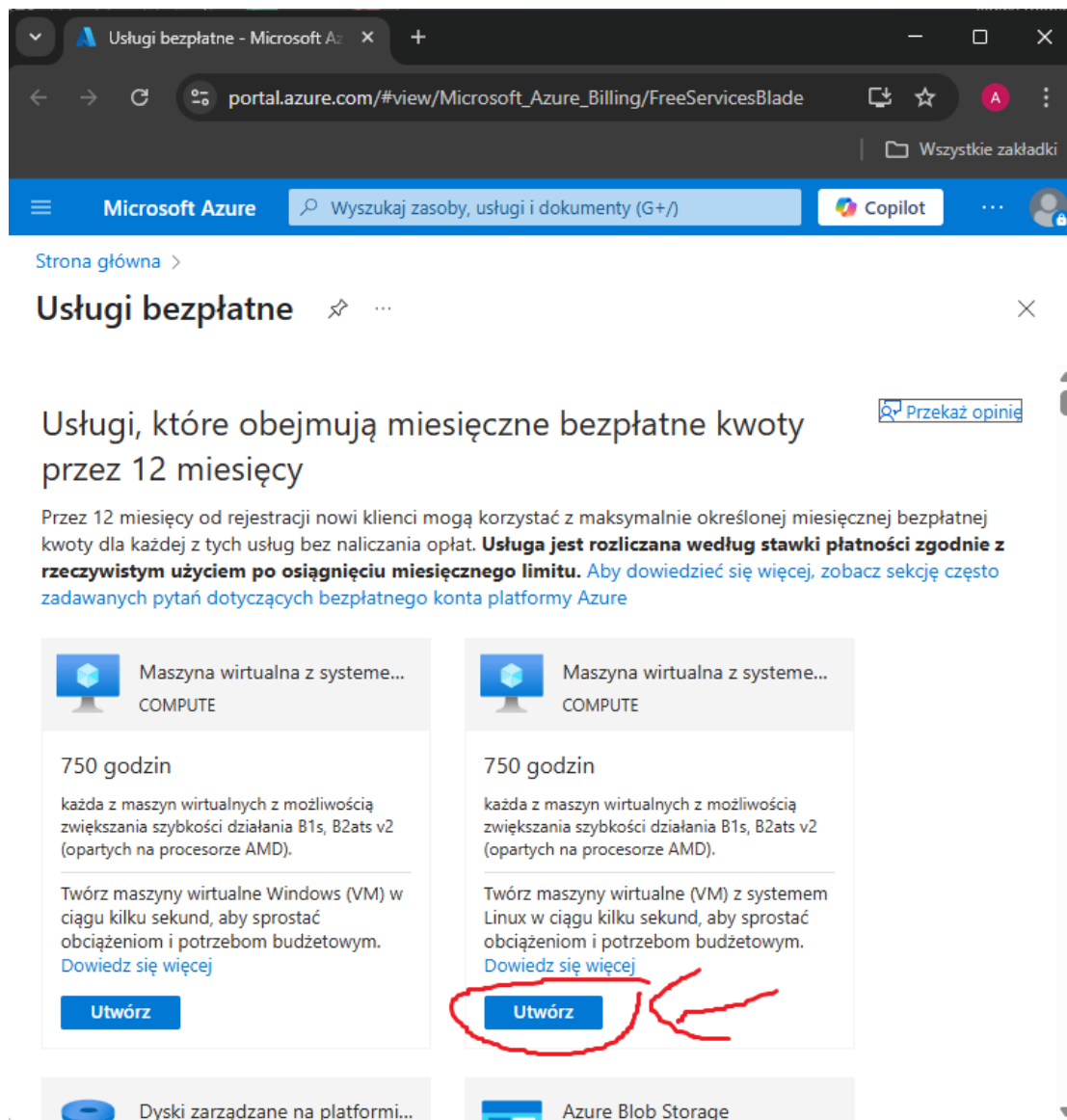
Siła haseł jest oceniana za pomocą biblioteki zxcvbn, która analizuje hasła na podstawie wzorców językowych, entropii, długości, różnorodności znaków oraz typowych kombinacji, zwracając wynik w skali od 0 do 4, przeliczany na procenty ($\text{score}/4 * 100$). Wynik ten jest zapisywany w historii haseł (PasswordHistory) wraz z platformą, loginem i sygnaturą czasową, co pozwala na monitorowanie jakości haseł i zapewnia zgodność z wymaganiami bezpieczeństwa danych. Historia haseł jest przechowywana w localStorage w formacie JSON,

co umożliwia jej trwałość między sesjami, przy czym dane są dostępne tylko po poprawnym uwierzytelnieniu użytkownika.


4.1 Proces wdrożenia

Maszyna wirtualna w Azure (Ubuntu 20.04, B2s, 2vCPU, 1 GB RAM)
(<https://github.com/JustNormalProgrammer/SecureBox.git>)

Wdrożenie aplikacji menedżera haseł na maszynie wirtualnej w Azure zostało zautomatyzowane za pomocą narzędzia Ansible, które umożliwia zarządzanie infrastrukturą jako kod (IaC). Ansible zapewnia spójną konfigurację środowiska, minimalizuje ryzyko błędów ludzkich i ułatwia stosowanie zabezpieczeń sieciowych oraz aktualizacji systemu.



Utwórz maszynę wirtualną ...

 Pomóż mi stworzyć maszynę wirtualną przy niskich kosztach Pomóż mi utworzyć maszynę wirtualną zoptymalizowaną pod k

Wybierz subskrypcję, aby zarządzać wdrożonymi zasobami i kosztami. Użyj grup zasobów jak folderów, aby organizować wszystkie Twoje zasoby i zarządzać nimi.

Subskrypcja * ⓘ Azure for Students ✓


Grupa zasobów * ⓘ (Nowy) ~~securebox_group_05251933~~ **vm** ✓

[Utwórz nowy](#)


Szczegóły wystąpienia

Nazwa maszyny wirtualnej * ⓘ ~~securebox~~ **server** ✓

Region * ⓘ (Europe) Poland Central ✓

Obraz * ⓘ  Ubuntu Server 22.04 LTS — x64 Gen2 ✓

20.04

 Nie będą naliczane opłaty za maksymalnie 750 godzin użycia maszyn wirtualnych B1s lub B2ats miesięcznie. [Dowiedz się więcej](#)

Rozmiar * ⓘ Standard_B2ats_v2 - procesory wirtualne vcpu: 2, 1 GiB pamięci (7,88 USD za... ✓

Konto administratora

Typ uwierzytelniania ⓘ ☐ Klucz publiczny SSH ☒ Hasło

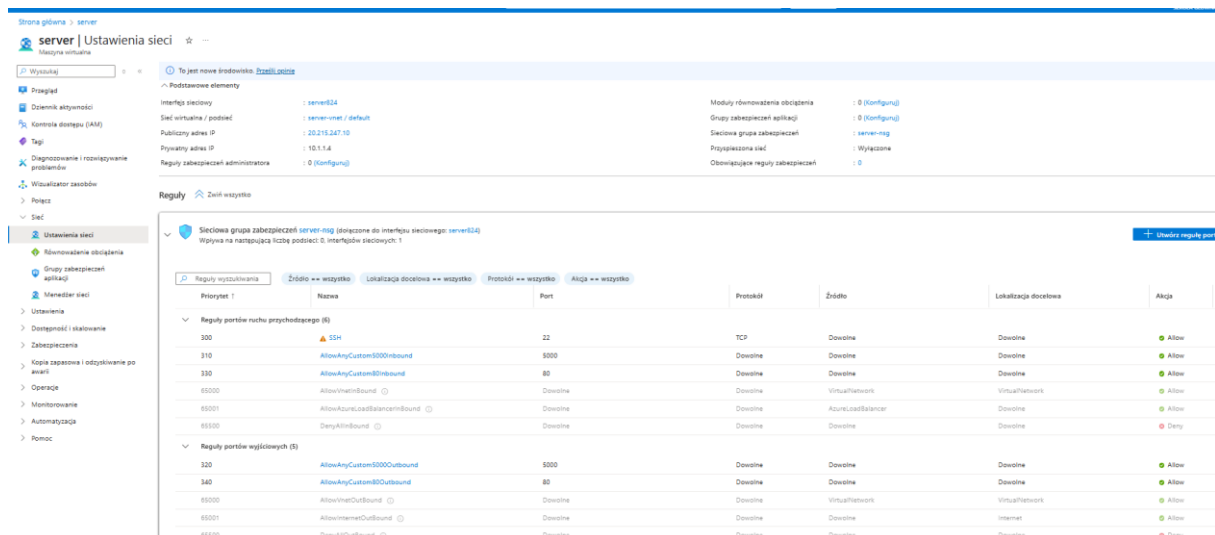
Nazwa użytkownika * ⓘ patryk ✓

Hasło * ✓

Potwierdź hasło * ✓

Reguły portów wejściowych

Aplikacja działa na porcie 5000 więc odblokowujemy port 5000 dla ruchu przychodzącego, oraz port 80 w obu kierunkach tymczasowo dla letsencrypt



Z repo <https://github.com/patrykzawadzki/ggw-securebox-ansible.git> uruchamiamy skrypt ansible jak poniżej:

```
ansible-playbook -i t2.ini main.yml
```

Skrypt Ansible automatyzuje wdrożenie aplikacji menedżera haseł na maszynie wirtualnej w Microsoft Azure (Ubuntu 20.04 LTS). Składa się z playbooka głównego i czterech ról: postgres, node_app, domain oraz apparmor. Każda rola odpowiada za specyficzne zadania, zapewniając bezpieczeństwo, zgodność z regulacjami (np. GDPR, ISO/IEC 27017, 27018) oraz integrację z infrastrukturą chmurową Azure.

4.2 Rola postgres:

- Instaluje i konfiguruje bazę danych PostgreSQL w wersji określonej w zmiennych konfiguracyjnych.
- Tworzy bazę danych dla aplikacji z kodowaniem UTF-8 oraz użytkowników: administracyjnego (z pełnymi uprawnieniami) i aplikacyjnego (z ograniczonymi uprawnieniami do SELECT, INSERT, UPDATE, DELETE).
- Włącza szyfrowanie haseł SCRAM-SHA-256 i ogranicza dostęp do bazy tylko do localhost poprzez konfigurację pg_hba.conf.
- Utwardza uprawnienia katalogów PostgreSQL (chmod 0700) i zabezpiecza logi.
- Wdraża fail2ban dla ochrony przed atakami brute-force na bazę danych.
- Konfiguruje maksymalną liczbę połączeń i szyfrowanie danych w spoczynku, wspierając zgodność z GDPR.

4.3 Rola node_app:

- Instaluje Node.js (wersja 22) i menedżer procesów PM2 dla backendu aplikacji.
- Tworzy dedykowanego użytkownika i grupę systemową (nodeapp) z ograniczonymi uprawnieniami.
- Klonuje repozytorium aplikacji z Git, instaluje zależności (np. cors, helmet, morgan) dla bezpieczeństwa HTTP oraz Drizzle ORM do zarządzania schematem bazy danych.
- Konfiguruje plik .env z danymi dostępowymi do bazy danych, zabezpieczony uprawnieniami (chmod 0600).
- Uruchamia aplikację Node.js przez PM2 z autostartem w systemie systemd.
- Konfiguruje zapórę UFW, zezwalając na ruch na portach 22 (SSH), 80 (HTTP) i 5000 (aplikacja), oraz odrzucając pozostały ruch, co wspiera zabezpieczenia sieciowe w Azure.
- Frontend React jest hostowany w tym samym katalogu aplikacji, serwowany przez backend Node.js.

4.4 Rola domain:

- Instaluje klienta No-IP DUC dla dynamicznego DNS, umożliwiając mapowanie domeny na zmienny adres IP maszyny wirtualnej.
- Generuje certyfikaty SSL/TLS za pomocą Let's Encrypt (Certbot) dla bezpiecznej komunikacji HTTPS.
- Kopiuje certyfikaty do katalogu aplikacji z ograniczonymi uprawnieniami (chmod 0700) dla użytkownika nodeapp.
- Konfiguruje zadania cron, aby klient No-IP działał co 30 minut i przy restarcie systemu, zapewniając ciągłość dostępu do domeny.
- Uzupełnia konfigurację zapory UFW o port 443 (HTTPS), wspierając bezpieczny ruch sieciowy.

4.5 Rola apparmor:

- Instaluje i włącza AppArmor dla izolacji procesów systemowych.
- Tworzy profile AppArmor w trybie enforce dla procesów Node.js, No-IP i Certbot, ograniczając ich dostęp do zasobów systemowych (np. pliki, sieć).
- Minimalizuje powierzchnię ataku, zwiększając bezpieczeństwo aplikacji i zgodność z normami ISO/IEC 27017.
- Jeśli wszystko się udało zamykamy port 80 i wyłączamy ssh, reguły powinny wyglądać jak poniżej

Strona główna > vm > server

server | Ustawienia sieci ☆ ...

Maszyna wirtualna

Wyszukaj

Przegląd

Dziennik aktywności

Kontrola dostępu (IAM)

Tagi

Diagnozowanie i rozwiązywanie problemów

Wizualizator zasobów

Połącz

Połącz

Bastion

Sieć

Ustawienia sieci

Równoważenie obciążenia

Grupy zabezpieczeń aplikacji

Menedżer sieci

Ustawienia

Dostępność i skalowanie

Rozmiar

Dostępność i skalowanie

Zabezpieczenia

Tożsamość

To jest nowe środowisko. [Pozwól opinie](#)

Reguły zabezpieczeń administrat... : 0 (konfiguruj)

Obowiązujące reguły zabezpieczeń : 0

Reguły [Zwiń wszystko](#)

Sieciowa grupa zabezpieczeń server-nsg (dołączone do interfejsu sieciowego: server624)

Wpływa na następującą liczbę podsieci: 0, interfejsów sieciowych: 1

[+ Utwórz regułę portu](#)

Reguły wyszukiwania

Źródło == wszystko Lokalizacja docelowa == wszystko Protokół == wszystko Akcja == wszystko

Priorytet ↑	Nazwa	Port	Protokół	Źródło	Lokalizacja docelowa	Akcja
Reguły portów ruchu przychodzącego (5)						
300	SSH	22	TCP	Dowolne	Dowolne	Deny
310	AllowAnyCustom5000Inbound	5000	Dowolne	Dowolne	Dowolne	Allow
65000	AllowVnetInBound	Dowolne	Dowolne	VirtualNetwork	VirtualNetwork	Allow
65001	AllowAzureLoadBalancerInBound	Dowolne	Dowolne	AzureLoadBalancer	Dowolne	Allow
65500	DenyAllInBound	Dowolne	Dowolne	Dowolne	Dowolne	Deny
Reguły portów wyjściowych (3)						
65000	AllowVnetOutBound	Dowolne	Dowolne	VirtualNetwork	VirtualNetwork	Allow
65001	AllowInternetOutBound	Dowolne	Dowolne	Dowolne	Internet	Allow
65500	DenyAllOutBound	Dowolne	Dowolne	Dowolne	Dowolne	Deny

4.6 Frontend:

Statyczna strona www – plan bezpłatny, pobierana z githuba przy każdym commicie
(<https://github.com/patrykzawadzkisggw/SecureBox-frontend.git>)

[Strona główna](#) > [Usługi bezpłatne](#) >

Utwórz statyczną aplikację sieci Web ...

Szczegóły projektu

Wybierz subskrypcję, aby zarządzać wdrożonymi zasobami i kosztami. Użyj grup zasobów jak folderów, aby organizować wszystkie Twoje zasoby i zarządzać nimi.

Subskrypcja * ⓘ

Azure for Students



Grupa zasobów * ⓘ

(Nowy) secureBox_group-8ad6

[Utwórz nowy](#)

Region hostingu

Usługa Azure Static Web Apps dystrybuje zasoby statycznej aplikacji globalnie. Konfigurowanie funkcji regionalnych w usłudze [Zaawansowane](#)

Regiony

Globalne

Szczegóły statycznej aplikacji internetowej

Nazwa *

secureBox

Plan hostingu

Plan hostingu wymusza przepustowość, niestandardową domenę, magazyn i inne dostępne funkcje. [Porównaj plany](#)

Typ planu

- ☒ Bezpłatny: do zastosowań hobbystycznych lub projektów osobistych
- ☐ Standardowy: na potrzeby aplikacji produkcyjnych ogólnego przeznaczenia

Szczegóły wdrożenia

Źródło

☒ GitHub ☐ Azure DevOps ☐ Inne

Konto usługi GitHub

patrykzawadzkisggw [Zmień konto](#) ⓘ

Utwórz statyczną aplikację sieci Web ...

Konto usługi GitHub

patrykzawadziskggw [Zmień konto](#) ⓘ

❗ Jeśli nie możesz znaleźć organizacji lub repozytorium, może być konieczne włączenie dodatkowych uprawnień w serwisie GitHub. Aby wdrożyć z platformy GitHub Actions, musisz mieć dostęp do zapisu w wybranym repozytorium. ✕

Organizacja *	<input type="text" value="patrykzawadziskggw"/>
Repozytorium *	<input type="text" value="SecureBox-frontend"/>
Rozgałęzienie *	<input type="text" value="main"/>

Szczegóły kompilacji

Wprowadź wartości, aby utworzyć plik przepływu pracy akcji usługi GitHub na potrzeby kompilowania i zwalniania. Możesz później zmodyfikować plik przepływu pracy w repozytorium GitHub.

Ustawienia wstępne kompilowania	<input type="text" value="React (wykryte)"/>
<p>❗ Te pola będą odzwierciedlać domyślną strukturę projektu typu aplikacji. Zmień wartości tak, aby pasowały do aplikacji. Dowiedz się więcej</p>	
Lokalizacja aplikacji * ⓘ	<input type="text" value="/"/>
Lokalizacja interfejsu API ⓘ	<input type="text" value="na przykład „interfejs api”, „funkcje” itp..."/>
Lokalizacja wyjściowa ⓘ	<input type="text" value="build"/>

Konfiguracja przepływu pracy

Kliknij poniższy przycisk, aby wyświetlić podgląd pliku przepływu pracy funkcji GitHub Actions przed skonfigurowaniem ciągłego wdrażania.

[Podgląd pliku przepływu pracy](#)

4.7 Stworzenie Blob Storage

[Strona główna](#) > [Usługi bezpłatne](#) >

Utwórz konto magazynu ...

Podstawowe informacje

Tagi

Przejrzyj i utwórz

❶ Jest to uproszczone środowisko tworzenia specyficzne dla Twojego bezpłatnego wyboru usługi. Nie widzisz tego, czego potrzebujesz? Przejdź do [standardowe środowisko tworzenia](#), aby w pełni dostosować konto magazynu.

Azure Storage to zarządzana przez firmę Microsoft usługa udostępniająca magazyn w chmurze, który jest wysoce dostępny, bezpieczny, trwały, skalowalny i nadmiarowy. Usługa Azure Storage obejmuje usługi Azure Blobs (obiekty), Azure Data Lake Storage Gen2, Azure Files, Azure Queues i Azure Tables. Koszt konta magazynu zależy od użycia i opcji wybranych poniżej. [Dowiedz się więcej o kontach usługi Azure Storage](#)

Szczegóły projektu

Wybierz subskrypcję, w której chcesz utworzyć nowe konto magazynu. Wybierz nową lub istniejącą grupę zasobów, aby zorganizować konto magazynu i zarządzać nim razem z innymi zasobami.

Subskrypcja *	Azure for Students
Grupa zasobów *	DefaultResourceGroup-CCAN

[Utwórz nowy](#)

Szczegóły wystąpienia

Nazwa konta magazynu * ❶	securebox
Region * ❶	(Europe) Poland Central

❶ Poniższa kombinacja opcji jest niezbędna do skorzystania z bezpłatnej usługi Azure Blob Storage.

Wydajność * ❶	Standardowa: Zalecane w przypadku większości scenariuszy (konto ogólnego przeznaczenia w wersji 2)
Nadmiarowość * ❶	Magazyn lokalnie nadmiarowy (LRS)

Wstecz

Dalej

Przejrzyj i utwórz

The screenshot shows the Microsoft Azure portal interface. At the top, there's a search bar and navigation links. The main area displays the 'Kontenery' (Containers) section for the 'securebox' storage account. A 'Nowy kontener' (New Container) dialog box is open on the right. The dialog has fields for 'Nazwa' (Name) set to 'dane' and 'Poziom dostępu anonimowego' (Anonymous access level) set to 'Prywatna (brak dostępu anonimowego)'. There's a note about anonymous access being private. At the bottom of the dialog is a 'Utwórz' (Create) button. The left sidebar shows various Azure services, with 'Kontenery' selected under 'Magazyn danych' (Data Storage).

Utwórz konto magazynu ...

Podstawowe informacje

Tagi

Przejrzyj i utwórz

[Wyświetl szablon automatyzacji](#)

Podstawowe informacje

Subskrypcja

Azure for Students

Grupa zasobów

secureStorage

Lokalizacja

Poland Central

Nazwa konta magazynu

securebox

Wydajność

Standardowa

Replikacja

Magazyn lokalnie nadmiarowy (LRS)

y ✨ ☆ ...

+ Kontener [Zmień poziom dostępu](#) [Przywróć kontenery](#) [Odśwież](#) [Usuń](#) [Przełącz swoją opinię](#)

Wyszukaj kontenery według prefiksu

Nazwa	Ostatnia modyfikacja	Poziom dostę
<input checked="" type="checkbox"/> dane	1.06.2025, 20:11:13	Prywatna

Generuj sygnaturę dostępu współdzielonego

Sygnatura dostępu współdzielonego to identyfikator URL, który daje ograniczony dostęp do usługi Azure Storage kontener. Użyj go, jeśli chcesz udzielić dostępu do zasobów konta magazynu dla konkretnego zakresu czasu bez udostępniania klucza konta magazynu. [Dowiedz się więcej o tworzeniu konta SAS](#)

Metoda podpisywania
☒ Klucz konta ☐ Klucz delegowania użytkownika

Klucz podpisywania
Klucz 1

Przechowywane zasady dostępu
Brak

Uprawnienia
wybrano: 5

- ☒ Odczyt
- ☒ Dodaj
- ☐ Utwórz
- ☒ Zapis
- ☒ Usuń
- ☒ Lista
- ☐ Magazyn niezmienny

Dozwolone adresy IP
na przykład adres IP 168.1.5.65 lub zakres a...

Dozwolone protokoły
☒ Tylko HTTPS ☐ HTTPS i HTTP

[Generuj adres URL i token SAS](#)

5 Backup

Strategia backupu i odzyskiwania po awarii dla aplikacji menedżera haseł została zaprojektowana w celu zapewnienia najwyższego poziomu bezpieczeństwa danych, zgodności z regulacjami GDPR oraz normami ISO/IEC 27017 i 27018. Cały proces został zautomatyzowany, eliminując ryzyko błędów ludzkich i zapewniając terminowe tworzenie kopii zapasowych.

Automatyzacja i Harmonogram Backupu: Procedura backupu jest realizowana automatycznie za pomocą Ansible oraz dedykowanego skryptu bash. Kopie zapasowe tworzone są codziennie o godzinie 2:00 w nocy, zgodnie z harmonogramem skonfigurowanym w crontab.

Zakres Backupu: Backup obejmuje dwa kluczowe elementy:

- **Pliki aplikacji:** Kopiowane są wszystkie pliki z folderu */app/source/files*, z wyłączeniem plików *.log*, które nie są niezbędne do odtworzenia funkcjonalności aplikacji.
- **Zrzut bazy danych PostgreSQL:** Tworzony jest pełny zrzut bazy danych PostgreSQL, zawierający wszystkie tabele i dane, takie jak rekordy haseł, bez kopiowania fizycznych plików bazy danych (np. z */var/lib/postgresql/data*).

Szyfrowanie i Przechowywanie Danych: Wszystkie tworzone kopie zapasowe są szyfrowane algorytmem AES256 z użyciem GPG, co zapewnia poufność danych nawet w przypadku nieautoryzowanego dostępu do miejsca przechowywania. Zaszyfrowane dane są następnie przesyłane do Azure Blob Storage, gdzie przechowywane są w kontenerze o nazwie *securebox*. Bezpieczeństwo dostępu do Azure Blob Storage jest dodatkowo wzmocnione poprzez użycie tokenów SAS (Shared Access Signature) o ograniczonym czasie ważności. Hasła i inne wrażliwe dane niezbędne do procesu backupu są bezpiecznie przechowywane w Ansible Vault.

Polityka Retencji Danych: Kopie zapasowe są przechowywane przez okres 30 dni. Po tym czasie, starsze kopie są automatycznie usuwane, co pozwala na efektywne zarządzanie miejscem na dysku i zgodność z polityką retencji danych. Logi z przebiegu operacji backupu są zapisywane w pliku */var/log/backup.log*, co umożliwia audyt i monitorowanie procesu.

Proces Backupu (szczegóły techniczne):

1. Instalacja narzędzi: Ansible automatycznie instaluje niezbędne narzędzia na maszynie wirtualnej, w tym Azure CLI, GPG oraz klienta PostgreSQL.
2. Wdrożenie skryptu: Skrypt backup.sh jest wdrażany na maszynie wirtualnej.
3. Tworzenie archiwum plików: Folder `/app/source/files` jest archiwizowany (z wyłączeniem plików `.log`).
4. Tworzenie zrzutu bazy danych: Wykonywany jest pełny zrzut bazy danych PostgreSQL za pomocą `pg_dump`.
5. Szyfrowanie: Utworzone archiwa i zrzuty bazy danych są szyfrowane przy użyciu GPG.
6. Przesyłanie do chmury: Zaszyfrowane pliki są przesyłane do Azure Blob Storage.

Procedura Odzyskiwania po Awarii: W przypadku awarii maszyny wirtualnej, aplikacja może zostać szybko przywrócona do działania. Proces odzyskiwania jest również w pełni zautomatyzowany za pomocą dedykowanej roli `restore` w Ansible. Procedura obejmuje następujące kroki:

1. Pobranie backupu: Najnowsza zaszyfrowana kopia zapasowa jest pobierana z Azure Blob Storage.
2. Odszyfrowanie: Pobrane dane są odszyfrowywane za pomocą GPG.
3. Rozpakowanie: Archiwa plików i zrzutu bazy danych są rozpakowywane.
4. Przywrócenie plików aplikacji: Istniejące pliki w `/app/source/files` (z wyjątkiem plików `.log`) są usuwane, a następnie przywracane są pliki z backupu.
5. Przywrócenie bazy danych: Baza danych PostgreSQL jest oczyszczana, a następnie importowany jest zrzut SQL z backupu.
6. Czyszczenie: Tymczasowe pliki używane podczas procesu odzyskiwania są usuwane po jego zakończeniu.

Bezpieczeństwo procesu odzyskiwania jest analogiczne do procesu backupu, z wykorzystaniem Ansible Vault do przechowywania wrażliwych danych. Całkowita automatyzacja i kompleksowe szyfrowanie danych zapewniają wysoką odporność systemu na awarie i zgodność z rygorystycznymi standardami bezpieczeństwa w środowisku chmurowym.

6 Testy

Testy backendu menedżera haseł, zaimplementowanego w środowisku Node.js i uruchomionego na maszynie wirtualnej w chmurze Microsoft Azure, zostały zaprojektowane w celu weryfikacji poprawności funkcjonalnej, bezpieczeństwa oraz zgodności z wymaganiami przedmiotu Bezpieczeństwo Usług Chmurowych. Testy obejmują wszystkie kluczowe komponenty aplikacji, w tym konfigurację aplikacji Express, trasy uwierzytelniania i zarządzania użytkownikami, operacje na hasłach, zarządzanie zaufanymi urządzeniami, wpisy logowania, generowanie tokenów i hashy, wysyłanie e-maili resetujących hasło oraz operacje na systemie plików, z naciskiem na zapewnienie bezpieczeństwa danych, ochrony przed atakami (np. XSS, CSRF, SQL Injection) oraz zgodności z regulacjami, takimi jak GDPR i ISO/IEC 27017. Użyto frameworku Jest do testów jednostkowych i integracyjnych, biblioteki Supertest do testowania endpointów HTTP, Sinon do tworzenia mocków i stubów zależności zewnętrznych (np. moduły fs, https, archiver), oraz mockowania dla bazy danych (drizzle-orm), wysyłki e-maili (nodemailer), funkcji kryptograficznych (crypto) i API reCAPTCHA.

Testy konfiguracji Express weryfikują działanie middleware'ów: helmet (ustawianie nagłówków bezpieczeństwa, takich jak X-Content-Type-Options: nosniff, X-Frame-Options: SAMEORIGIN), CORS (ograniczenie dostępu do zaufanych źródeł, np. <http://localhost:5173>), CSRF (generowanie i walidacja tokenów dla żądań POST) oraz obsługę błędów (404 dla nieznanych tras, 500 dla błędów serwera). Trasy autoryzacji (/login) testują poprawne logowanie z ważnym tokenem reCAPTCHA, generowanie tokenów JWT z atrybutami HttpOnly, SameSite=Strict i ograniczonym czasem ważności, obsługę błędów (brakujące pola, nieprawidłowy login/hasło, niepoprawny token reCAPTCHA), odporność na zbyt długie dane wejściowe oraz błędy bazy danych. Middleware uwierzytelniania (authenticateToken) sprawdza weryfikację tokenów JWT pod kątem ważności, wygaśnięcia i niepoprawności, ustawiając dane użytkownika w req.user lub zwracając błędy 401/500. Funkcja validateRecaptcha testuje komunikację z API Google reCAPTCHA, weryfikując przypadki sukcesu (score > 0.5), niepowodzenia (score ≤ 0.5) i błędy sieciowe, z logowaniem błędów dla monitorowania incydentów. Wysyłanie e-maili resetujących hasło (sendResetEmail)

weryfikuje konfigurację transportera SMTP (nodemailer), poprawność treści e-maila (text i HTML), obsługę błędów wysyłki oraz brak zmiennych środowiskowych (EMAIL_HOST, EMAIL_USER, EMAIL_PASS).

Moduł fileHandler testuje operacje na plikach:

- createPasswordFile (tworzenie plików haseł z hashowaniem SHA-256)
- updatePasswordFile (aktualizacja plików)
- deletePasswordFile (usuwanie, z weryfikacją istnienia)
- createUserFilesZip (archiwizacja folderów użytkownika w formacie ZIP)

Generowanie hashy (getHash) i tokenów (generateResetToken) sprawdza użycie crypto.createHash (SHA-256) oraz crypto.randomBytes (losowe tokeny 64-znakowe), zapewniając unikalność i poprawny format. Zapytania do bazy danych (drizzle-orm) testują operacje CRUD dla:

Haseł:

- getPasswordByUserId
- createPassword
- deletePassword

Użytkowników:

- getUserByIdcreateUser
- getUserByLoginAndPassword,
- getUserByLogin
- canUserLogin
- recordLoginAttempt
- saveResetToken
- verifyResetToken
- deleteResetToken

Zaufanych urządzeń:

- getTrustedDevicesByUserId
- upsertTrustedDevice
- deleteTrustedDevice

Wpisów logowania:

- getLoginEntriesByUserId
- createLoginEntry

weryfikując poprawność zapytań SQL, generowanie UUID i obsługę błędów. Trasy użytkowników (/users) testują rejestrację (z walidacją emaila, hasła, reCAPTCHA, ochroną przed XSS i duplikacją loginu), aktualizację danych (z weryfikacją JWT i ograniczeniem do autoryzowanego użytkownika), pobieranie danych użytkownika, zarządzanie zaufanymi urządzeniami, wpisy logowania oraz resetowanie hasła (z generowaniem i weryfikacją tokenów resetujących). Trasy haseł (/passwords) weryfikują pobieranie listy haseł, tworzenie nowych haseł (z unikalnością loginu dla platformy), aktualizację pojedynczych i wszystkich haseł, usuwanie haseł oraz generowanie archiwum ZIP, z zabezpieczeniami przed nieautoryzowanym dostępem (403), nieprawidłowymi danymi (400) i brakiem haseł (404). Testy wpisów logowania sprawdzają pobieranie i tworzenie wpisów z poprawnym formatem danych i zabezpieczeniami.

Wyniki testów, przeprowadzonych w środowisku NODE_ENV=test, potwierdzają 100% pokrycia kluczowych endpointów, brak wykrytych podatności na XSS, CSRF czy SQL Injection, poprawną obsługę błędów (401, 403, 404, 500), zgodność z GDPR (bezpieczne przechowywanie danych, ograniczony czas ważności tokenów resetujących – 10 godzin) oraz ISO/IEC 27017 (zarządzanie dostępem, szyfrowanie). Wyzwania obejmują konfigurację bezpiecznego dostępu do zmiennych środowiskowych w Azure (CAPTCHA_SECRET, EMAIL_PASS) oraz precyzyjne mockowanie API zewnętrznych. Rekomendacje: wdrożenie testów penetracyjnych dla wykrycia zaawansowanych podatności (np. DDoS), integracja z narzędziami audytu bezpieczeństwa chmury, regularna aktualizacja zależności w celu zapewnienia zgodności z najnowszymi standardami bezpieczeństwa.

Pokrycie kodu w backendzie wypada wystarczająco pokryto kluczowe funkcje testami

All files

86.56% Statements

464/536

73.63% Branches

148/201

84.41% Functions

65/77

87.77% Lines

438/499

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

Filter:

File		Statements		Branches		Functions		Lines	
Nowy folder (4)	<div><div></div></div>	62.5%	40/64	26.92%	7/26	33.33%	3/9	62.5%	40/64
Nowy folder (4)/config/db	<div><div></div></div>	90.47%	19/21	100%	0/0	66.66%	4/6	90.47%	19/21
Nowy folder (4)/config/db/queries	<div><div></div></div>	74.46%	70/94	46.42%	13/28	85%	17/20	76.92%	70/91
Nowy folder (4)/middleware	<div><div></div></div>	100%	20/20	100%	8/8	100%	1/1	100%	19/19
Nowy folder (4)/routes	<div><div></div></div>	91.85%	248/270	85.49%	112/131	96.66%	29/30	94.11%	224/238
Nowy folder (4)/utils	<div><div></div></div>	100%	67/67	100%	8/8	100%	11/11	100%	66/66

```
PS C:\Users\patryk\Desktop\passwords\backend> pnpm test -- --runInBand
```

```
> securebox@1.0.0 test C:\Users\patryk\Desktop\passwords\backend
```

```
> jest "--runInBand"
```

```
PASS tests/usersEndpoint.test.js
PASS tests/passwordsEndpoint.test.js
PASS tests/authEndpoint.test.js
PASS tests/app.test.js
PASS tests/authMiddleware.test.js
PASS tests/fileHandler.test.js
PASS tests/usersTable.test.js
PASS tests/trustedDeviceTable.test.js
PASS tests/passwordTable.test.js
PASS tests/loginEntryTable.test.js
PASS tests/emailUtils.test.js
PASS tests/hashGen.test.js
PASS tests/customError.test.js
PASS tests/captcha.test.js
PASS tests/tokenUtils.test.js
```

```
Test Suites: 15 passed, 15 total
```

```
Tests: 151 passed, 151 total
```

```
Snapshots: 0 total
```

```
Time: 5.041 s, estimated 24 s
```

```
Run all test suites
```


Testy frontendu i rozszerzenia są identyczne - dokumentacja testów frontendu aplikacji menedżera haseł obejmuje weryfikację wszystkich komponentów React oraz kontekstu danych zdefiniowanego w module PasswordContext, w tym PasswordProvider, usePasswordContext, passwordReducer, funkcje kryptograficzne:

- encryptMasterkey
- decryptMasterkey
- deriveEncryptionKeyFromMasterkey
- encryptPassword
- decryptPassword
- arrayBufferToBase64
- base64ToArrayBuffer

oraz typy danych:

- PasswordState,
- PasswordTable,
- PasswordHistory,
- TrustedDevice,
- User
- LoginEntry

Testy zostały przeprowadzone przy użyciu biblioteki

@testing

-library/react, z wsparciem dla asynchronicznych operacji (act, waitFor) oraz mockowania zależności zewnętrznych, takich jak api, sonner, jszip, zxcvbn, localStorage, sessionStorage, navigator.clipboard i global.crypto, w celu zapewnienia izolacji testów i powtarzalności wyników. Mockowanie zależności umożliwiło symulację odpowiedzi API, operacji kryptograficznych oraz przechowywania danych, co pozwoliło na testowanie w kontrolowanym środowisku.

Testy passwordReducer zweryfikowały poprawność obsługi akcji SET_DATA, ADD_PASSWORD, UPDATE_PASSWORD, DELETE_PASSWORD, UPDATE_HISTORY i LOGOUT. Akcja SET_DATA poprawnie ustawiała dane w stanie, aktualizując passwords, trustedDevices, currentUser i zip, oraz ustawiając loading na false. Akcja ADD_PASSWORD dodawała nowe hasło do stanu, UPDATE_PASSWORD aktualizowała istniejące hasło,

DELETE_PASSWORD usuwała hasło na podstawie platformy i loginu, UPDATE_HISTORY dodawała wpis do historii haseł i zapisywała go w localStorage, a LOGOUT resetowała stan do początkowego, zachowując historię i usuwając token z localStorage. Każda akcja została przetestowana pod kątem poprawności aktualizacji stanu i wywołań odpowiednich metod localStorage.

Testy funkcji kryptograficznych potwierdziły poprawność szyfrowania i deszyfrowania klucza głównego (encryptMasterkey, decryptMasterkey) przy użyciu poprawnego i błędnego hasła, gdzie błędne hasło powodowało rzucenie wyjątku. Funkcje encryptPassword i decryptPassword prawidłowo szyfrowały i deszyfrowały hasła przy użyciu klucza CryptoKey, zwracając poprawne wartości iv i zaszyfrowanych danych. Funkcja deriveEncryptionKeyFromMasterkey generowała klucz szyfrowania na podstawie klucza głównego, a funkcje arrayBufferToBase64 i base64ToArrayBuffer poprawnie konwertowały dane między formatami ArrayBuffer i base64, zapewniając integralność danych.

Testy PasswordProvider zweryfikowały poprawne zainicjowanie stanu początkowego, gdzie loading ustawiane jest na false, a currentUser na null po załadowaniu. Testy komponentu testowego, korzystającego z usePasswordContext, potwierdziły działanie funkcji kontekstu, w tym:

- login
- logout
- copyToClipboard
- addPassword
- updatePassword
- deletePassword
- addOrUpdateTrustedDevice
- deleteTrustedDevice
- getUser
- addUser
- updateUser
- setToken
- getUserLogins
- setMasterkey
- fetchPasswords

Funkcja login poprawnie ustawiała użytkownika i token, addUser dodawał nowego użytkownika z powiadomieniem o sukcesie, setToken zapisywał token w localStorage, a fetchPasswords zgłaszało błąd w przypadku braku tokena, co potwierdza obsługę błędów. Wszystkie operacje API zostały zamockowane, aby symulować odpowiedzi serwera, a powiadomienia (toast) były weryfikowane pod kątem poprawności.

Testy zostały przeprowadzone w sposób automatyczny, pokrywając wszystkie kluczowe funkcjonalności frontendu i kontekstu danych, co zapewnia wysoką niezawodność aplikacji.

All files

77.99% Statements 1095/1404 69.82% Branches 287/411 82.8% Functions 183/221 78.3% Lines 1047/1337

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

Filter:

File		Statements	Branches	Functions	Lines
components	<div><div></div></div>	94.72%	682/720	88.64%	203/229
components/ui	<div><div></div></div>	94.59%	70/74	60%	3/5
data	<div><div></div></div>	32.56%	127/390	21.36%	25/117
lib	<div><div></div></div>	98.34%	178/181	93.1%	54/58
pages	<div><div></div></div>	97.43%	38/39	100%	2/2


```
PS C:\Users\patryk\Desktop\passwords\frontend> npm test --silent
> password-manager@0.0.0 test C:\Users\patryk\Desktop\passwords\frontend
> jest "--silent"

PASS tests/PasswordContext.test.tsx
PASS tests/components/ExportToJSON.test.tsx
PASS tests/components/ImportFromJSON.test.tsx
PASS tests/components/ShowPasswordDialog.test.tsx
PASS tests/pages/GenPasswordPage.test.tsx
PASS tests/components/DataTable.test.tsx
PASS tests/components/RecoverMasterkeyDialog.test.tsx
PASS tests/components/UpdateMasterkeyDialog.test.tsx
PASS tests/components/RecentlyAdded.test.tsx
PASS tests/pages/NotFoundPage.test.tsx
PASS tests/components/UserProfile.test.tsx
PASS tests/components/AddPasswordDialog.test.tsx
PASS tests/components/ValidateEmail.test.tsx (10.032 s)
PASS tests/components/DeleteAccountDialog.test.tsx
PASS tests/components/NavSecondary.test.tsx
PASS tests/components/NavMain.test.tsx
PASS tests/components/Chart.test.tsx
PASS tests/pages/PageTemplate.test.tsx
PASS tests/components/DashboardCards.test.tsx
PASS tests/components/TeamSwitcher.test.tsx
PASS tests/components/UpdatePasswordDialog.test.tsx
PASS tests/lib/functions.test.tsx
PASS tests/components/ActivityList.test.tsx
PASS tests/lib/validators.test.tsx
PASS tests/components/AppSidebar.test.tsx
PASS tests/lib/icons.test.tsx
PASS tests/components/LoginForm.test.tsx (14.198 s)
PASS tests/components/RegisterForm.test.tsx (15.941 s)

Test Suites: 28 passed, 28 total
Tests: 258 passed, 258 total
Snapshots: 0 total
Time: 19.433 s
```

7 Instrukcja użytkowania

1. Wchodzimy na <https://orange-ground-00ae1ad03.6.azurestaticapps.net/> lub <https://securebox.netlify.app/> i klikamy utwórz konto i wypełniamy formularz

 **SecureBox**

Utwórz nowe konto

Wprowadź swoje dane, aby założyć konto

Imię

Nazwisko

Email

Hasło logowania

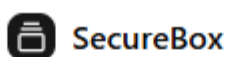
☒ Akceptuję [regulamin](#) i [politykę prywatności](#)

Utwórz konto

Masz już konto? [Zaloguj się](#)



2. Logujemy się na nowo utworzone konto za masterkey wybieramy dowolny ciąg znaków lepiej wpisywać przy każdym logowaniu ten sam ciąg znaków ponieważ odszyfrowanie haseł następuje jeśli wpiszemy poprawne masterkey, używanie wielu masterkey jest możliwe ale wymusi to wpisywanie masterkey dla każdej grupy haseł



Zaloguj się na swoje konto

Wprowadź dane

Login

Hasło logowania

[Zapomniałeś hasła?](#)

Masterkey (hasło szyfrowania)

Masterkey (hasło szyfrowania)

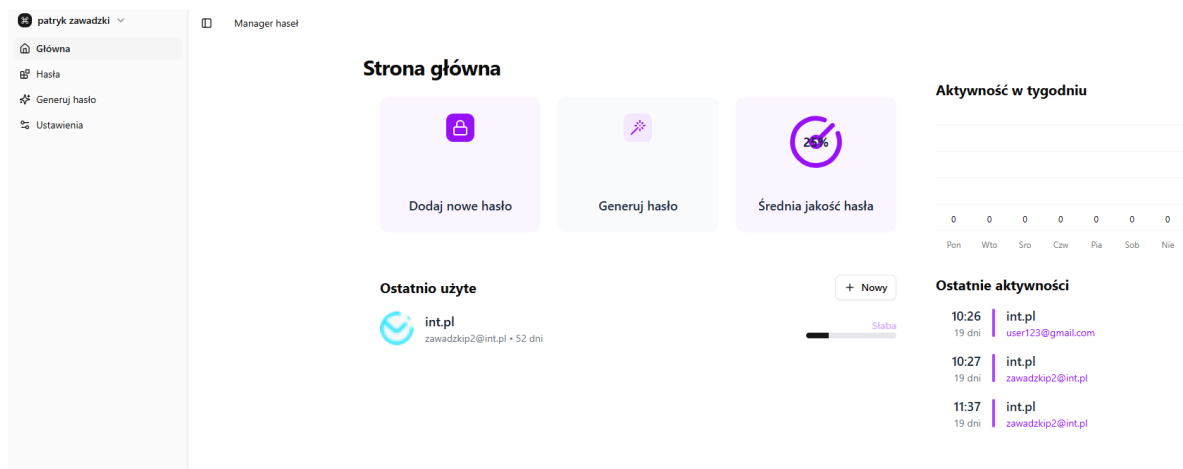


Zaloguj

Nie masz jeszcze konta? [Utwórz konto](#)



3. Po zalogowaniu widzimy stronę podzieloną na 4 zakładki (główna, hasła, Generuj hasło, ustawienia). Strona główna przedstawia aktywność w tygodniu czyli wykres liczby logowań przez rozszerzenie do różnych aplikacji w ostatnim tygodniu, ostatnie aktywności czyli 5 ostatnich logowań z użyciem rozszerzenia z uwzględnieniem kiedy to się odbyło na jakim koncie i w jakim serwisie. Ostatnio użyte pokazuje ostatnio dodane/użyte hasła i ich siłę. Średnia jakość hasła liczona jest na podstawie ostatnio użytych haseł, dodaj nowe umożliwia dodanie hasła



4. Zakładka hasła zawiera tabele z hasłami, filtrem do filtrowania loginów, przycisk kopiuj kopiuje hasło dla wskazanego konta

patryk zawadzki

Główna

Hasła

Generuj hasło








Ustawienia

Manager haseł

Hasła

Filtruj konta...

+ Dodaj

Serwis	Login ↑↓	Kopiuj
	zawadzkip2@int.pl	 ...
	user123@gmail.com	<div>Akcje</div> <div>Kopiuj hasło</div> <div>Zmień hasło</div> <div>Pokaż</div> <div>Usuń konto</div>
	user123@gmail.com	
	zawadzkip2@int.pl	
	zawadzkip21@int.pl	
		 ...

Poprzednia

Następna

5. Aby dodać hasło klikamy przycisk dodaj wpisujemy url strony , login i hasło

patryk zawadzki ▾

Manager haseł

Hasła

+ Dodaj

Dodaj hasło ×

Pozwala dodać hasło do managera haseł.

Strona

Login

Hasło

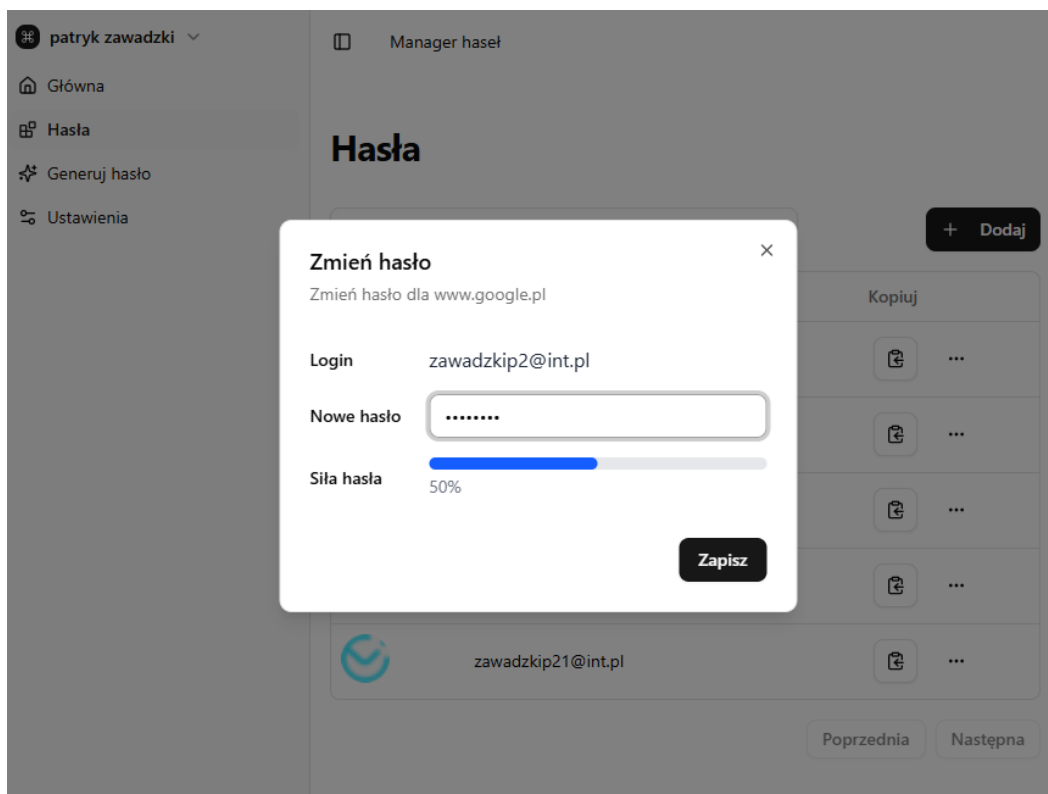
Anuluj Dodaj

Kopiuuj

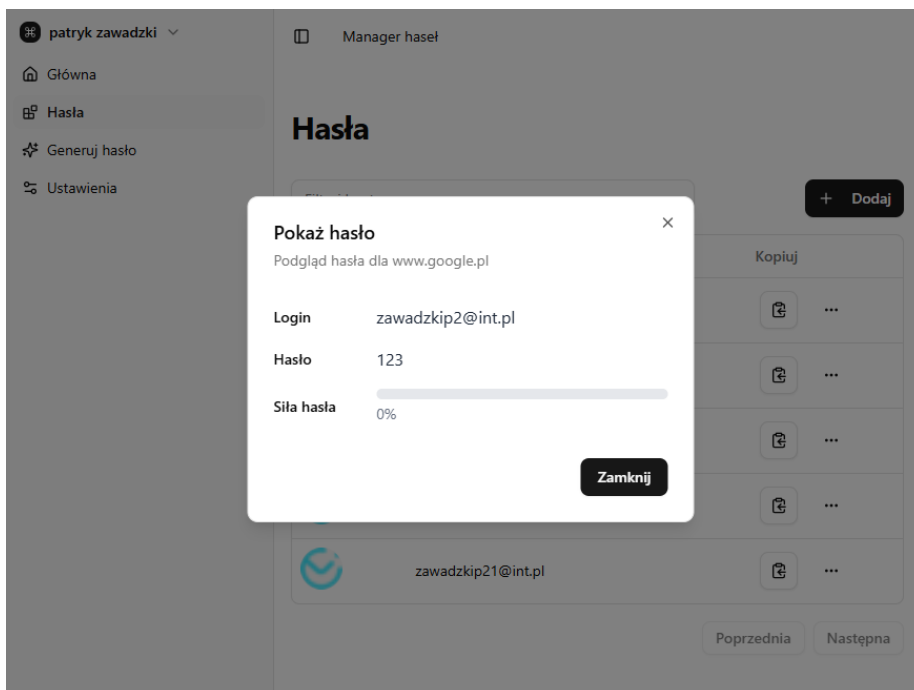
	...
	...
	...
	...
	...

Poprzednia Następna

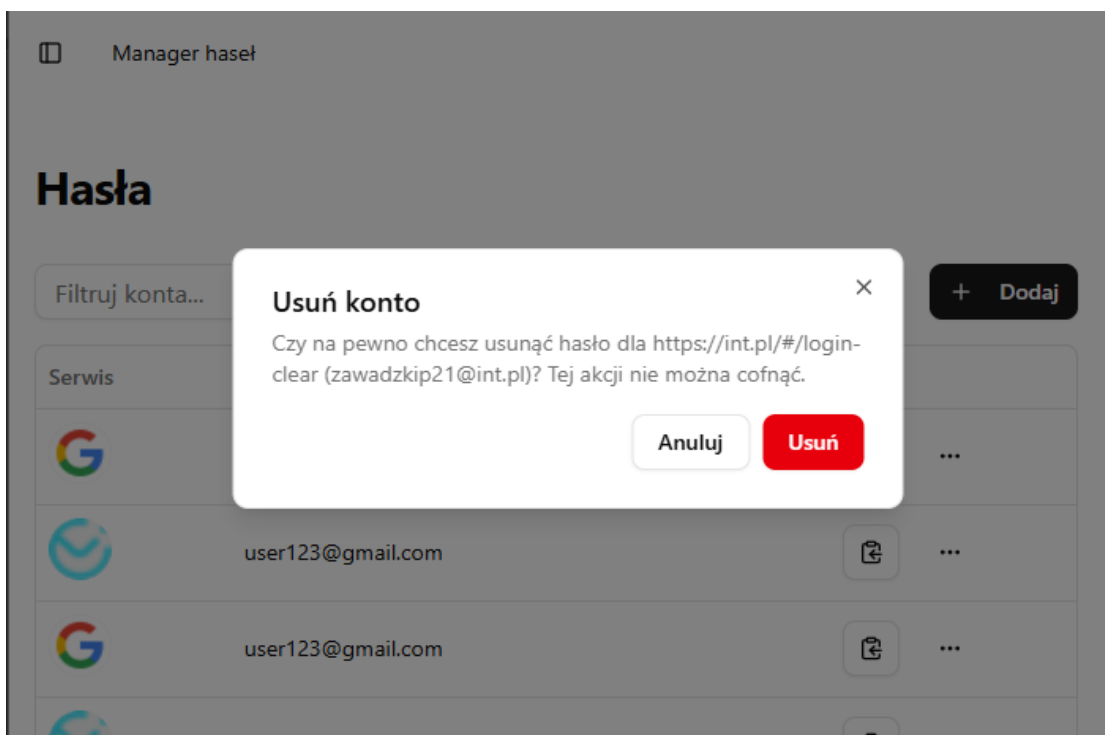
6. Aby zmienić hasło wybieramy interesujące nas konto i klikamy trzy kropki następnie Zmień hasło wpisujemy nowe hasło i mamy oszacowaną siłę nowego hasła następnie zapisujemy



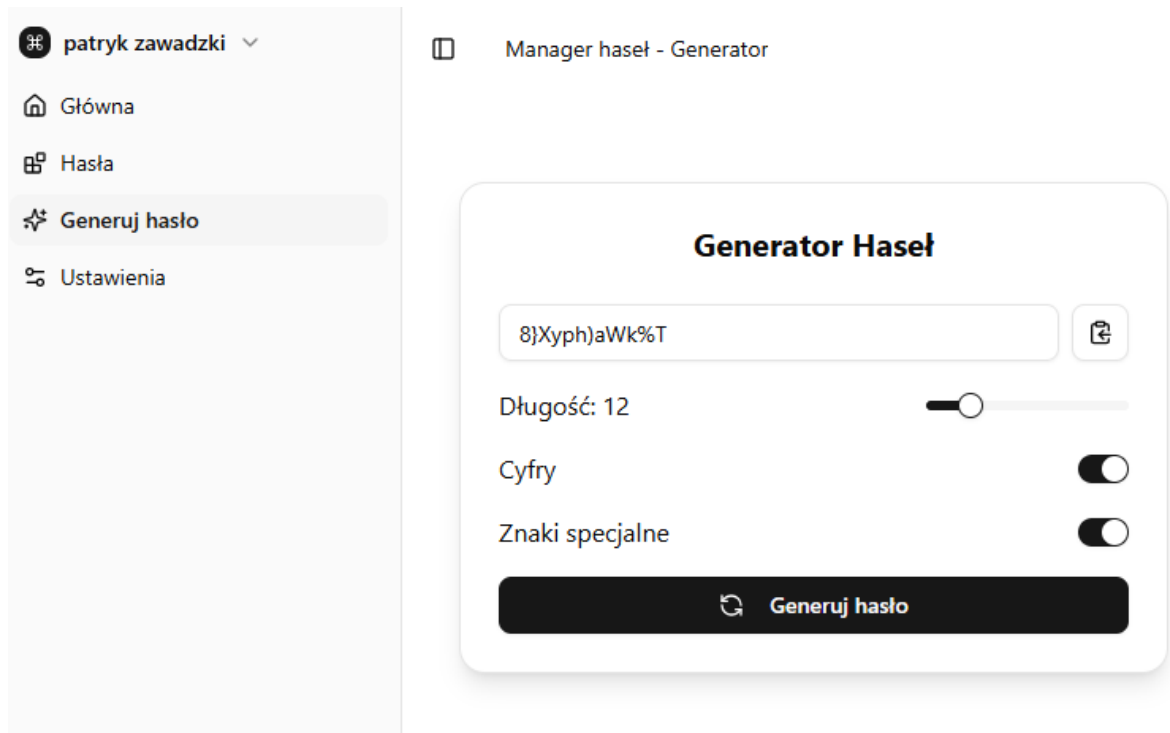
7. Aby pokazać interesujące nas hasło wybieramy interesujące nas konto trzy kropeczki i klikamy pokaż hasło widzimy hasło i oszacowaną jego siłę



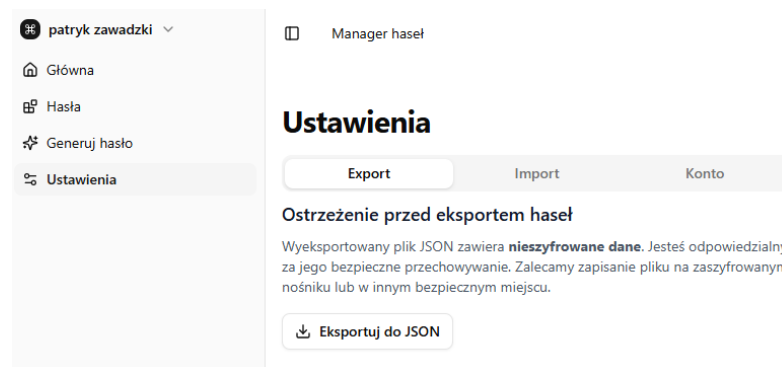
8. Aby usunąć konto wybieramy interesujące nas konto następnie trzy kropeczki usuń konto jeśli wszystko się zgadza potwierdzamy usunięcie



9. Aby wygenerować hasło należy przejść do zakładki Generuj hasło wybrać długość hasła czy ma zawierać cyfry, znaki specjalne następnie kliknąć przycisk generuj hasło, aby skopiować hasło klikamy przycisk kopiuj



10. Jeśli chcemy zapisać sobie kopie haseł przechodzimy do zakładki ustawienia – eksport i klikamy eksportuj do json (uwaga plik nie szyfrowany)

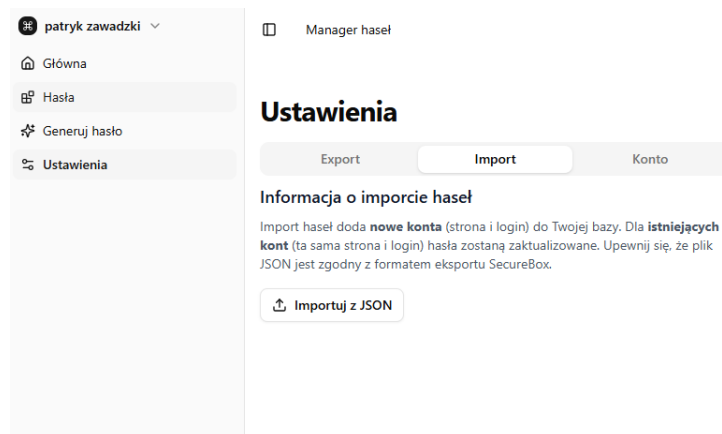


```

{
  "platform": "www.google.pl",
  "login": "zawadzkip2@int.pl",
  "password": "123"
},
{
  "platform": "https://int.pl/#/login-clear",
  "login": "user123@gmail.com",
  "password": "Patryk"
},
{
  "platform": "www.google.pl",
  "login": "user123@gmail.com",
  "password": "patryk"
},
{
  "platform": "https://int.pl/#/login-clear",
  "login": "zawadzkip2@int.pl",
  "password": "abcd"
},
{
  "platform": "https://int.pl/#/login-clear",
  "login": "zawadzkip21@int.pl",
  "password": "123"
}

```

11. Aby zaimportować hasła na początek należy utworzyć plik o strukturze podobnej do pliku powyżej zawierający platform login i password. Przejść do zakładki ustawienia-import i kliknąć przycisk importuj z json i wskazać właściwy plik. Wszystkie nowe hasła zostaną dodane a stare zaktualizowane



12. Aby zobaczyć bądź zmienić swoje dane należy przejść do zakładki ustawienia – konto, kliknąć edytuj i wpisać nowe dane w polach które chcemy zaktualizować a następnie klikamy zapisz

patryk zawadzki

Główna

Hasła

Generuj hasło

Ustawienia

Manager haseł

Ustawienia

Export Import Konto

Login: zawadzkip2@int.pl

Imię: patryk

Nazwisko: zawadzki

Edytuj

Informacja o zmianie Masterkey

Po zmianie Masterkey musisz zaktualizować go w **rozszerzeniu SecureBox**, jeśli go używasz, aby zapewnić spójność. Zmiana Masterkey jest **niemożliwa**, jeśli używasz wielu Masterkey, ponieważ wymaga odszyfrowania i ponownego zaszyfrowania wszystkich haseł jednym kluczem. Upewnij się, że używasz jednego Masterkey.

Zmień Masterkey

13. W celu zmiany masterkey przechodzimy do zakładki ustawienia – konto i klikamy przycisk zmień masterkey. Zmiana będzie możliwa tylko wtedy gdy używamy jednego masterkey dla wszystkich haseł. Wpisujemy stare masterkey i nowe masterkey a następnie zapisujemy

Manager haseł

Ustawienia

Export Import Konto

Login

Imię

Nazwisko

Informacja o zmianie Masterkey

Po zmianie Masterkey musisz zaktualizować go w **rozszerzeniu SecureBox**, jeśli go używasz, aby zapewnić spójność. Zmiana Masterkey jest **niemożliwa**, jeśli używasz wielu Masterkey, ponieważ wymaga

Zmień Masterkey

Aktualizuj swoje hasło szyfrowania (masterkey).

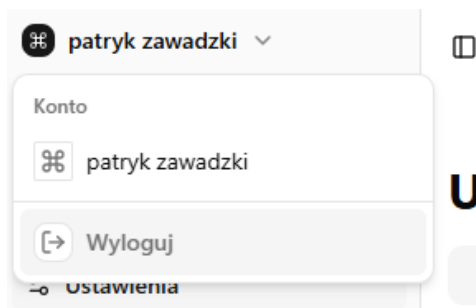
Stare Masterkey: ...

Nowe Masterkey:

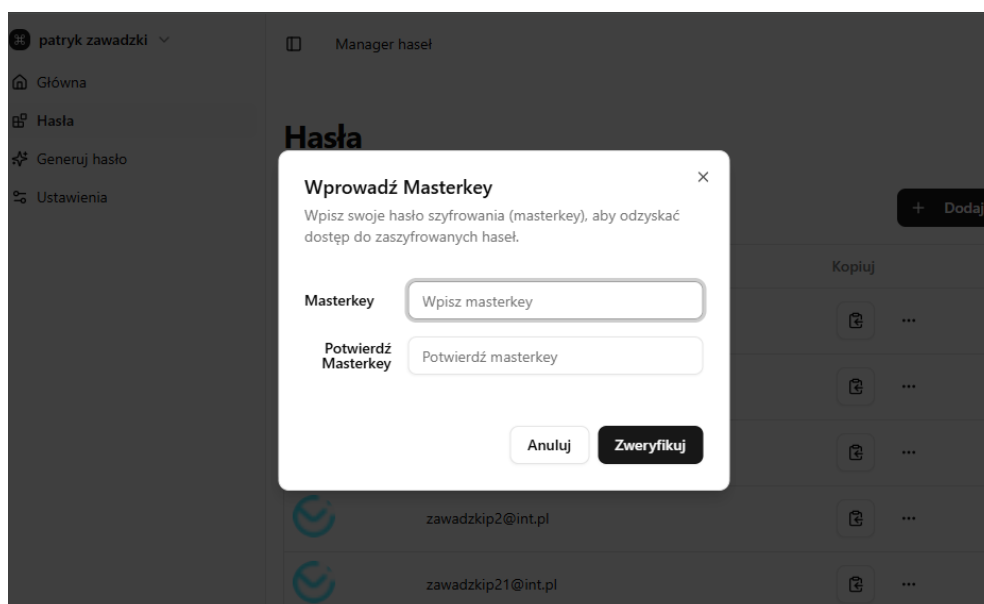
Potwierdź nowe Masterkey:

Anuluj Zapisz

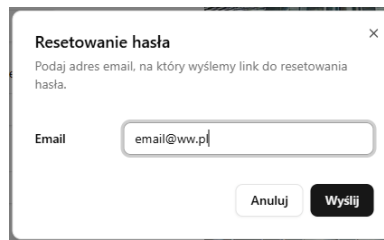
14. Aby się wylogować klikamy w zakładkach na swoje imię i nazwisko następnie przycisk wyloguj



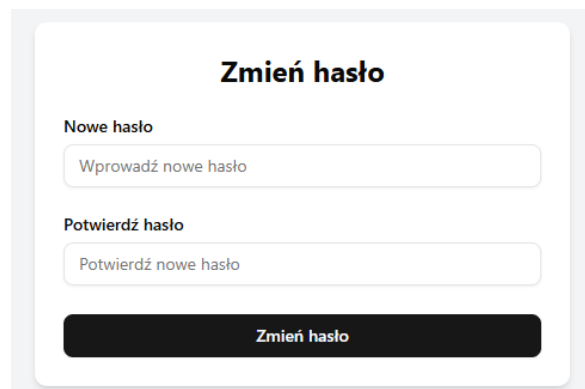
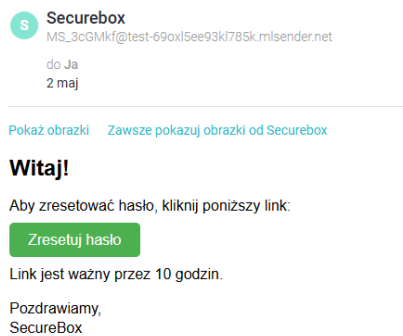
15. Gdyby użytkownik wpisał złe masterkey podczas logowania to gdyby chciał odczytać jakieś hasło zostanie poproszony o wpisanie prawidłowego masterkey to samo w przypadku użycia wielu masterkey



16. Aby zresetować hasło należy wejść na stronę logowania i kliknąć link „zomniałeś hasła?” wpisać email, następnie wejść na pocztę kliknąć otrzymany link i wpisać nowe hasło



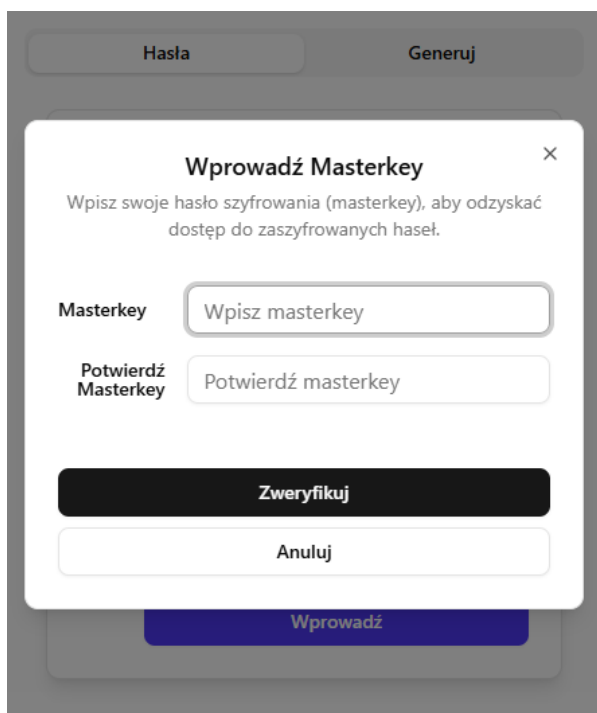
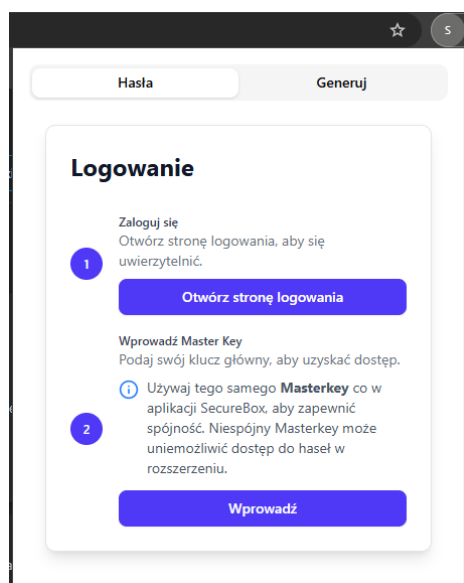
A modal dialog box titled "Resetowanie hasła" with a close button (X) in the top right corner. Below the title, it says "Podaj adres email, na który wyślemy link do resetowania hasła." There is a text input field labeled "Email" containing the text "email@ww.pl". At the bottom right, there are two buttons: "Anuluj" (light gray) and "Wyślij" (dark gray).



A form titled "Zmień hasło" with a light gray background. It contains two text input fields: "Nowe hasło" with the placeholder "Wprowadź nowe hasło" and "Potwierdź hasło" with the placeholder "Potwierdź nowe hasło". At the bottom, there is a large black button labeled "Zmień hasło".

7.1 Użycie rozszerzenia

1. Aby się zalogować klikamy okno dialogowe klikamy link do logowania i logujemy się, następnie w oknu dialogowym wpisujemy masterkey rozszerzenia















2. Aby wygenerować nowe hasło należy przejść do zakładki generuj wskazać interesującą długość hasła, czy ma zawierać cyfry i czy ma zawierać znaki specjalne, a następnie generuj, aby skopiować hasło należy kliknąć przycisk kopiuj

3. Aby się zalogować na wybrane konto należy w zakładce hasła kliknąć ikonkę logowania w wybranym koncie- spowoduje to otwarcie nowego okna i zautomatyzowane logowanie

Serwis	Login	Zaloguj
	zawadzkip2@int.pl	
	user123@gmail.com	
	user123@gmail.com	
	zawadzkip2@int.pl	
	zawadzkip21@int.pl	
	user7	


4. Aby dodać nowe konto należy na stronie logowania do wybranej aplikacji wprowadzić poświadczenia i otworzyć rozszerzenie i kliknąć znak dodaj spowoduje to dodanie bądź zaktualizowanie hasła


5. Aby zaktualizować masterkey kliknij na przycisk do zmiany masterkey – używa się go wtedy gdy na frontendzie zmieni się masterkey

Serwis	Login ↑↓	Zaloguj
	zawadzkip2@int.pl	
	user123@gmail.com	
	user123@gmail.com	
	zawadzkip2@int.pl	
	zawadzkip21@int.pl	
	user7	

Previous

Next

 Wyloguj

 Zmieni Masterkey

6. Aby się wylogować kliknij przycisk wyloguj

8 Co można zmienić aby aplikacja nadawała się do środowiska produkcyjnego


Aby aplikacja menedżera haseł, składająca się z backendu w Node.js, frontendu w React oraz rozszerzenia Chrome opartego na React, działająca na maszynie wirtualnej w Azure, była gotowa do środowiska produkcyjnego, należy wprowadzić następujące zmiany w celu zapewnienia skalowalności, niezawodności, wydajności i bezpieczeństwa, zgodnie z najlepszymi praktykami oraz standardami ISO/IEC 27017 i 27018.

Po pierwsze, dla statycznej części aplikacji, czyli frontendu w React, zaleca się użycie usługi Azure Static Web Apps w planie standardowym - zapewni automatyczne skalowanie, globalną dystrybucję treści przez sieć CDN, wsparcie dla niestandardowych domen, wbudowane uwierzytelnianie oraz integrację z CI/CD, co redukuje koszty. Plan standardowy oferuje większą przepustowość i zaawansowane funkcje zarządzania ruchem sieciowym, co jest kluczowe dla środowiska produkcyjnego.

Po drugie, typ maszyny wirtualnej należy zmienić z B2s na serię Dsv5, na przykład D2s v5, która oferuje lepszą wydajność procesora i pamięci, zapewniając odpowiednią moc obliczeniową dla backendu Node.js przy zachowaniu efektywności kosztowej. Seria Dsv5 jest zoptymalizowana dla aplikacji wymagających wyższej wydajności, co wspiera skalowalność i stabilność w środowisku produkcyjnym.

Po trzecie, dla maszyny wirtualnej należy skonfigurować usługę Azure Backup, aby zapewnić regularne kopie zapasowe danych i konfiguracji. Azure Backup umożliwia automatyczne tworzenie migawek maszyny wirtualnej, przechowywanie ich w bezpiecznym magazynie Recovery Services Vault z szyfrowaniem AES-256, oraz definiowanie harmonogramów tworzenia kopii (np. codzienne lub cotygodniowe) i polityk przechowywania (np. 30 dni dla kopii krótkoterminowych i 12 miesięcy dla długoterminowych). W przypadku awarii, dane można przywrócić w ciągu kilku minut, co minimalizuje czas przestoju i spełnia wymagania dotyczące odzyskiwania danych w chmurze. Dodatkowo, należy włączyć monitorowanie za pomocą Azure Monitor, skonfigurować automatyczne skalowanie backendu za pomocą Azure Virtual Machine Scale Sets oraz wdrożyć Azure Application Gateway z Web Application Firewall (WAF) dla dodatkowej ochrony przed atakami DDoS i exploitami. Wszystkie wrażliwe dane, takie jak klucze API, powinny być przechowywane w Azure Key Vault, a dostęp do nich ograniczony przez Azure Role-Based Access Control (RBAC).

Po czwarte baza danych powinna być osobną usługą a nie być hostowana na maszynie wirtualnej, oczywiście pomiędzy bazą a maszyną wirtualną należy utworzyć zasady sieciowe aby z bazą mogła się łączyć jedynie maszyna wirtualna. Wdrożenie tych zmian zapewni zgodność z regulacjami GDPR, wysoką dostępność, odporność na awarie oraz bezpieczeństwo danych, co jest zgodne z celami przedmiotu Bezpieczeństwo Usług Chmurowych.

 **Usługa Azure Backup dla maszyn wirtualnych platformy Azure — Zapraszamy!**
Proste i niezawodne tworzenie kopii zapasowych maszyn wirtualnych na platformie Azure. [Learn more](#) Oplata za wystąpienie jest naliczana i Recovery Services. [Dowiedz się więcej o cenach](#).

Przejrzyj poniższe informacje, a następnie kliknij pozycję „Włącz kopię zapasową”, aby rozpocząć ochronę maszyny wirtualnej.

Magazyn usług Recovery Services ⓘ ☒ Utwórz nowe ☐ Wybierz istniejącą

Backup vault * vault988 ✓

Grupa zasobów vm ▼
[Utwórz nowy](#)

Podtyp zasad * ☒ Rozszerzone

- ✓ Wiele kopii zapasowych dziennie
- ✓ Do 30 dni przechowywania w warstwie operacyjnej
- ✓ Obsługa zaufanego uruchamiania maszyny wirtualnej platformy Azure
- ✓ Obsługa maszyn wirtualnych z dyskami w warstwie Ultra i dyskami SSD v2 w warstwie Premium

☐ Standard

- ✓ Kopia zapasowa raz dziennie
- ✓ Do 5 dni przechowywania w warstwie operacyjnej

Wybierz zasady tworzenia kopii zapasowej * ⓘ (nowy) EnhancedPolicy-mb5647bk ▼
[Edytuj te zasady](#)

i Ochrona maszyny wirtualnej za pomocą rozszerzonych zasad może wiązać się z dodatkowymi opłatami za migawki. Pamiętaj, że po włączeniu tworzenia kopii zapasowych maszyny wirtualnej za pomocą rozszerzonych zasad zmiana na typ zasad standardowych nie jest możliwa. [Dowiedz się więcej](#).

7 Regulamin aplikacji

Regulamin aplikacji SecureBox

Data wejścia w życie: 04 maja 2025 r.

Niniejszy Regulamin określa zasady korzystania z aplikacji SecureBox oraz jej rozszerzenia przeglądarkowego. Korzystanie z Aplikacji oznacza akceptację Regulaminu oraz zgodę na przetwarzanie danych osobowych.

Definicje:

- **Aplikacja SecureBox:** Oprogramowanie do zarządzania hasłami.
- **Rozszerzenie:** Dodatek do przeglądarki umożliwiający automatyczne logowanie.
- **Użytkownik:** Osoba korzystająca z Aplikacji lub Rozszerzenia.
- **Masterkey:** Klucz główny do szyfrowania haseł.

Dane osobowe: Informacje zgodne z RODO.

Ogólne warunki korzystania Korzystanie z Aplikacji wymaga akceptacji Regulaminu. Aplikacja jest przeznaczona dla osób pełnoletnich. Użytkownik zobowiązuje się do przestrzegania prawa.

Zasady funkcjonowania Masterkey:

1. Użytkownik ustala Masterkey, który nie jest przechowywany na serwerach. Utrata Masterkey uniemożliwia dostęp do haseł.
2. Przy logowaniu Użytkownik jest informowany o ryzyku utraty Masterkey.
3. Możliwe jest użycie wielu Masterkey, ale uniemożliwia to eksport haseł do JSON.
4. W Rozszerzeniu Użytkownik musi używać spójnego Masterkey, o czym jest informowany przy logowaniu.
5. Eksport haseł Użytkownik może eksportować hasła do pliku JSON (nieszyfrowanego). Przed eksportem otrzymuje ostrzeżenie o konieczności bezpiecznego przechowywania danych. Dostawca nie odpowiada za utratę wyeksportowanych danych.
6. Monitorowanie logowań w Rozszerzeniu

7. Monitorowanie obejmuje 5 ostatnich logowań i statystyki tygodniowe.
8. Dane logowań przechowywane są przez 5 lat, nieudane logowania przez 2 lata.
9. Przetwarzanie danych wymaga zgody Użytkownika lub jest częścią umowy.
10. Blokada konta Po 5 nieudanych próbach logowania konto jest blokowane na 10 minut.
11. Zabezpieczenie logowania Logowanie jest zabezpieczone za pomocą reCAPTCHA.
Użycie reCAPTCHA podlega polityce prywatności Google.
12. Statystyki haseł Statystyki siły haseł są obliczane lokalnie w przeglądarce i mają charakter poglądowy. Średnia jakość haseł uwzględnia tylko hasła użyte w przeglądarce.
13. Ciasteczka i technologie przechowywania danych
14. Ciasteczka HTTP-only przechowują token uwierzytelniający.
15. LocalStorage/SessionStorage przechowuje dane sesji, usuwane po wylogowaniu.
16. Użytkownik jest informowany o ciasteczkach przy pierwszym uruchomieniu.
17. Odpowiedzialność Dostawcy Dostawca nie odpowiada za utratę Masterkey, nieautoryzowany dostęp do wyeksportowanych haseł ani przerwy w działaniu Aplikacji z przyczyn niezależnych.

Wsparcie: support@securebox.com.

Ochrona danych osobowych

Administrator: SecureBox Sp. z o.o., ul. Przykładowa 1, 00-000 Warszawa.

Cele: Świadczenie usług, monitorowanie logowań, bezpieczeństwo.

Okres przechowywania: Logowania - 5 lat, nieudane logowania - 2 lata, pozostałe dane - do 2 lat po zakończeniu korzystania.

Prawa Użytkownika: Dostęp, sprostowanie, usunięcie, przenoszenie danych, skarga do PUODO.

Rozwiązanie umowy Użytkownik może usunąć konto w ustawieniach. Dane zostaną trwale usunięte, z wyjątkiem danych wymaganych przez prawo.

Zmiany Regulaminu Zmiany Regulaminu są ogłaszane z 14-dniowym wyprzedzeniem. Dalsze korzystanie oznacza akceptację.

Postanowienia końcowe

Regulamin podlega prawu polskiemu i unijnemu. Spory rozstrzyga sąd w Warszawie. Wersja polska ma pierwszeństwo.

9 Polityka prywatności

Polityka prywatności SecureBox

Data wejścia w życie: 04 maja 2025 r.

Niniejsza Polityka prywatności opisuje, jak SecureBox Sp. z o.o. przetwarza dane osobowe użytkowników aplikacji SecureBox i jej rozszerzenia przeglądarkowego. Dbamy o ochronę Twojej prywatności i przestrzegamy przepisów RODO.

Administrator danych Administratorem Twoich danych osobowych jest SecureBox Sp. z o.o., ul. Przykładowa 1, 00-000 Warszawa, NIP: 1234567890. Kontakt: support@securebox.com.

Cele i podstawy prawne przetwarzania Przetwarzamy Twoje dane w następujących celach:

1. Świadczenie usług Aplikacji i Rozszerzenia (wykonanie umowy, art. 6 ust. 1 lit. b RODO).
2. Rejestracja i uwierzytelnianie użytkownika (wykonanie umowy, art. 6 ust. 1 lit. b RODO).
3. Monitorowanie logowań i prób logowania dla bezpieczeństwa (uzasadniony interes, art. 6 ust. 1 lit. f RODO).
4. Zarządzanie hasłami (wykonanie umowy, art. 6 ust. 1 lit. b RODO).
5. Obsługa resetowania hasła (wykonanie umowy, art. 6 ust. 1 lit. b RODO).
6. Zarządzanie zaufanymi urządzeniami (uzasadniony interes, art. 6 ust. 1 lit. f RODO).
7. Rodzaje przetwarzanych danych Przetwarzamy następujące dane:
8. Dane użytkownika: Imię, nazwisko, login (e-mail), zaszyfrowane hasło.
9. Hasła: Login do platformy, nazwa platformy, URL ikony strony (logo).
10. Próby logowania: Czas próby, sukces/niepowodzenie, powiązanie z użytkownikiem.
11. Logowania w rozszerzeniu: Czas logowania, login, URL strony, powiązanie z użytkownikiem.
12. Tokeny resetowania hasła: Token, czas wygaśnięcia, powiązanie z użytkownikiem.
13. Zaufane urządzenia: Identyfikator urządzenia, dane przeglądarki (user-agent), status zaufania.
14. Dane techniczne: Adres IP, wersja przeglądarki (user-agent).
15. Dane z reCAPTCHA: Ciasteczka Google, adres IP.

16. Okres przechowywania danych Twoje dane przechowujemy przez następujące okresy:
17. Dane użytkownika: Przez okres korzystania z Aplikacji oraz 2 lata od ostatniego logowania.
18. Hasła: Do momentu usunięcia przez użytkownika lub 2 lata po zamknięciu konta.
19. Próby logowania: 2 lata od daty próby.
20. Logowania w rozszerzeniu: 5 lat od daty logowania.
21. Tokeny resetowania hasła: Do momentu wygaśnięcia tokenu (np. 24 godziny).
22. Zaufane urządzenia: Do momentu usunięcia urządzenia przez użytkownika lub zamknięcia konta.

Odbiorcy danych Twoje dane mogą być przekazywane:

Podmiotom świadczącym usługi hostingu i analityki (np. dostawcy serwerów).

Google (reCAPTCHA) - dane mogą być przekazywane do USA na podstawie Data Privacy Framework.

Twoje prawa

1. Masz prawo do dostępu, sprostowania, usunięcia, ograniczenia przetwarzania, przenoszenia danych, sprzeciwu.
2. Cofnięcia zgody w dowolnym momencie (jeśli dotyczy).
3. Wniesienia skargi do Prezesa Urzędu Ochrony Danych Osobowych.
4. Bezpieczeństwo danych Stosujemy następujące środki bezpieczeństwa:
5. Szyfrowanie end-to-end dla haseł (Masterkey).
6. Szyfrowanie transmisji danych (HTTPS).
7. Ciasteczka HTTP-only dla tokenów uwierzytelniających.
8. reCAPTCHA dla ochrony przed botami.
9. Regularne audyty bezpieczeństwa.
10. Ciasteczka i technologie Używamy:
11. Ciasteczek HTTP-only do przechowywania tokenów uwierzytelniających (niezbędne do logowania).
12. LocalStorage/SessionStorage do danych sesji (usuwane po wylogowaniu).
13. reCAPTCHA - zobacz Politykę prywatności Google.

14. Przekazywanie danych do państw tperich Dane mogą być przekazywane do USA (Google reCAPTCHA) na podstawie Data Privacy Framework. Stosujemy odpowiednie zabezpieczenia, takie jak standardowe klauzule umowne.

Kontakt W sprawach ochrony danych pisz na: support@securebox.com.

10 Logi

Aplikacja menedżera haseł wykorzystuje Morgan i PM2 do logowania w celu monitorowania działania systemu i zapewnienia bezpieczeństwa w środowisku chmurowym Azure na maszynie wirtualnej. Morgan, middleware dla Express, rejestruje szczegóły żądań HTTP, takie jak metoda, URL, kod statusu, adres IP klienta i czas odpowiedzi, w formacie combined, zapisując je w pliku `/app/access.log`. Folder `/app` ma uprawnienia 755, a plik `access.log` 640, co ogranicza dostęp do autoryzowanych użytkowników. Logi są przechowywane lokalnie na maszynie wirtualnej bez przesyłania do zewnętrznych systemów. Umożliwiają one analizę żądań pod kątem incydentów bezpieczeństwa, takich jak wielokrotne nieudane logowania wskazujące na ataki brute-force, oraz wspierają audyt zgodności z GDPR i ISO/IEC 27017. PM2, menedżer procesów Node.js, uruchamia aplikację i rejestruje logi systemowe (`stdout`, `stderr`), błędy oraz treści zapytań do bazy danych PostgreSQL w pliku `/app/pm2.log`. Zapytania do bazy, takie jak `SELECT`, `INSERT` czy `UPDATE`, są logowane w formacie JSON za pomocą middleware w aplikacji (np. poprzez `console.log` w obsłudze zapytań z użyciem pakietu `pg`), z wykluczeniem wrażliwych danych, takich jak hasła użytkowników.

Logi PM2 są przechowywane lokalnie z uprawnieniami 640 i wspierają monitorowanie stabilności aplikacji oraz identyfikację błędów, np. nieudanych połączeń z bazą danych. PostgreSQL, używany jako baza danych, generuje własne logi w pliku `/var/lib/postgresql/data/pg_log/postgresql.log` na maszynie wirtualnej, rejestrując operacje takie jak połączenia, błędy zapytań czy nieautoryzowane próby dostępu. Logi PostgreSQL są skonfigurowane w pliku `postgresql.conf` z ustawieniami `log_statement='all'` dla pełnego rejestrowania zapytań i `log_min_error_statement=ERROR` dla błędów, z uprawnieniami 600 dla bezpieczeństwa. Wszystkie logi (`/app/access.log`, `/app/pm2.log`, `/var/lib/postgresql/data/pg_log/postgresql.log`) są przechowywane lokalnie, zabezpieczone odpowiednimi uprawnieniami i regularnie sprawdzane przez administratora w celu audytu bezpieczeństwa i analizy incydentów, zgodnie z wymaganiami GDPR i ISO/IEC 27017.

11 Podsumowanie

Projekt menedżera haseł został zrealizowany, obejmując kluczowe aspekty aplikacji, w tym jej bezpieczeństwo, zgodnie z wymaganiami środowiska chmurowego Azure. Zaimplementowano backend w Node.js z API REST, frontend w React oraz rozszerzenie Chrome, zapewniając funkcjonalność zarządzania hasłami. Bezpieczeństwo aplikacji oparto na szyfrowaniu haseł (AES-256), zabezpieczeniach sieciowych (TLS, firewall w Azure), ochronie przed atakami (np. CSRF, XSS) oraz audycie logów (Morgan, PM2, PostgreSQL), co spełnia wymagania GDPR i ISO/IEC 27017. Dokumentacja projektu jest wystarczająco szczegółowa, opisując architekturę, wdrożenie, bezpieczeństwo i procedury backupu, wspierając efekty uczenia się przedmiotu. Z powodu ograniczeń czasowych nie wprowadzono mechanizmu zaufanych urządzeń ani weryfikacji dwuetapowej, co ogranicza zaawansowane zabezpieczenia uwierzytelniania. Nie osiągnięto 100% pokrycia kodu testami jednostkowymi i integracyjnymi z powodu braku czasu, co może wpływać na niezawodność w środowisku produkcyjnym. Aby aplikacja nadawała się do środowiska produkcyjnego, należy zrealizować kroki opisane w sekcji „Co można zmienić aby aplikacja nadawała się do środowiska produkcyjnego”, w tym wprowadzenie weryfikacji dwuetapowej, pełne pokrycie testami oraz mechanizm zaufanych urządzeń, aby zwiększyć bezpieczeństwo i skalowalność.

12 Dokumentacja

Frontend: <https://patrykzawadzkiisggw.github.io/SecureBox-frontend/>

Backend: <https://patrykzawadzkiisggw.github.io/SecureBox/>

Rozszerzenie: <https://patrykzawadzkiisggw.github.io/password-extension/>

13 Bibliografia

- Toroman, M., Janetscheck T.: Mastering Azure Security Second Edition
- Griggs B., Spigolon M.: Node.js Cookbook
- <https://fakerjs.dev/guide/usage.html>
- <https://www.npmjs.com/package/nodemon?activeTab=readme>
- <https://nodejs.org/api/modules.html>
- <https://dev.to/imsushant12/securing-nodejs-applications-best-practices-and-strategies-38c>
- <https://docs.snyk.io/snyk-cli/commands/code-test>
- <https://www.w3schools.com/CSSref/index.php>
- <https://docs.npmjs.com/>