

Akademia Techniczno-Humanistyczna w Bielsku-Białej

Projekt
Cyberbezpieczeństwo

Sprawozdanie nr 1

Systemy bezpieczeństwa oparte na hasłach

GRUPA:2a / SEMESTR:7 / ROK: 2022

NAZWISKO I IMIĘ / NR INDEKSU

1. Aleksander Michalec / 055932

2. Jakub Wykręt / 055837

3. Szymon Białek / 055872

1. Program powinien zapewniać pracę w dwóch rolach: administratora (użytkownik o stałej nazwie ADMIN) oraz zwykłego użytkownika.

```
8     public class User : EntityBase, IAggregateRoot
9     {
10         public string Login { get; set; } = string.Empty;
11         public string Password { get; set; } = string.Empty;
12         public UserRole Role { get; set; }
13         public DateTime? ExpirationTime { get; set; }
14         public bool IsBlocked { get; set; }
15         public virtual ICollection<UsedPassword>? UsedPasswords { get; set; }
16     }
```

Rys. 1

Jak można zauważyć na Rys. 1, model użytkownika, gdzie pole Role jest typu enum, przechowuje dwie role: Administrator (0) oraz User (1).

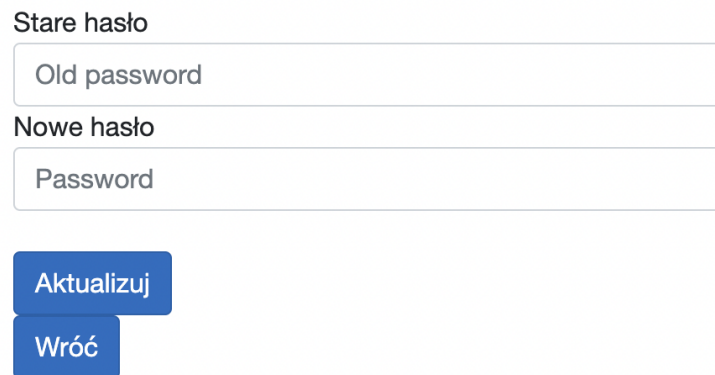
Za pomocą znaczników i atrybutów z klasy AuthenticationStateProvider (Rys. 2), które rozszerzyliśmy o własne Identities, ustalamy która rola ma dostęp do poszczególnych stron.

```
12     <div class="nav-item px-3">
13         <NavLink class="nav-link" href="" Match="NavLinkMatch.All">
14             <span class="oi oi-home" aria-hidden="true"></span> Home
15         </NavLink>
16     </div>
17     <AuthorizeView Roles="Administrator">
18         <div class="nav-item px-3">
19             <NavLink class="nav-link" href="/uzytkownicy">
20                 <span class="oi oi-plus" aria-hidden="true"></span> Użytkownicy
21             </NavLink>
22         </div>
23
24         <div class="nav-item px-3">
25             <NavLink class="nav-link" href="/settings">
26                 <span class="oi oi-list-rich" aria-hidden="true"></span> Ustawienia
27             </NavLink>
28         </div>
29     </AuthorizeView>
```

```
1     @page "/settings"
2     @attribute [Authorize(Roles = "Administrator")]
3
4     <h1>Ustawienia</h1>
5
```

Rys. 2

2. W rolę administratora program musi zawierać następujące funkcje:



Stare hasło

Nowe hasło

Aktualizuj

Wróć

Rys. 3

Na Rys. 3 przedstawiliśmy możliwość zmiany hasła przez Administratora

User1

Usuń użytkownika

Data wygaśnięcia:

23.10.2022 20:01:11

Edytuj

Login

User1

Stare hasło

Old password

Nowe hasło

Password

Ważność hasła

2

dni

Zablokuj użytkownika ☐

Aktualizuj

Wróć

Rys. 4

Na Rys. 4 zaprezentowaliśmy możliwość modyfikacji, blokowania oraz usuwania konta użytkownika.

Dodaj użytkownika

Login: Hasło: Wygaśnięcie hasła: ☐

Rys. 5

Natomiast Rys. 5 prezentuje możliwość dodawania użytkownika przez Admina. Ustawia on również termin wygaśnięcia hasła.

Cyberbezpieczeństwo

Home

Użytkownicy

Ustawienia

Użytkownicy

Login	Hasło wygasa:
ADMIN	
User1	21.10.2022 10:04:30
User3	13.11.2022 22:00:00
User4	24.10.2022 22:00:00
User5	19.10.2022 22:00:00
User5	22.10.2022 22:00:00
User6	29.10.2022 12:34:35
User8	21.10.2022 13:34:32

Rys. 6

Ta część pokazuje, że Admin może również przeglądać listę użytkowników i modyfikować ich dane.(Rys. 6)

Ustawienia

Złożoność hasła:

☒ Co najmniej 8 znaków ☒ Co najmniej jedna wielka litera ☒ Co najmniej jeden znak specjalny

Rys. 7

Ta część program pokazuje, że Admin może ustawiać złożoność hasła. (Rys. 7)

Natomiast przy zakończeniu pracy, po wyłączeniu karty następuje reset autentyfikacji.

3. W roli użytkownika program powinien zawierać tylko funkcje zmiany hasła użytkownika (jeśli stare hasło jest wpisane poprawnie) i zakończenia pracy

User3

Edytuj

Stare hasło

Old password

Nowe hasło

Password

Aktualizuj

Wróć

Rys. 8

Rys. 8 prezentuje możliwości użytkownika. Może on tylko zmieniać swoje hasło.

4. Po uruchomieniu program powinien poprosić użytkownika podać swój identyfikator i hasło do konta w specjalnym oknie logowania. Wprowadzając hasło, jego znaki należy zawsze zastąpić wyświetlanym na ekranie symbolem "•".

LOGIN

Login

User3

Password

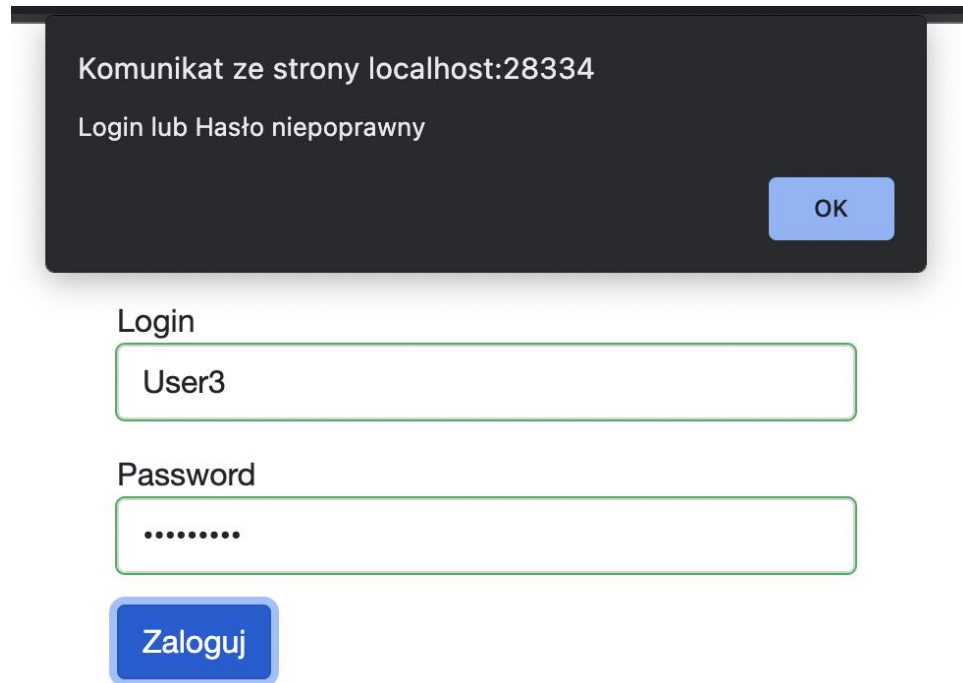
••••••|

Zaloguj

Rys. 9

Możemy tutaj zauważyć, że po uruchomieniu aplikacji program wykryje, że nie jesteśmy zalogowani, więc przekieruje nas do strony logowania. (Rys.9)

5. Komunikat w przypadku wprowadzenia niepoprawnego identyfikatora lub hasła: „Login lub Hasło niepoprawny”.



The image shows a web application interface. At the top, a dark grey modal box displays an error message in white text: "Komunikat ze strony localhost:28334" followed by "Login lub Hasło niepoprawny". A blue "OK" button is in the bottom right of the modal. Below the modal, the login form is visible. It has a "Login" label above a text input field containing "User3". Below that is a "Password" label above a password input field filled with dots. At the bottom of the form is a blue "Zaloguj" button.

Rys. 10

Rys. 10 prezentuje nam błąd, który wyskakuje jak wprowadzimy niepoprawne hasło.

6. Przy pierwszym logowaniu system powinien prosić o zmianę hasła dostępu ustalonego przez administratora na hasło własne, znane tylko użytkownikowi, utworzone według zadania indywidualnego. Nowe hasło należy podać oraz powtórzyć.

Twoje hasło wymaga zmiany

Hasło

Potwierdź hasło

Aktualizuj

Rys. 11

Zaprezentowaliśmy tutaj wiadomość, która wyświetla się użytkownikowi, gdy ten musi zmienić swoje hasło. (Rys. 11)

7. Należy stosować bezpieczny algorytm hashujący do przechowywania haseł.

```
public class HashProvider
{
    public static string HashPassword(string password)
    {
        byte[] salt;
        byte[] buffer2;
        if (password == null)
        {
            throw new ArgumentNullException("password");
        }
        using (Rfc2898DeriveBytes bytes = new Rfc2898DeriveBytes(password, 0x10, 0x3e8))
        {
            salt = bytes.Salt;
            buffer2 = bytes.GetBytes(0x20);
        }
        byte[] dst = new byte[0x31];
        Buffer.BlockCopy(salt, 0, dst, 1, 0x10);
        Buffer.BlockCopy(buffer2, 0, dst, 0x11, 0x20);
        return Convert.ToBase64String(dst);
    }
}
```

Rys. 12

	Id [PK] integer	Login text	Password text
	1	ADMIN	AKCR1THZKROiOzsL/LhA4tl...
	2	User1	AAYFrct0UONFU/Aj7QJJD+e...

Rys. 13

Rys. 12 prezentuje wycinek z kodu, gdzie zapisany jest algorytm hashujący. Natomiast Rys. 13 pokazuje zhashowane już hasło zapisane w bazie.

8. Zadania indywidualne(Ograniczenia dotyczące haseł użytkowników)

Hasło musi zawierać:

1. co najmniej 8 znaków, co najmniej jedną wielką literę, co najmniej jeden znak specjalny;

```

8 public class LoginValidator : AbstractValidator<UserDTO>
9 {
10     private PasswordLimitationController _limitationController { get; set; }
11
12     public LoginValidator(bool first, bool second, bool third)
13     {
14         RuleFor(form => form.Password).NotEmpty().WithMessage("Hasło nie może być puste")
15         .MinimumLength(8).WithMessage("Hasło musi składać się z co najmniej 8 znaków");
16
17         if (second)
18             RuleFor(form => form.Password).Matches(@"[A-Z]").WithMessage("Hasło musi się składać z co najmniej jednej wielkiej litery");
19
20         if (third)
21             RuleFor(form => form.Password).Matches(@"[!@?*\.\.].").WithMessage("Hasło musi zawierać co najmniej jeden znak specjalny");
22     }
23 }

```

Rys. 14

Stare hasło

Nowe hasło

- Hasło musi składać się z co najmniej 8 znaków
- Hasło musi się składać z co najmniej jednej wielkiej litery
- Hasło musi zawierać co najmniej jeden znak specjalny

Aktualizuj

Wróć

Rys. 15

Rys. 14 prezentuje wycinek z kodu, na którym mamy ograniczenia dla hasła. Natomiast Rys. 15 pokazuje błędy, które wyskakują w aplikacji, jeśli wprowadzimy niepoprawne hasło.