Uwierzytelnianie i autoryzacja użytkowników

Projektowanie i programowanie systemów internetowych I

mgr inż. Krzysztof Rewak

6 maja 2018

Wydział Nauk Technicznych i Ekonomicznych
Państwowa Wyższa Szkoła Zawodowa im. Witelona w Legnicy

Plan prezentacji

1. Definicje
2. Uwierzytelnianie z sesją
3. Uwierzytelnianie bez sesji
4. Autoryzacja
5. Podsumowanie

Definicje

Użytkownik

Kim jest **użytkownik**?

Użytkownik

Uogólniając będzie to osoba korzystająca z systemu - w tym wypadku, internetowego.

Użytkownik

Czyli kto?

- gość wchodzący pierwszy raz na naszą witrynę?
- osoba znająca hasło dostępu?
- osoba z prywatnym zestawem hasła i nazwy użytkownika?
- odpytanie przez API?

Użytkownik

```
public class User {

    private String id;
    private String name;
    private String password;

}
```

Czy to wystarczy do obsługi użytkowników?

Uwierzytelnianie

Jeżeli system internetowy udostępnia możliwość tzw. zalogowania się użytkownika, należy wówczas mówić o mechanizmie **uwierzytelniania**.

Uwierzytelnianie

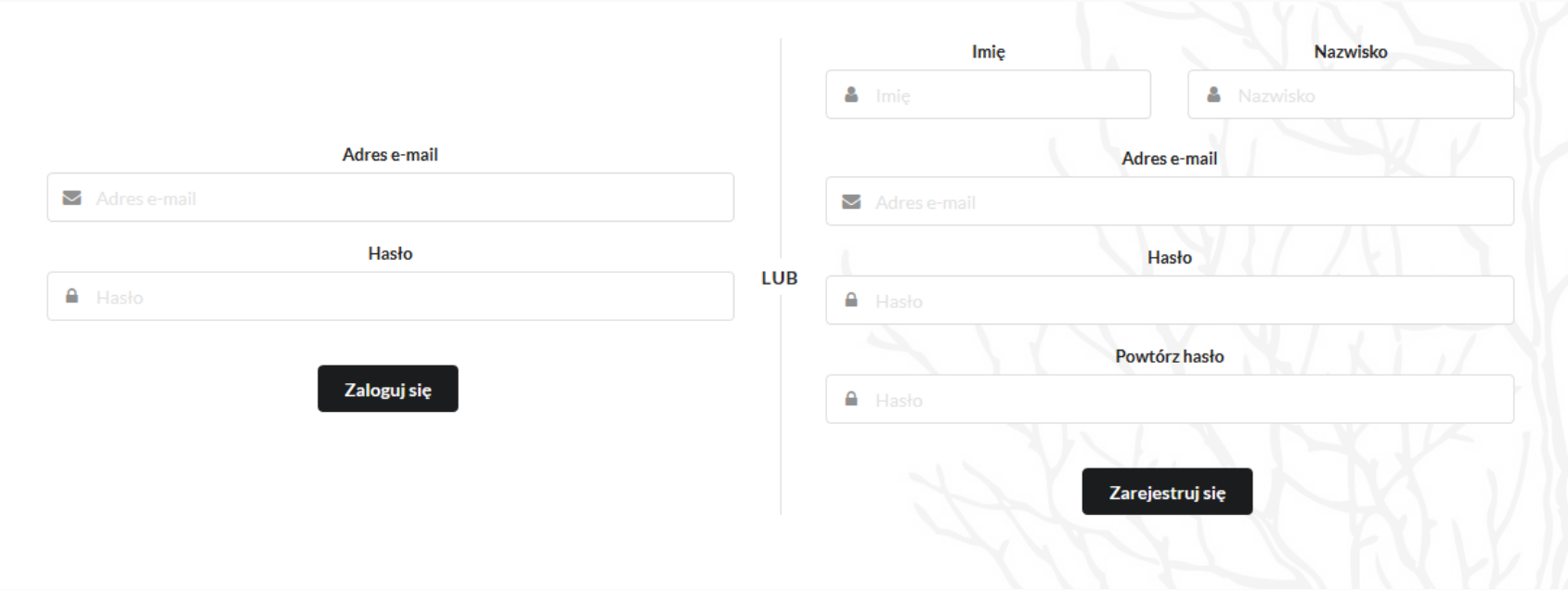
Uwierzytelnianie (ang. *authentication*) (ale nigdy nie *autentykacja*!), czyli weryfikacja użytkownika, najczęściej polega na porównaniu uprzednio zapisanych danych z danymi wpisywanymi przez użytkownika.

Autoryzacja

Nigdy nie należy mylić uwierzytelniania z **autoryzacją**, która jest sprawdzeniem czy dany użytkownik może wykonać daną operację.

Uwierzytelnianie z sesją

Gdzie spotkać uwierzytelnianie?



Rysunek 1: Przykładowy formularz logowania i rejestracji

Gdzie spotkać uwierzytelnianie?

- poczta elektroniczna
- bankowość elektroniczna
- portale społecznościowe
- aplikacje użytkowe

Gdzie spotkać uwierzytelnianie?

- praktycznie wszędzie

Czym można uwierzytelniać?

W podstawowym wariancie będzie to **unikalna** nazwa użytkownika lub jego adres poczty elektronicznej oraz prywatne hasło.

Niektóre systemy umożliwiają logowanie się loginem, mejlem lub numerem telefonu. Inne wymagają zapamiętania odgórnie narzuconej nazwy użytkownika. Jaka jest najsensowniejsza forma? To już zależy od upodobań klienta.

Crème de la crème

```
public function loginAction(Request $request): Response {

  $user = $this->findUserByLogin($request->login);

  if($user) {
    if($this->checkHash($request->password, $user->password)) {
      $this->registerSession($user);
      $this->responseArray->setSuccessStatus();

      return $this->renderResponse();
    }
  }

  $this->security->hash(rand());
  return $this->renderResponse();

}
```

Krok po kroku

Uwierzytelnianie może wyglądać następująco:

1. użytkownik przesyła swój login i hasło przez formularz
2. sprawdzane jest czy użytkownik o takim loginie istnieje
3. jeżeli istnieje, wówczas sprawdzane jest czy przesłane hasło pasuje do tego uprzednio zapisanego
4. jeżeli pasuje, wówczas użytkownik powinien zostać zalogowany
5. jeżeli hasło nie pasuje lub login nie istnieje, użytkownik nie powinien zostać zalogowany

Krok w bok

O czym warto pamiętać, nad czym warto się zastanowić, co warto zaimplementować?

- użytkownik musi zostać poinformowany o statusie akcji, ale niekoniecznie o wszystkich jej szczegółach (*podany błędy login lub hasło* zamiast *taki użytkownik w bazie danych nie istnieje*)
- czy każdy użytkownik może się zalogować? (co ze zbanowanymi lub nieaktywowanymi?)
- czy obsługa formularza jest zabezpieczona?

Ale czym tak naprawdę jest logowanie?

Do uwierzytelniania wykorzystuje się pojęcie **sesji**, która jest zapamiętywana na serwerze i najczęściej identyfikowana przez przechowywany w ciasteczku identyfikator.

Sesję można przechowywać w plikach, bazie danych, systemach pamięci podręcznej lub innych formach. Większość nowoczesnych frameworków webowych oferuje wbudowane i gotowe rozwiązania sesji.

Więc czym tak naprawdę jest wylogowanie?

```
public function logoutAction(): Response {
  $this->session->remove("auth");

  $this->responseArray
    ->setMessage("Wylogowano poprawnie.")
    ->setSuccessStatus();

  return $this->renderResponse();
}
```

Uwierzytelnianie bez sesji

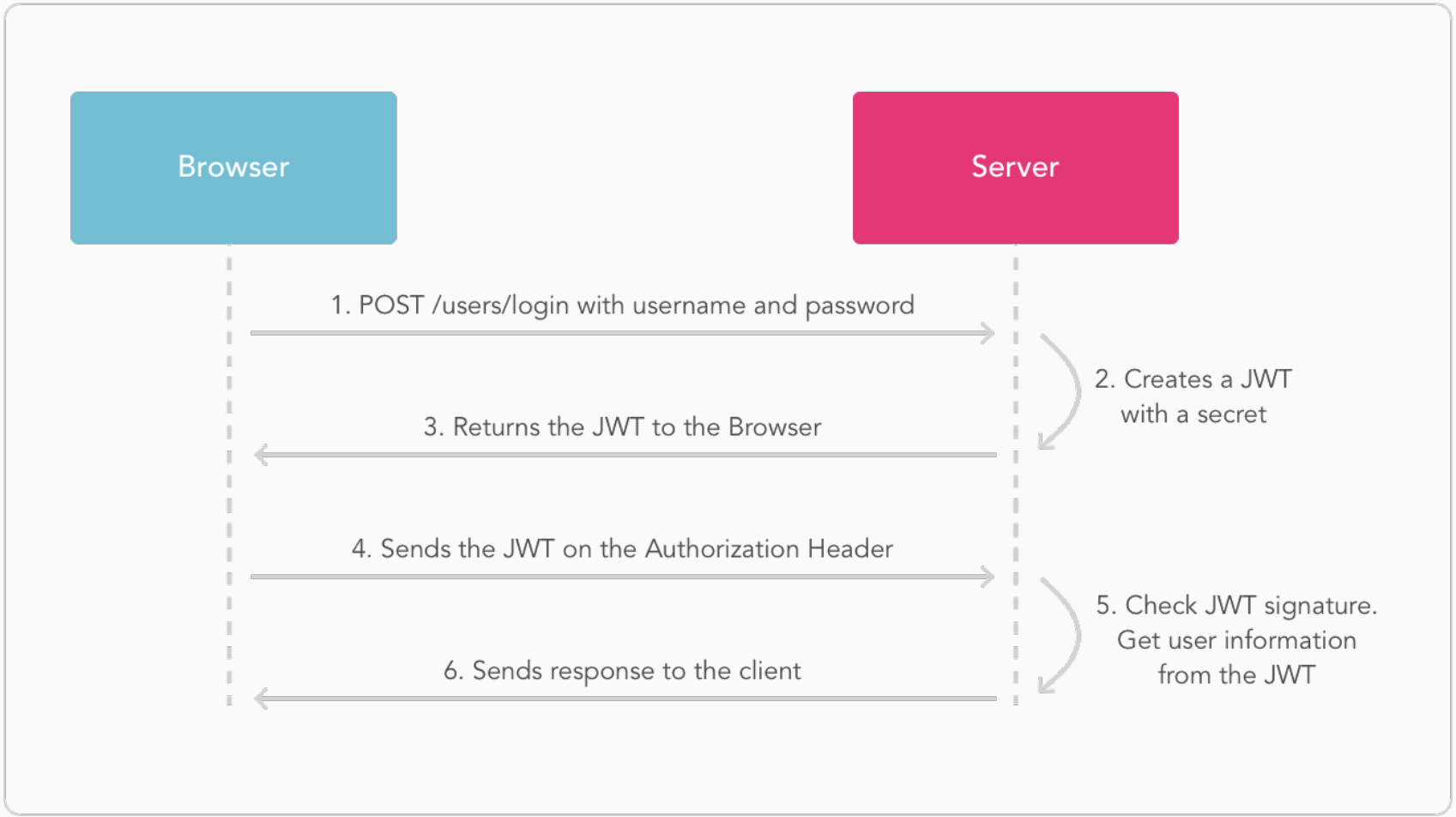
Bezsesyjność

Co się dzieje, jeżeli frontend stoi na jednym serwerze, a backend na drugim?

Bezsesyjność

Wówczas obie aplikacji muszą ze sobą rozmawiać w pewien sposób (REST?) poprzez protokół HTTP.

JWT



Rysunek 2: https://jwt.io/introduction/

Autoryzacja

Kto i co?

Autoryzacja jest mechanizmem, który odpowiada na pytanie *kto co może?*

Kto i co?

Najprostszy przykład:

- kto: zalogowany użytkownik ID $= 1$
- co: może wejść do edycji swojego profilu

ale:

- kto: zalogowany użytkownik ID $= 1$
- co: nie może wejść do edycji nieswojego profilu

oraz:

- kto: niezalogowany użytkownik
- co: nie może wejść do edycji żadnego profilu

Kto i co?

Wynik autoryzacji można uzależnić od wielu czynników:

- zablokowanie funkcjonalności dla użytkowników, którzy nie opłacili *wersji premium*,
- zablokowanie sklepu internetowego w niedzielę dla użytkowników z Polski,
- zablokowanie dostępu dla użytkowników forum z mniejszą niż n liczbą postów,
- i tak dalej...

ACL

Popularnym podejściem jest implementacja **ACL**, *access control list*, czyli lista kontroli dostępu.

ACL

Przykładowa lista kontroli dostępu może wyglądać następująco:

użytkownik	moduł	akcja	dostęp
jsmith	dashboard	get	true
jsmith	products	get	true
jsmith	products	post	false
jsmith	product	get	true
jsmith	product	patch	false
jsmith	product	delete	false

Autoryzacja

Gdzie można zastosować autoryzację?

- na poziomie routingu?
- na poziomie kontrolerów?
- na poziomie serwisów?

Inne sposoby autoryzacji

Istnieją inne sposoby autoryzacji:

- ręczna autoryzacja
- system ról
- system pozwoleń
- system bram i polityk

Część z nich występuje pojedynczo, część można ze sobą łączyć. Jakie jest najlepsze podejście? To, które pasuje do implementowanego systemu.

Podsumowanie

Bibliografia i ciekawe źródła

 https://jwt.io/introduction/

Pytania?

Kod prezentacji dostępny jest w repozytorium git pod adresem
`https://bitbucket.org/krewak/pwsz-ppsi`

Wszystkie informacje dot. kursu dostępne są pod adresem
`http://pwsz.rewak.pl/kursy/4`