# **Sprawozdanie 4**

# **AJAX – Rafał Bysiek**

Celem zadania jest implementacja części klienckiej aplikacji do zarządzania biblioteką.

Aplikacja posiada funkcjonalność obsługi po stronie przeglądarki internetowej wszystkich endpointów utworzonych poprzednio w Springu.

Utworzono odpowiedni interfejs graficzny oraz podstrony umożliwiające obsługę użytkownika oraz zarządzanie biblioteką.

Aplikacja została utworzona w IntelliJ IDEA, oraz uruchamiana jest przy pomocy Tomcat Server w wersji 9.   
Obsługa requestów - aplikacji odbywa się przy pomocy programu Postman.

Do testowania działania części backendowej użyto XDEBUG dodając do request ?XDEBUG\_SESSION\_START=1

# **Struktura projektu**

# 

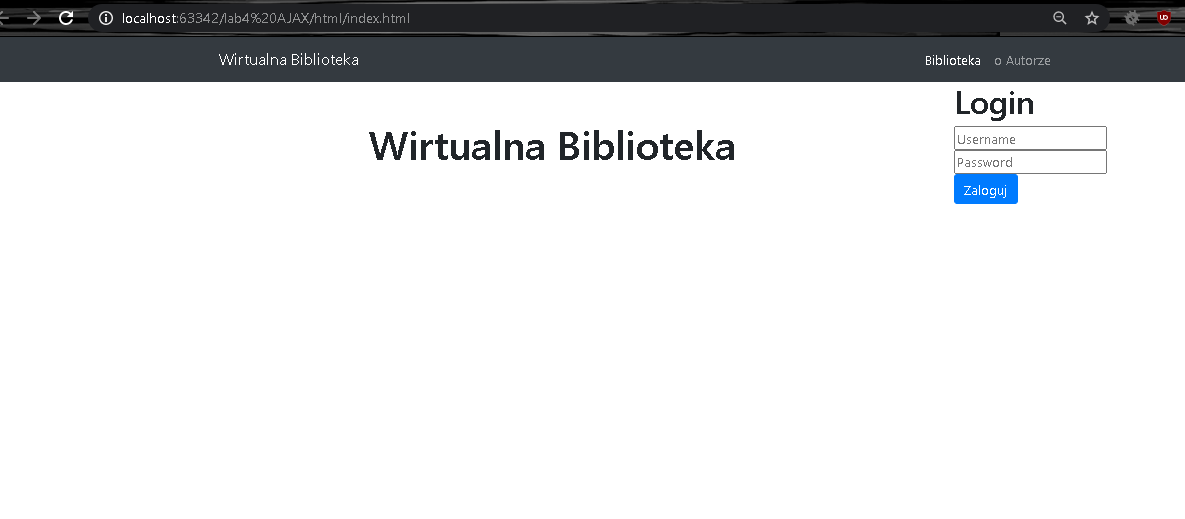
Projekt podzielono na 3 sekcje: html, js, css.

# **Logowanie**

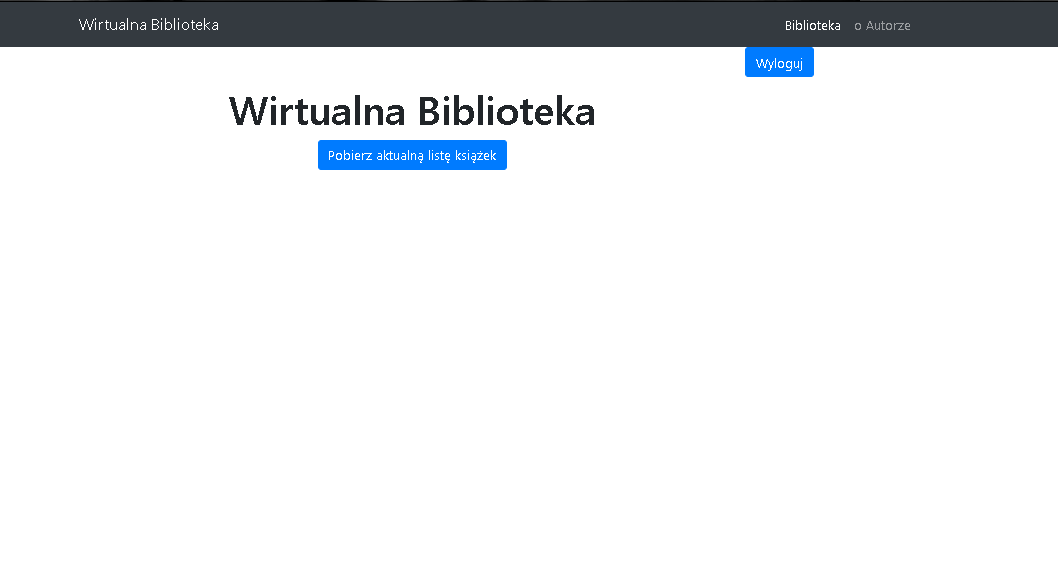
Po uruchomieniu aplikacji wita nas napis Wirtualna Biblioteka oraz formularz logowania.   
Warto w tym miejscu zwrócić uwagę na szary pasek z nawigacją.

Znajduje się tam odnośnik przenoszący na główną stronę (tzw. Homepage) – z lewej

Oraz dwa przyciski nawigacyjne – z prawej.



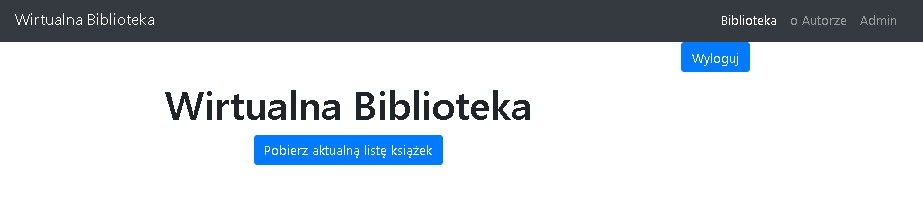
Formularz logowania odwołuje się do części backendowej aplikacji gdzie dokonuje autoryzacji i jeśli zwrócono status 200 użytkownikowi prezentowana jest strona główna aplikacji.



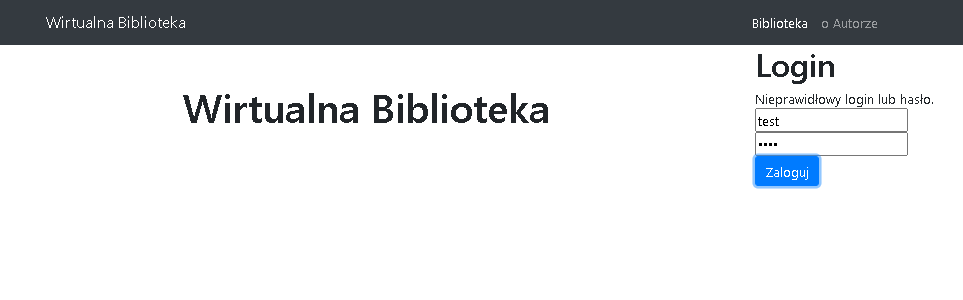
Po zalogowaniu znika również formularz logowania, który zatąpiony zostaje przyciskiem wyloguj.

W tym miejscu użytkownik może pobrać listę książek lub wylogować się.

Jeśli zalogujemy się jako administrator zobaczymy dodatkowo w pasku nawigacyjnym odnośnik do strony przeznaczonej dla administratora.



Podanie błędnych danych lub problem autoryzacją zostanie opatrzony komunikatem.



http\_request = new ***XMLHttpRequest***();  
http\_request.onload = function (xhr) {  
 if (xhr.target.status == 200) {  
 ...  
 } else {  
 if (xhr.target.status == 401) {  
 ***document***.getElementById("message\_login").innerHTML = "Nieprawidłowy login lub hasło."  
 } else {  
 ***document***.getElementById("message\_login").innerHTML = "Wystąpił problem przy próbie logowania. Skontaktuj się z administratorem"  
 }  
 }  
};

**Logowanie Spring**

By umożliwić ewentualne dodatkowe akcje podczas logowania / wylogowania użytkownika, dodano endpointy w części backendowej.

Wykjorzystanu tu defaultowe zwracanie exception (401 przy podaniu nieprawidłowych danych)

Jeśli wszystko przebiegnie pomyślnie zwracany jest status 200 oraz rola użytkownika którego dane zostały wprowadzone.

@SpringBootApplication  
@RestController  
public class LoginController {  
  
 @CrossOrigin  
 @PostMapping("/login")  
 public String doPost(  
 @RequestParam("username") String username,  
 @RequestParam("password") String password,  
 HttpServletRequest request  
 ) throws Exception {  
  
 request.logout();  
 request.login(username,password);  
 Authentication authentication = SecurityContextHolder.*getContext*().getAuthentication();  
  
 return authentication.getAuthorities().toString();  
  
 }  
  
 @CrossOrigin  
 @GetMapping("/logout")  
 public void logout(HttpServletRequest request) throws Exception {  
 try {  
 request.logout();  
 } catch (Exception e) {  
 throw new Exception();  
 }  
 }  
}

**Obsługa autoryzacji w aplikacji**

Requesty wysyłane do backendu dotyczące zasobów wymagających specjalnej autoryzacji jak dodanie czy usunięcie książki opatrzone są dodatko autoryzacją Basic Auth.

By usprawnić działanie, dane te przechowywane są w ciasteczkach.

Do sprawnego pobierania wartości, które nas interesują dodano metodę zwracającą wartość ciasteczka.

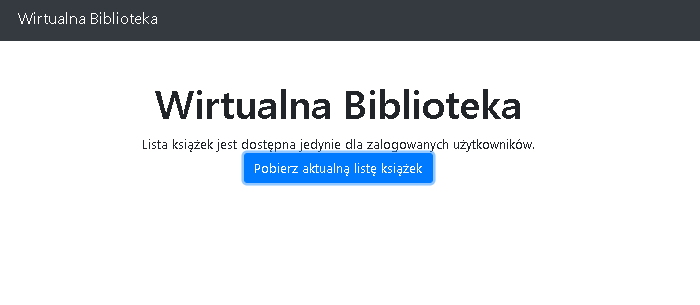
function getCookie(name) {  
 name = name + "=";  
 var ca = ***document***.cookie.split(';');  
 for(var i = 0; i < ca.length; i++) {  
 var c = ca[i];  
 while (c.charAt(0) == ' ') {  
 c = c.substring(1);  
 }  
 if (c.indexOf(name) == 0) {  
 return c.substring(name.length, c.length);  
 }  
 }  
 return "";  
}

Jeśli użytkownik jakimś sposobem spróbuje uruchomić akcję, która jest dla niego niedostępna lub wystąpi błąd, klient zostanie o tym poinformowany odpowiednim komunikatem.

http\_request = new ***XMLHttpRequest***();  
http\_request.onload = function(xhr) {  
 if (xhr.target.status == 200) {

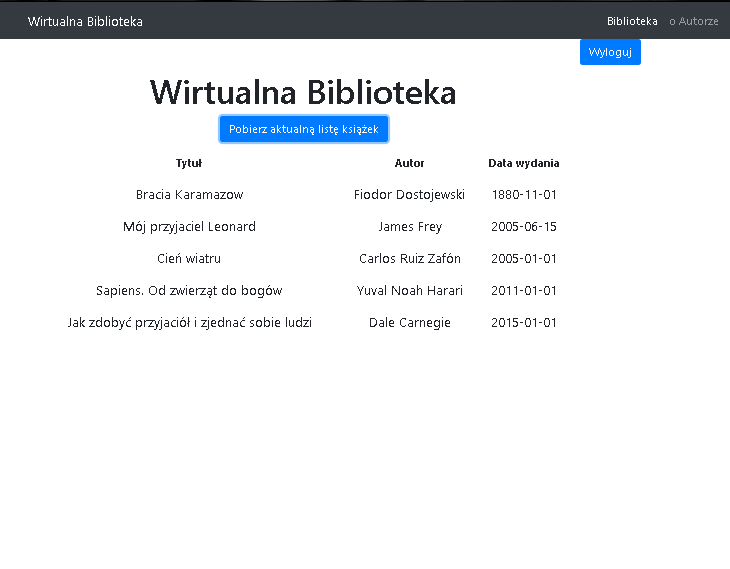
...

} else {  
 if (xhr.target.status == 401) {  
 ***document***.getElementById("message\_dashboard").innerHTML = "Lista książek jest dostępna jedynie dla zalogowanych użytkowników."  
 } else {  
 ***document***.getElementById("message\_dashboard").innerHTML = "Wystąpił problem przy próbie pobrania listy książek. Proszę skontaktować się z administratorem."  
 }  
}



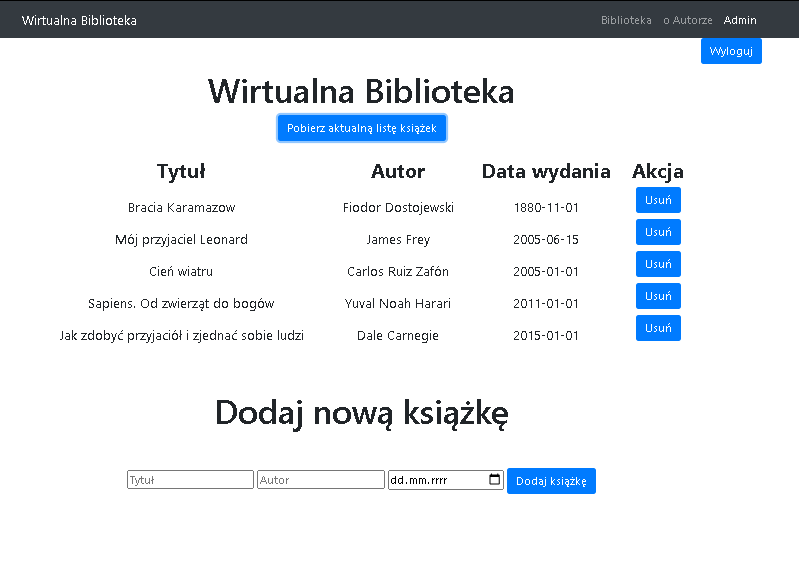
# **Lista książek użytkownik**

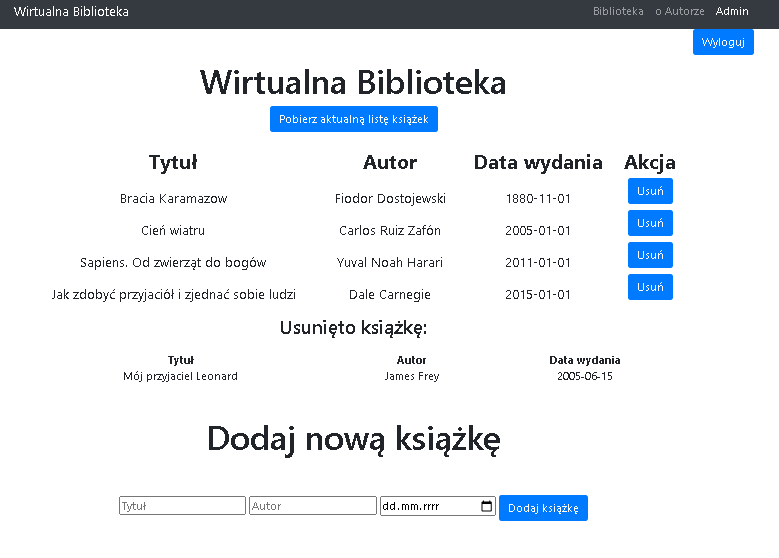
Klikając przycisk Pobierz aktualną listę książek.



# **Lista książek administrator**

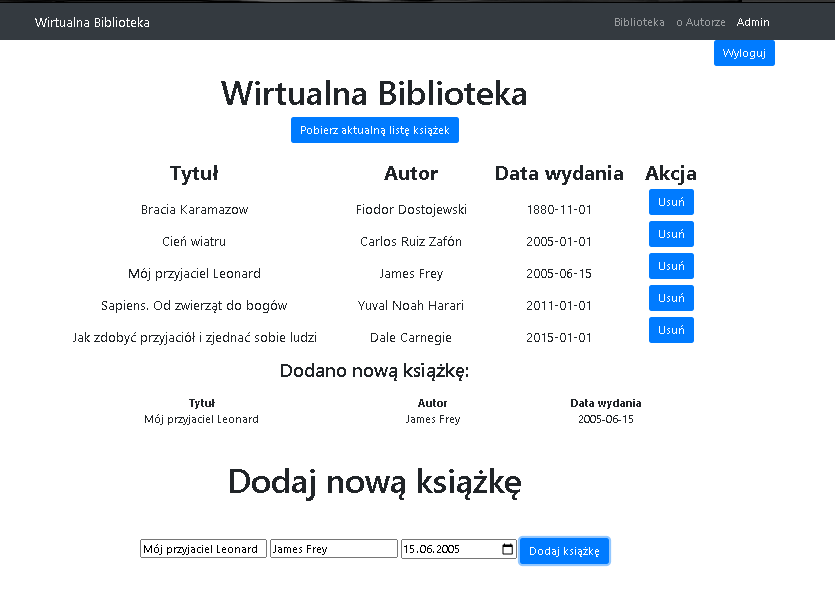
Zakładka Admin daje nam dodatkowo, poza pobraniem listy książek, możliwość dodania i usunięcia książki.

Usunięcie książki: W każdym wierszu znajduje się przycisk **Usuń** którym możemy usunąć daną książkę z listy. Lista książek zostanie automatycznie odświeżona oraz pojawi się komunikat potwierdzający przeprowadzoną operację.



Dodanie książki: Należy wypełnić formularz oraz kliknąć **Dodaj książkę**.

Książka zostanie dodana do listy, lista zostanie automatycznie odświeżona, oraz pojawi się komunikat potwierdzający przeprowadzoną operację.



Lista książek jest często pobierana a lista często odświeżana dlatego akcja pobrania i odświeżenia listy została wydzielona do osobnej metody a nawet osobnego pliku.

function loadFromApi(isAdmin) {  
 var url = "http://localhost:8080/dashboard";  
 if (getCookie('role') === "ROLE\_USER" || getCookie('role') === "ROLE\_ADMIN")  
 {  
 http\_request = new ***XMLHttpRequest***();  
 http\_request.onload = function(xhr) {  
 if (xhr.target.status == 200) {  
 var data = ***JSON***.parse(xhr.target.response);  
 if (isAdmin && getCookie('role') === "ROLE\_ADMIN") {  
 ***document***.getElementById("table\_books").innerHTML =  
 "<tr>" + "<th class='book\_row'>" + "Tytuł" + "</th>" + "<th class='book\_row'>" + "Autor" + "</th>" + "<th class='book\_row'>" + "Data wydania" + "</th>"+ "<th class='book\_row'>" + "Akcja" + "</th>" + "</tr>" +  
 data.books.map(  
 function(val) {  
 return "<tr>" + "<td class='book\_row'>" + val.title + "</td>" + "<td class='book\_row'>" + val.author + "</td>" + "<td class='book\_row'>" + val.releaseDate + "</td>"+ "<td><button type=\"button\" class=\"btn btn-primary\" onClick={RemoveBook(" + val.id + ")}>Usuń</button></td>" + "</tr>";  
 }).join('')  
 } else {  
 ***document***.getElementById("table\_books").innerHTML =  
 "<tr>" + "<th>" + "Tytuł" + "</th>" + "<th>" + "Autor" + "</th>" + "<th>" + "Data wydania" + "</th>" + "</tr>" +  
 data.books.map(  
 function(val) {  
 return "<tr>" + "<td class='book\_row'>" + val.title + "</td>" + "<td class='book\_row'>" + val.author + "</td>" + "<td class='book\_row'>" + val.releaseDate + "</td>" + "</tr>";  
 }).join('')  
 }  
 } else {  
 if (xhr.target.status == 401) {  
 ***document***.getElementById("message\_dashboard").innerHTML = "Lista książek jest dostępna jedynie dla zalogowanych użytkowników."  
 } else {  
 ***document***.getElementById("message\_dashboard").innerHTML = "Wystąpił problem przy próbie pobrania listy książek. Proszę skontaktować się z administratorem."  
 }  
 }  
 };  
 http\_request.open('GET', url, true);  
 http\_request.setRequestHeader("Authorization", "Basic " + btoa(getCookie("username") + ":" + getCookie("password")));  
 http\_request.send(null);  
 } else {  
 ***document***.getElementById("message\_dashboard").innerHTML = "Lista książek jest dostępna jedynie dla zalogowanych użytkowników."  
 }  
}

Wpłynęło to bardzo pozytywnie na przejrzystość kodu.

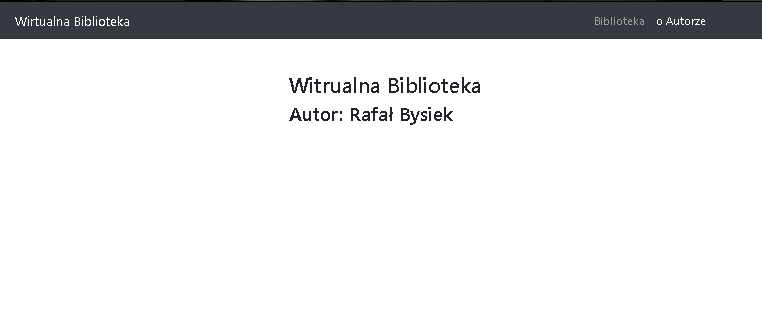
Wydzielono również metody dodania i usunięcia książek, gdyż używane są jedynie przez administratora i nie ma potrzeby ładowania tego pliku dla użytkowników innych niż administrator.

Flaga isAdmin jest przekazywana właśnie na stronie przeznaczonej dla administratora, by poinformować że chodzi o załadowanie rozbudowanej wersji tabelki z opcją usuwania.

Dodatkowo sprawdzana jest rola aktualnie zalogowanego użytkownika.

# Autor

Dodatkowo dodano podstronę z informacją o autorze aplikacji, gdzie można zamieścić dodatkowe informacje, kontakt, numer wersji itp.



Style przerzucono do pliku main.css

.book\_table {  
 width:100%;  
 margin-top:20px;  
}  
  
th.book\_row {  
 font-size:30px;  
}  
  
td.book\_row {  
 padding-top:20px;  
 font-size:20px;  
}  
  
.header\_title {  
 font-size:50px;  
}  
  
.middle\_action\_header {  
 padding:50px;  
}  
  
.new-book-form {  
 display:block  
}  
.author\_info {  
 margin-left: auto;  
 margin-right: auto;  
 display: table;  
 padding: 50px  
}