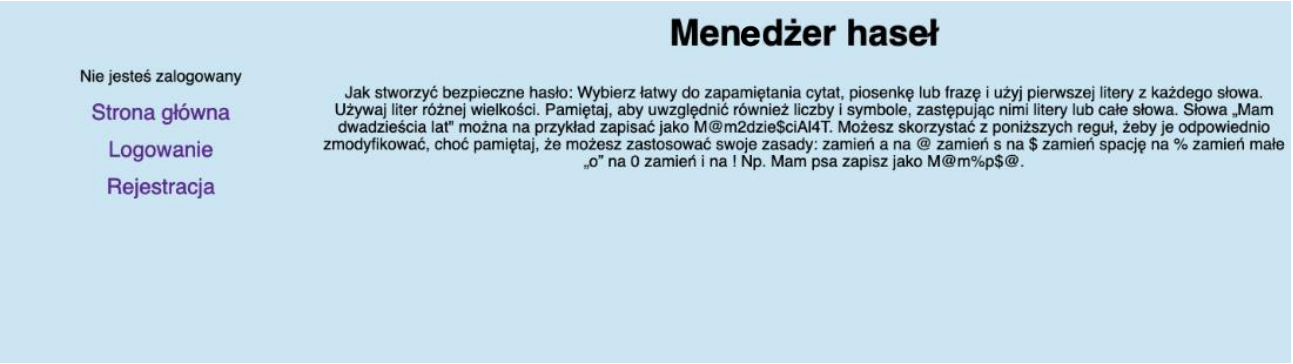
Aleksandra Kowalczyk

307414

Sprawozdanie z wykonania projektu: „bezpieczna aplikacja internetowa służąca do przechowywania haseł do różnych witryn z możliwością udostępniania haseł innym użytkownikom”

0. Aplikacja zawiera trzy główne panele:

- Strona główna, powitalna – ze wskazówką jak tworzyć bezpieczne hasło
- Logowanie
- Rejestracja



1. **restrykcyjna walidacja danych z negatywnym nastawieniem:**

Dane wejściowe podlegają walidacji na wstępie, by uchronić aplikacje przed niepożądanymi próbami ataku np. wstrzyknięcia kodu.

W pliku .js są zdefiniowane i sprawdzane wpisywane dane do poszczególnych pól:

- Hasło ma zawierać od 8 do 40 znaków, musi zawierać co najmniej jedną wielką, małą literę, cyfrę i znak specjalny
- Nazwa użytkownika ma zawierać od 3 do 25 znaków
- Nazwa użytkownika może zawierać jedynie litery wielkie lub małe bądź cyfry
- Nazwy użytkownika nie mogą się powtarzać (aplikacja sprawdza, czy nazwa jest już zajęta)
- Adres e-mail musi spełniać pewne wymogi zapisu (wykryje, gdy podamy nieprawidłowy adres)
- Pytania awaryjne nie mogą się powtarzać, należy wybrać dwa inne


```
const passwordRegex = /^.{8,40}$/;
const usernameRegex = /^[a-zA-Z0-9]{3,25}$/;
const emailRegex = /^[a-z0-9]+([\._]?[a-z0-9])+[@]\w+([.]\w+)+$/;
const answerRegex = /^[\wĄĆĘŁŃÓŚŹŻąćęłńóśźż.\-,\/ ]+$/;
```

2. przechowywanie hasła chronione funkcją hash, solą i pieprzem

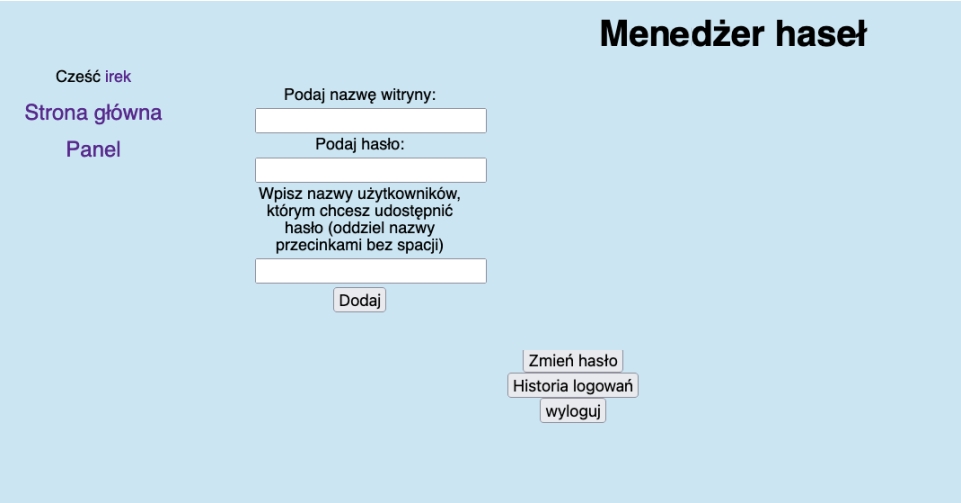
Hasło użytkownika jest przechowywane jako hash utworzony przy użyciu soli i pieprzu:

```
pepper = ''.join((secrets.choice(string.ascii_letters + string.digits + string.punctuation) for i in range(15)))
pepper = str.encode(pepper)
password = hashpw(password.encode(), gensalt(14) + pepper)
user["password"] = password
```

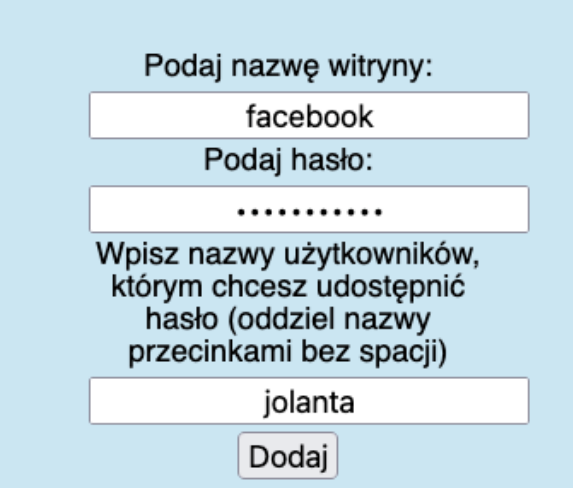
3. możliwość umieszczenia na serwerze haseł dostępnych prywatnie lub dla określonych użytkowników, zarządzanie uprawnieniami do zasobów

Aplikacja umożliwia udostępnienie hasła innemu użytkownikowi.

Należy przy dodawaniu witryny podać nazwy kont, których chcemy udostępnić hasło.

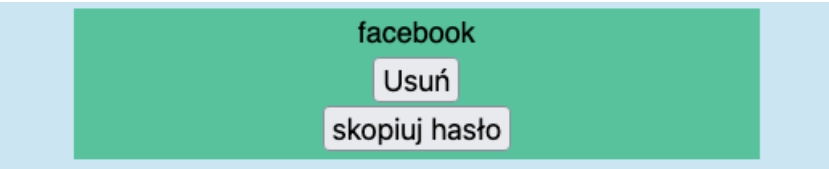


Przykład: użytkownik „irek” dodaje nowe hasło, które chce udostępnić użytkownikowi „jolanta”:

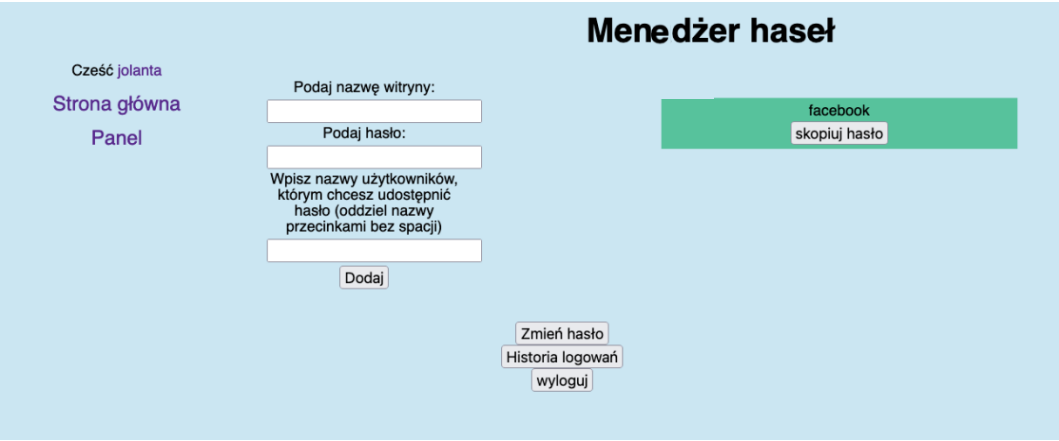


Użytkownik „irek” może usunąć wpis, bądź skopiować hasło do schowka.

Podczas procesu kopiowania hasło jest odszyfrowywane.



Użytkownik „jolanta” ma możliwość tylko skopiowania hasła, nie może go usunąć.



4. szyfrowanie symetryczne przechowywanych haseł

Do przechowywania zapisywanych haseł do witryn używam szyfrowania symetrycznego AES z trybem CTR.

```
i = randrange(9)
key_aes = keys_aes_list[i]

aes_enc = AES.new(key_aes, AES.MODE_CTR)
nonce = aes_enc.nonce

data = request.form.get('web_password')
data_bytes = bytes(data, 'utf-8')
cipher_text = aes_enc.encrypt(data_bytes)
```

5. zabezpieczenie transmisji poprzez wykorzystanie protokołu https

Transmisja odbywa się pod adresem https zabezpieczona certyfikatem:


Certyfikat

127.0.0.1

Nazwa podmiotu

Państwo	PL
Województwo	Lodzkie
Region	Lodz
Organizacja	ALEKSANDRA KOWALCZYK
Jednostka organizacyjna	AK
Nazwa pospolita	127.0.0.1
Adres e-mail	ak@op.pl

Nazwa wystawcy

Państwo	PL
Województwo	Lodzkie
Region	Lodz
Organizacja	ALEKSANDRA KOWALCZYK
Jednostka organizacyjna	AK
Nazwa pospolita	127.0.0.1
Adres e-mail	ak@op.pl

Ważność

Nieważny przed	Thu, 27 Jan 2022 13:16:32 GMT
Nieważny po	Fri, 27 Jan 2023 13:16:32 GMT

Informacje o kluczu publicznym

Algorytm	RSA
Rozmiar klucza	4096
Wykładnik	65537
Modulo	C8:73:72:0C:3C:5B:ED:F8:AB:4A:48:8E:F4:3F:37:1B:9A:3B:6D:92:63:AB:C...

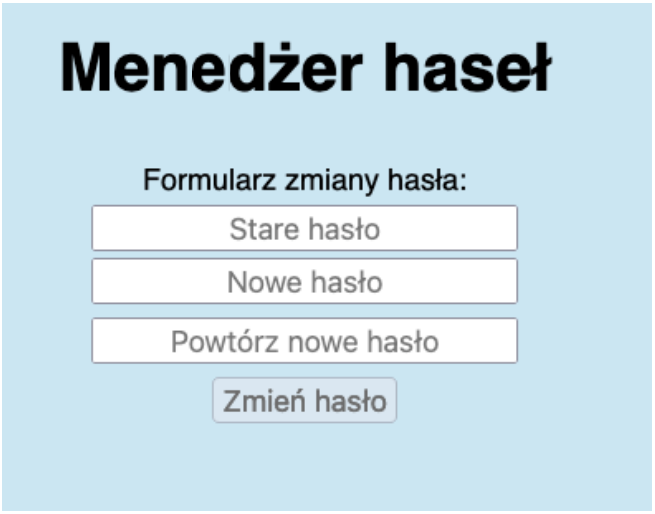
Różne

Numer seryjny	4D:E3:67:CC:7B:3B:A8:52:17:B1:7D:B1:08:CB:8A:B6:97:C7:64:78
Algorytm podpisu	SHA-256 with RSA Encryption
Wersja	3
Pobierz	PEM (certyfikat) PEM (łańcuch)

Odciski

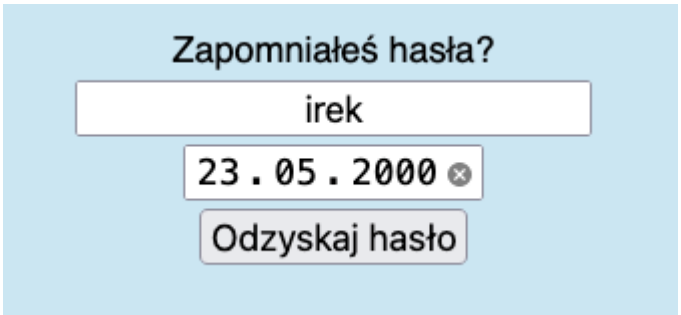
SHA-256	3B:ED:AA:4B:F3:F6:B1:8F:D2:FA:51:A2:EB:03:5B:82:9E:4E:8E:F1:18:08:39:...
SHA-1	16:D4:43:F6:88:63:A8:C3:2E:41:28:42:DC:8B:65:15:CE:FF:7B:01

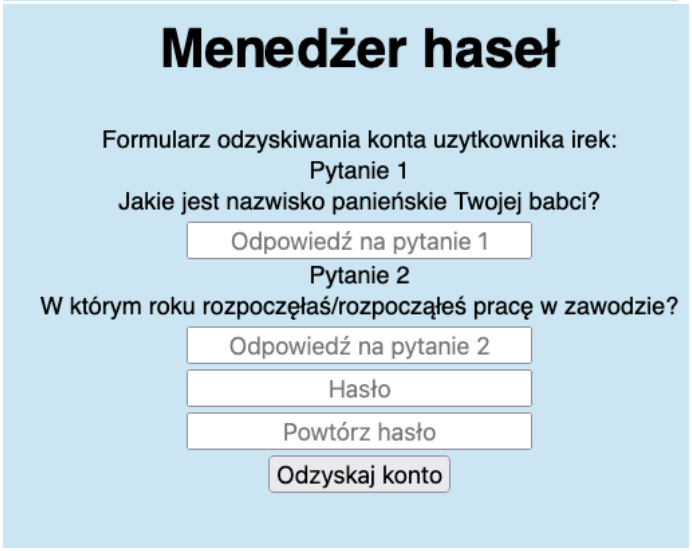
6. **możliwość zmiany hasła**



7. **możliwość odzyskania dostępu w przypadku utraty hasła**

W przypadku utraty hasła użytkownik może odzyskać hasło podając swoją nazwę i datę urodzenia, a następnie odpowiadając na pytania zabezpieczające:





Nie jest to najbezpieczniejszy sposób odzyskiwania hasła, ponieważ mimo, że odpowiedzi na pytania awaryjne są znane wąskiej grupie osób, nie są one tajne. Dodatkowo obecnie na wiele pytań można znaleźć odpowiedź na portalach społecznościowych.

Z tych powodów najlepiej byłoby umożliwić odzyskiwanie konto przy pomocy linku aktywacyjnego wysyłanego na skrzynkę mailową użytkownika.

8. dodatkowa kontrola spójności sesji (przeciw atakom CSRF i XSRF)

W celu ochrony przed atakami CSRF w plikach .html żądania “POST” i “GET” zabezpieczam poprzez token csfr:

```
<input type="hidden" name="csrf_token" value="{{ csrf_token() }}"/>
```

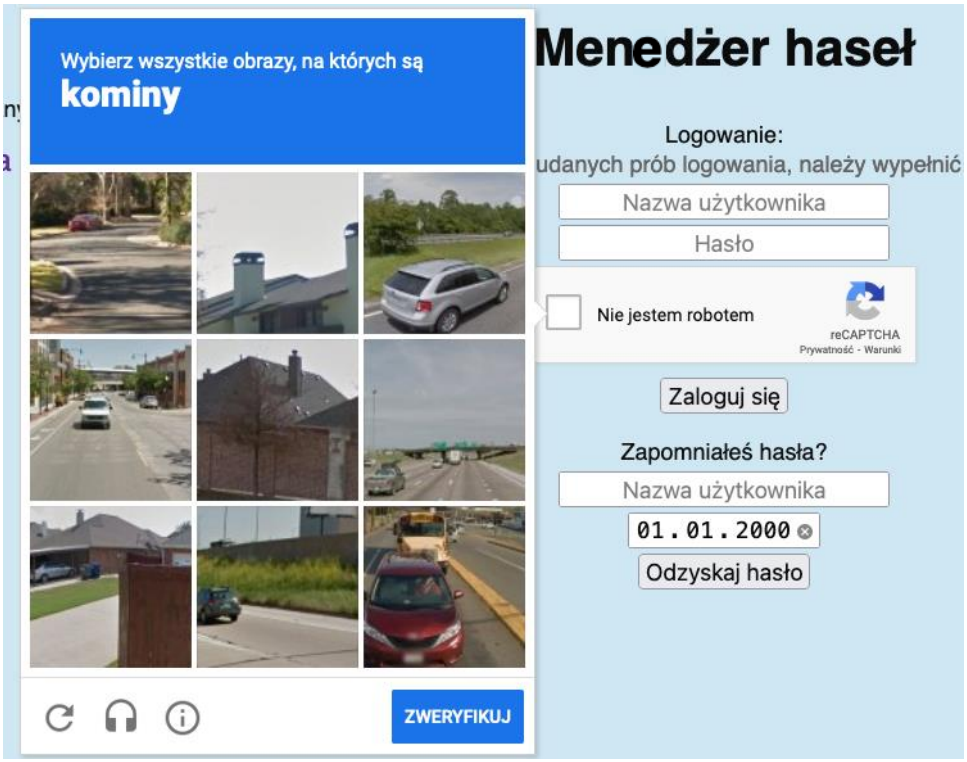
By powyższy kod mógł działać należy zdefiniować “SECRET KEY” aplikacji.

9. weryfikacja liczby nieudanych prób logowania

Aplikacja zlicza nieudane próby logowania, po przekroczeniu maksymalnej ilości prób (są to 3 podejścia) użytkownik musi wypełnić formularz RECAPTCHA, by móc się zalogować:

```
captcha = False
if attempts is not None:
    captcha = attempts > MAX_ATTEMPTS
if captcha:
    if not recaptcha.verify():
        flash("Zbyt wiele nieudanych prób logowania, należy wypełnić formularz captcha")
        return redirect(url_for("sign_in", captcha_needed=captcha))
```

Google umożliwia darmowe wygenerowanie klucza strony SITE_KEY i klucza sekretnego SECRET_KEY, które umożliwiają dodanie zabezpieczenia RECAPTCHA do własnej aplikacji.



10. dodanie opóźnienia przy weryfikacji hasła w celu wydłużenia ataków zdalnych

Podczas procesu logowania generowana jest randomowa liczba z zakresu od 1 do 5, a następnie dodawane jest takie opóźnienie:

```
rand = random.randint(1, 5)
time.sleep(rand)
```

11. sprawdzanie jakości hasła (jego entropii), kontrola siły hasła, żeby uświadomić użytkownikowi problem

Aplikacja wymaga od użytkownika, by hasło zawierało co najmniej jedną wielką, małą literę, cyfrę i znak specjalny. Hasło musi mieć co najmniej 8 znaków. Dodatkowo obliczana jest entropia(losowość) hasła ze wzoru:

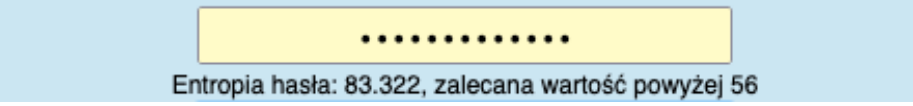
$$pw\_length * \log_2 alphabet\_length$$

gdzie pw_length to liczba znaków w haśle, a $\log_2 alphabet\_length$ to liczba bitów w jednym znaku.

Długość alfabetu ($alphabet\_length$) dla samych małych liter to 26 , dla samych wielkich liter 26, dla cyfr 10, dla znaków specjalnych 25. Jeśli nasze

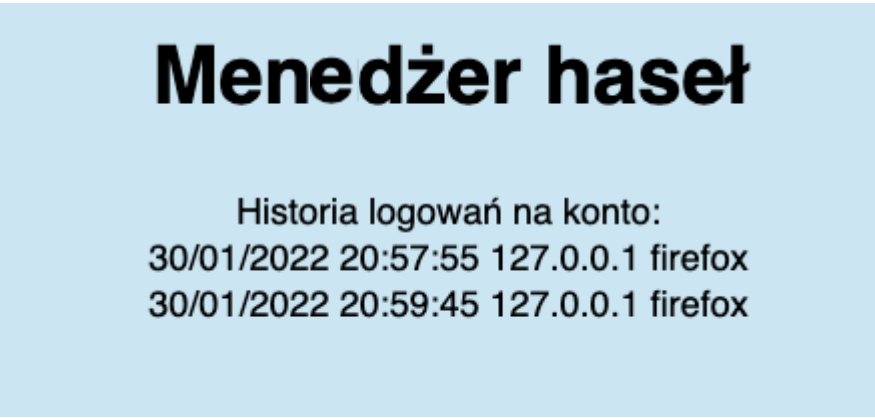
hasło zawiera co najmniej jeden znak z każdej grupy to nasz alfabet ma długość 87.

Użytkownik jest informowany, ile wynosi entropia jego hasła i jaka wartość jest zalecana (zalecana wartość powyżej 56).



12. Monitorowanie pracy systemu, informowanie użytkownika o nowych podłączeniach do jego konta

Historia logowań na konto jest zapisywana i wyświetlana użytkownikowi w odpowiednim module. Wyświetlana jest data, dokładna godzina logowania i użyta przeglądarka. Dzięki temu użytkownik może kontrolować czy nie nastąpiło logowanie wykonane nie przez niego.



13. ograniczone informowanie o błędach

W pewnych sytuacjach np. w przypadku błędu połączenia z bazą nie informuję o tym użytkownika, wyświetlam jedynie komunikat o tymczasowym błędzie:

```
flash("Nastąpił błąd, spróbuj później")
```

Podczas próby logowania/odzyskania hasła nie informuję użytkownika, które pole jest złe, wyświetlam ogólną informacje:

```
flash("Niepoprawne nazwa użytkownika i/lub hasło")
```