

AKADEMIA NAUK STOSOWANYCH W NOWYM SĄCZU

Wydział Nauk Inżynieryjnych
Informatyka Stosowana

DOKUMENTACJA PROJEKTOWA INŻYNIERIA OPROGRAMOWANIA

Księgarnia

Autorzy:
Dawid Barnaś
Kacper Górka
Paweł Cetnarowski

Prowadzący:
mgr inż. Daniel Drozd

Nowy Sącz 2022

Spis treści

1. Opis projektu	3
2. Identyfikacja podobnych istniejących projektów. W czym dane rozwiązanie będzie lepsze od konkurencji?	4
3. Definicja wymagań funkcjonalnych i нефункциональных	5
3.1. Wymagania funkcjonalne	5
3.2. Wymagania нефункциональные:	5
4. Identyfikacja problemów oraz proponowane rozwiązania	6
5. Dobór technologii	7
6. Diagram przypadków użycia	8
7. Scenariusze użycia	9
8. Diagram ERD	13
9. Diagram klas	14
10. Diagram sekwencji	15
11. Diagram aktywności - Logowanie	16
12. Wzorce projektowe - opis	17
13. Implementacja	21
14. Uruchomienie oraz działanie aplikacji	23
15. Testowanie	30
16. Podsumowanie projektu i wnioski	32
17. Bibliografia	33

1. Opis projektu

Celem projektu z Inżynierii Oprogramowania jest utworzenie strony internetowej księgarni, przy użyciu której użytkownik będzie mógł założyć swoje konto w celu korzystania z serwisu. Nie będzie to punkt obowiązkowy, ponieważ będzie można korzystać ze strony również jako gość. Księgarnia będzie oferowała usługi sprzedaży książek. Na stronie będzie wyodrębniony zbiór książek podzielony na kategorię, autorów, datę powstania itp. Będzie ukazany stan danej książki, czy jest ona dostępna bądź nie, ewentualnie czas oczekiwania na dany tytuł.

2. Identyfikacja podobnych istniejących projektów.

W czym dane rozwiązanie będzie lepsze od konkurencji?

Wiele internetowych księgarni ma pomieszany zbiór oferowanych produktów - można tam znaleźć gry, zabawki, filmy, sponsorowane produkty itp. Nasza księgarnia będzie oferowała tylko książki, więc użytkownik nie będzie musiał przebijać się przez ogromną ilość informacji, które go nie interesują, lecz od razu do celu swojego wyszukiwania. Dodatkowo, jeśli dana książka będzie przed terminem swojej premiery, będzie można ją nabyć w przedsprzedaży z automatyczną wysyłką oraz dostawą właśnie w ten dzień.

3. Definicja wymagań funkcjonalnych i нефункциональных

3.1. Wymagania funkcjonalne

Administrator

- Może dodawać i usuwać użytkowników
- Może sprawdzić historię zakupową użytkownika
- Może dodać nową książkę
- Może sprawdzić dane użytkownika
- Może się zalogować oraz wylogować
- Może wyszukiwać książkę po tytule, autorze lub kategorii

Użytkownik

- Może utworzyć konto
- Może się zalogować oraz wylogować
- Może sprawdzić swoją historię zakupionych książek
- Może dodawać oraz usuwać produkty z koszyka
- Może wyszukiwać książkę po tytule, autorze lub kategorii
- Może kupić książkę

3.2. Wymagania нефункциональные:

- Przyjemny interfejs
- Przy książce będzie znajdować się ilustracja okładki
- Użytkownik będzie mieć mniejszy interfejs od administratora

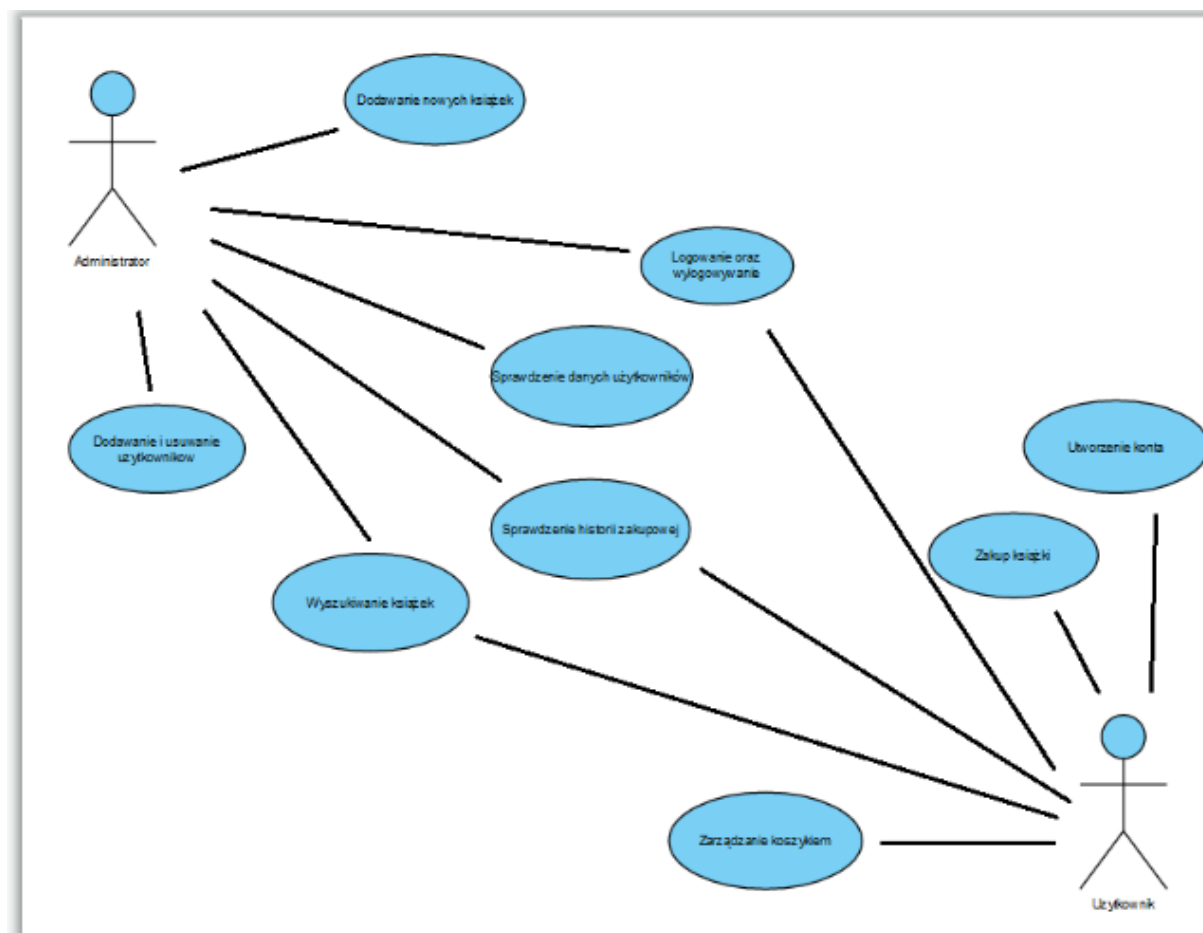
4. Identyfikacja problemów oraz proponowane rozwiązania

Ze względu na małą ilość czasu, nasz zespół nie zdołał wykonać zaplanowanego wcześniej koszyka.

5. Dobór technologii

- **Laravel** - to stworzony przez Taylora Otwell'a w 2011 roku framework PHP, który działa zgodnie ze strukturą MVC (Model View Controller). Jego sporą zaletą jest licencja Open Source, o którą framework jest oparty. Zapewnia to stały rozwój, dobre wsparcie oraz wszechstronność zastosowania. Dodatkowo poprzez dokumentację prowadzoną na wysokim poziomie jego nauka nie sprawia trudności, a próg wejścia do grona programistów Laravela jest stosunkowo niski.
- **PHP** - interpretowany, skryptowy język programowania zaprojektowany do generowania stron internetowych i budowania aplikacji webowych w czasie rzeczywistym. PHP jest najczęściej stosowany do tworzenia skryptów po stronie serwera WWW, ale może być on również używany do przetwarzania danych z poziomu wiersza poleceń, a nawet do pisania programów pracujących w trybie graficznym
- **HTML** - pozwala opisać strukturę informacji zawartych wewnątrz strony internetowej, nadając odpowiednie znaczenie semantyczne poszczególnym fragmentom tekstu – formując hiperłącza, akapity, nagłówki, listy – oraz osadza w tekście dokumentu obiekty plikowe, na przykład multimedia, lub elementy baz danych, na przykład interaktywne formularze danych.
- **CSS** - to język arkuszy stylów używany do opisywania prezentacji dokumentu napisanego w języku znaczników, takim jak HTML. CSS jest podstawową technologią World Wide Web, obok HTML i JavaScript.
- **JavaScript** - w skrócie JS – skryptowy oraz wieloparadygmatowy język programowania, stworzony przez firmę Netscape, najczęściej stosowany na stronach internetowych.
- **Bootstrap** - biblioteka CSS, rozwijana przez programistów Twittera, wydawany na licencji MIT. Zawiera zestaw przydatnych narzędzi ułatwiających tworzenie interfejsu graficznego stron oraz aplikacji internetowych. Bazuje głównie na gotowych rozwiązaniach HTML oraz CSS i może być stosowany m.in. do stylizacji takich elementów jak teksty, formularze, przyciski, wykresy, nawigacje i innych komponentów wyświetlanych na stronie. Biblioteka korzysta także z języka JavaScript.

6. Diagram przypadków użycia



7. Scenariusze użycia

Nazwa	Scenariusz 1 – Logowanie
Aktor	Administrator
Warunki początkowe	Administrator znajduje się w panelu logowania do witryny
Opis	Administrator loguje się do systemu
Ścieżki główne	<ol style="list-style-type: none"> 1. Administrator wprowadza swoje dane logowania i zatwierdza przyciskiem "Zaloguj" 2. Wynikiem pozytywnego przebiegu autoryzacji, jest przekierowanie administratora do głównego panelu zarządzania systemem
Ścieżka alternatywna	2a. Administrator podaje błędne dane logowania, możliwość ponownego wprowadzenia danych
Warunki końcowe	Administrator ma dostęp do zarządzania systemem

Nazwa	Scenariusz 2 - Rejestracja
Aktor	Użytkownik
Warunki początkowe	Użytkownik przechodzi do opcji rejestracji
Opis	Użytkownik wprowadza dane potrzebne do rejestracji konta
Ścieżki główne	<ol style="list-style-type: none"> 1. Użytkownik wprowadza login oraz hasło swojego konta 2. Konto zostaje założone
Ścieżka alternatywna	<ol style="list-style-type: none"> 2. a) Wpisane hasło nie spełnia wymagań, konto nie zostaje założone 2. b) Wpisany login jest już zajęty, konto nie zostaje założone
Warunki końcowe	Użytkownik ma dostęp do logowania

Nazwa	Scenariusz 3 - Logowanie
Aktor	Użytkownik
Warunki początkowe	Użytkownik znajduje się w panelu logowania do witryny
Opis	Użytkownik loguje się do systemu
Ścieżki główne	<ol style="list-style-type: none"> 1. Użytkownik wprowadza swoje dane logowania i zatwierdza przyciskiem "Zaloguj" 2. Wynikiem pozytywnego przebiegu autoryzacji, jest przekierowanie użytkownika do strony głównej systemu
Ścieżka alternatywna	2a. Użytkownik podaje błędne dane logowania, możliwość ponownego wprowadzenia danych
Warunki końcowe	Użytkownik ma dostęp do serwisu

Nazwa	Scenariusz 4 – Dodanie nowych książek
Aktor	Administrator
Warunki początkowe	Administrator jest zalogowany
Opis	Administrator dodaje nową książkę do systemu
Ścieżki główne	<ol style="list-style-type: none"> 1. Administrator przechodzi do panelu dodawania nowej książki 2. Aktor wpisuje dane dodawanej książki 3. Zatwierdza wpisane dane przyciskiem "Dodaj". 4. Książka zostaje dodana do systemu
Ścieżka alternatywna	4a. Książka już znajduje się w systemie więc nie zostaje dodana
Warunki końcowe	Dodana książka znajduje się w liście książek.

Nazwa	Scenariusz 5 – Dodanie nowych użytkowników
Aktor	Administrator
Warunki początkowe	Administrator jest zalogowany
Opis	Administrator dodaje nowego użytkownika do systemu
Ścieżki główne	<ol style="list-style-type: none"> 1. Administrator przechodzi do panelu dodawania nowego użytkownika. 2. Administrator wpisuje dane nowego użytkownika 3. Administrator zatwierdza wpisane dane przyciskiem "Dodaj". 4. Użytkownik zostaje dodany do systemu
Ścieżka alternatywna	4a) Podany login jest zajęty - konto nie zostaje utworzone 4b) Podane hasło nie spełnia wymagań - konto nie zostaje utworzone
Warunki końcowe	Konto użytkownika zostaje pomyślnie utworzone.

Nazwa	Scenariusz 6 – Zakup książki
Aktor	Użytkownik
Warunki początkowe	Użytkownik jest zalogowany
Opis	Użytkownik dokonuje zakupu książki
Ścieżki główne	<ol style="list-style-type: none"> 1. Użytkownik przechodzi do zakładki sklep 2. Użytkownik wyszukuje książkę wpisując nazwę, lub przegląda według kategorii 3. Użytkownik zatwierdza wybór książki oraz ilość 4. Użytkownik dokonuje zakupu książki za posiadane punkty
Ścieżka alternatywna	3a. W wypadku braku książki w magazynie, zostaje wyświetlony monit o chwilowym braku dostępności
Warunki końcowe	Książka zostaje dodana do historii zakupowej użytkownika

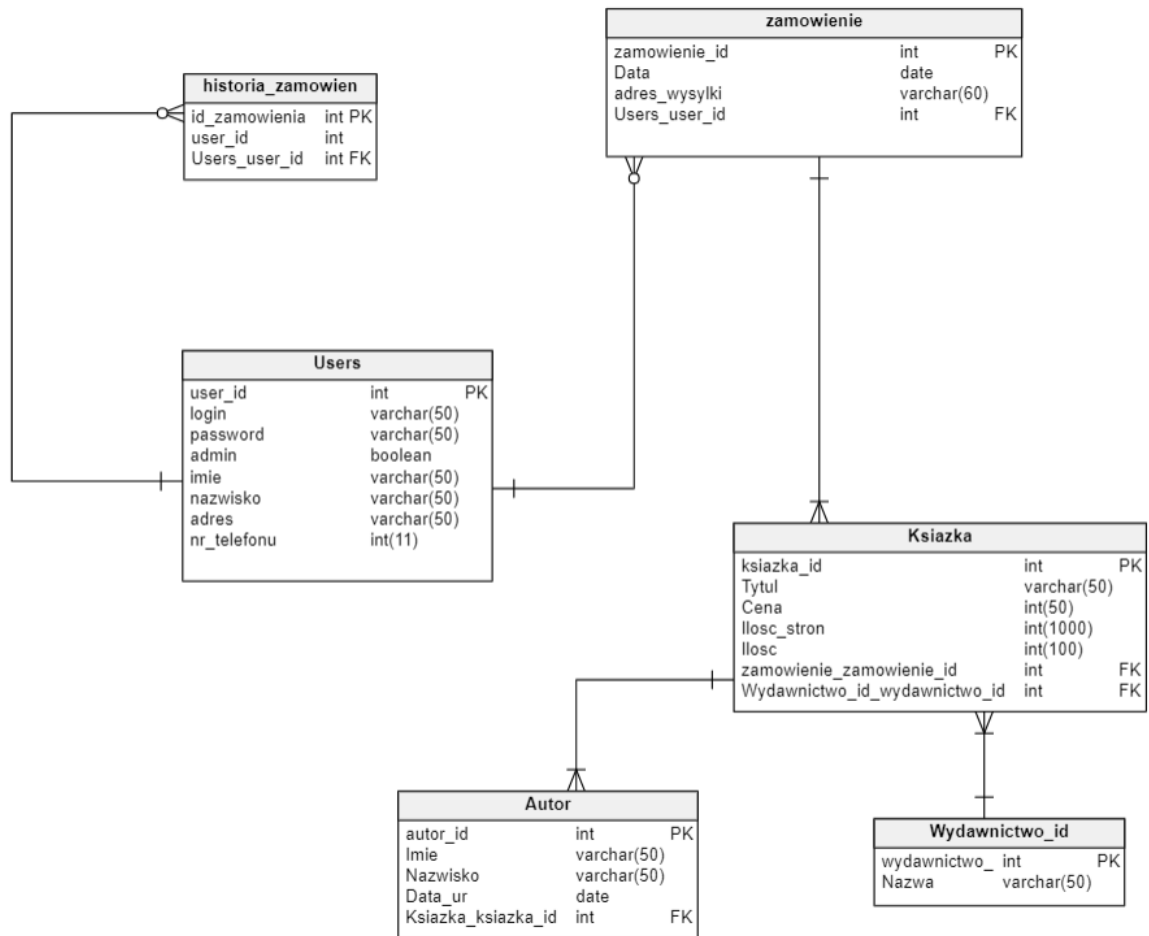
Nazwa	Scenariusz 7 – Usuwanie użytkownika
Aktor	Administrator
Warunki początkowe	Administrator jest zalogowany
Opis	Administrator usuwa konto użytkownika będące w systemie
Ścieżki główne	<ol style="list-style-type: none"> 1. Administrator przechodzi do panelu usuwania użytkownika 2. Administrator zaznacza przycisk "X" przy użytkowniku 3. Konto użytkownika zostaje usunięte z systemu.
Ścieżka alternatywna	
Warunki końcowe	Konto użytkownika zostaje usunięte z systemu.

Nazwa	Scenariusz 8 – Sprawdzenie danych użytkownika
Aktor	Administrator
Warunki początkowe	Administrator jest zalogowany
Opis	Administrator sprawdza dane użytkownika
Ścieżki główne	<ol style="list-style-type: none"> 1. Aktor przechodzi do listy użytkowników 2. W okienku wyszukiwania wpisuje imię bądź nazwisko użytkownika. 3. Lista wypisuje pasujące osoby do wyszukiwania wraz z danymi
Ścieżka alternatywna	3a. Brak podanej osoby w systemie czego skutkiem jest pusta lista.
Warunki końcowe	Administrator jest w stanie odczytać dane użytkownika z listy użytkowników.

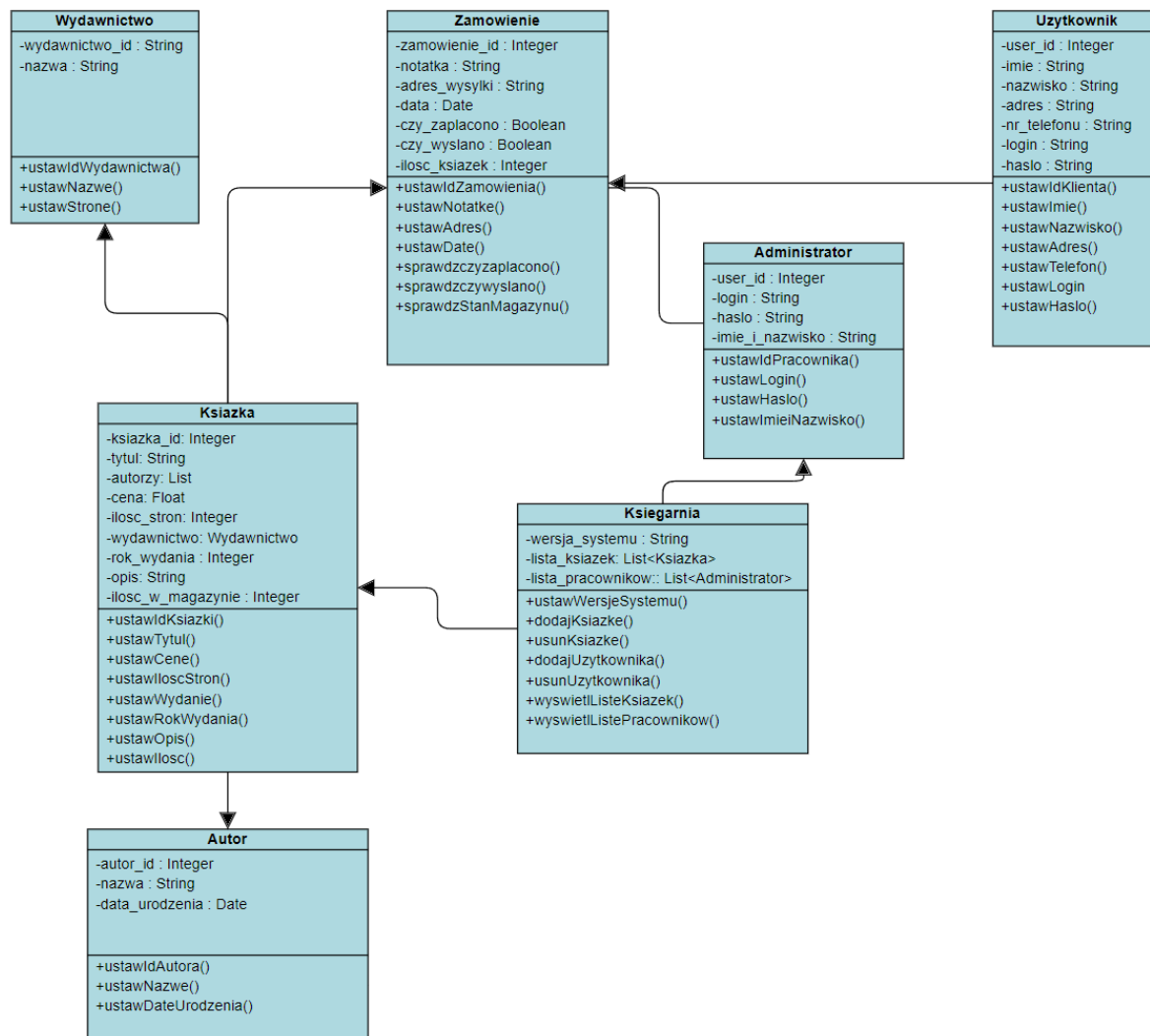
Nazwa	Scenariusz 9 – Wylogowanie
Aktor	Administrator, Użytkownik
Warunki początkowe	Aktor jest zalogowany na swoje konto
Opis	Aktor zostaje wylogowany z bieżącej sesji
Ścieżki główne	<ol style="list-style-type: none"> 1. Aktor klika w przycisk "Wyloguj" 2. Aktor zostaje wylogowany z systemu.
Ścieżka alternatywna	
Warunki końcowe	Aktor zostaje wylogowany z systemu

Nazwa	Scenariusz 10 – Wyszukiwanie książek
Aktor	Administrator, Użytkownik
Warunki początkowe	Aktor jest zalogowany na swoje konto
Opis	Aktor wyszukuje książkę
Ścieżki główne	<ol style="list-style-type: none"> 1. Aktor przechodzi do zakładki wszystkich tytułów 2. Aktor wpisuje tytuł książki, autora bądź kategorię 3. Książka zostaje prawidłowo wyszukana oraz wyświetlona na ekranie
Ścieżka alternatywna	3a) Wpisany tytuł, autor bądź kategoria nie istnieje, nie zostaje wyświetlony żaden tytuł
Warunki końcowe	Książka zostaje wyszukana

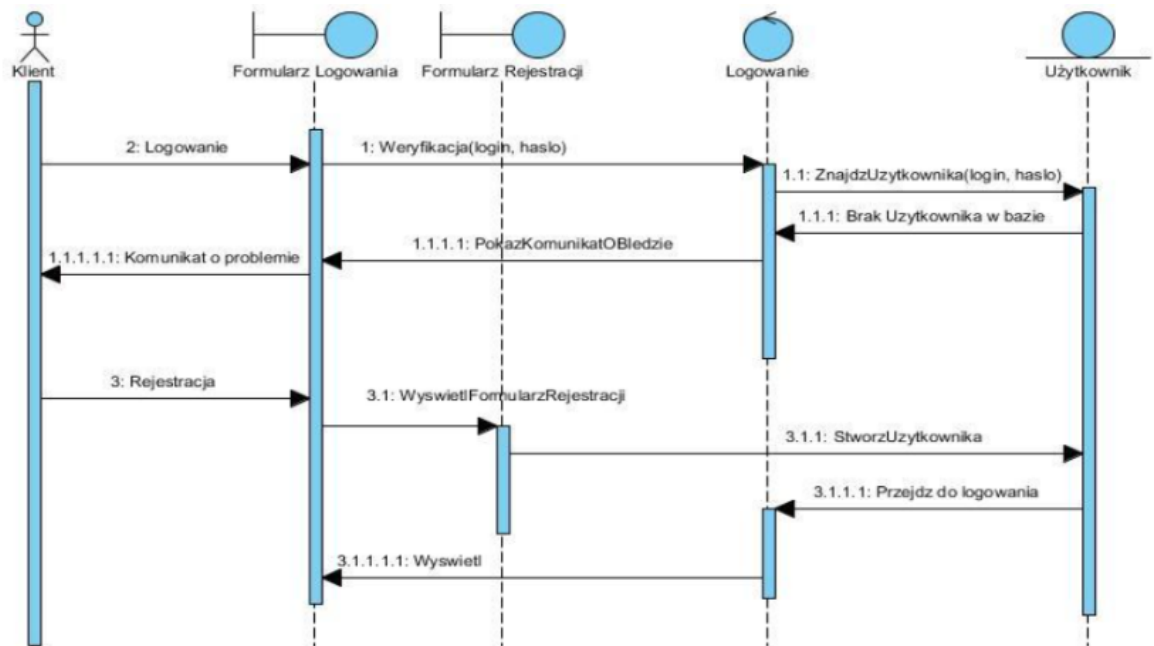
8. Diagram ERD



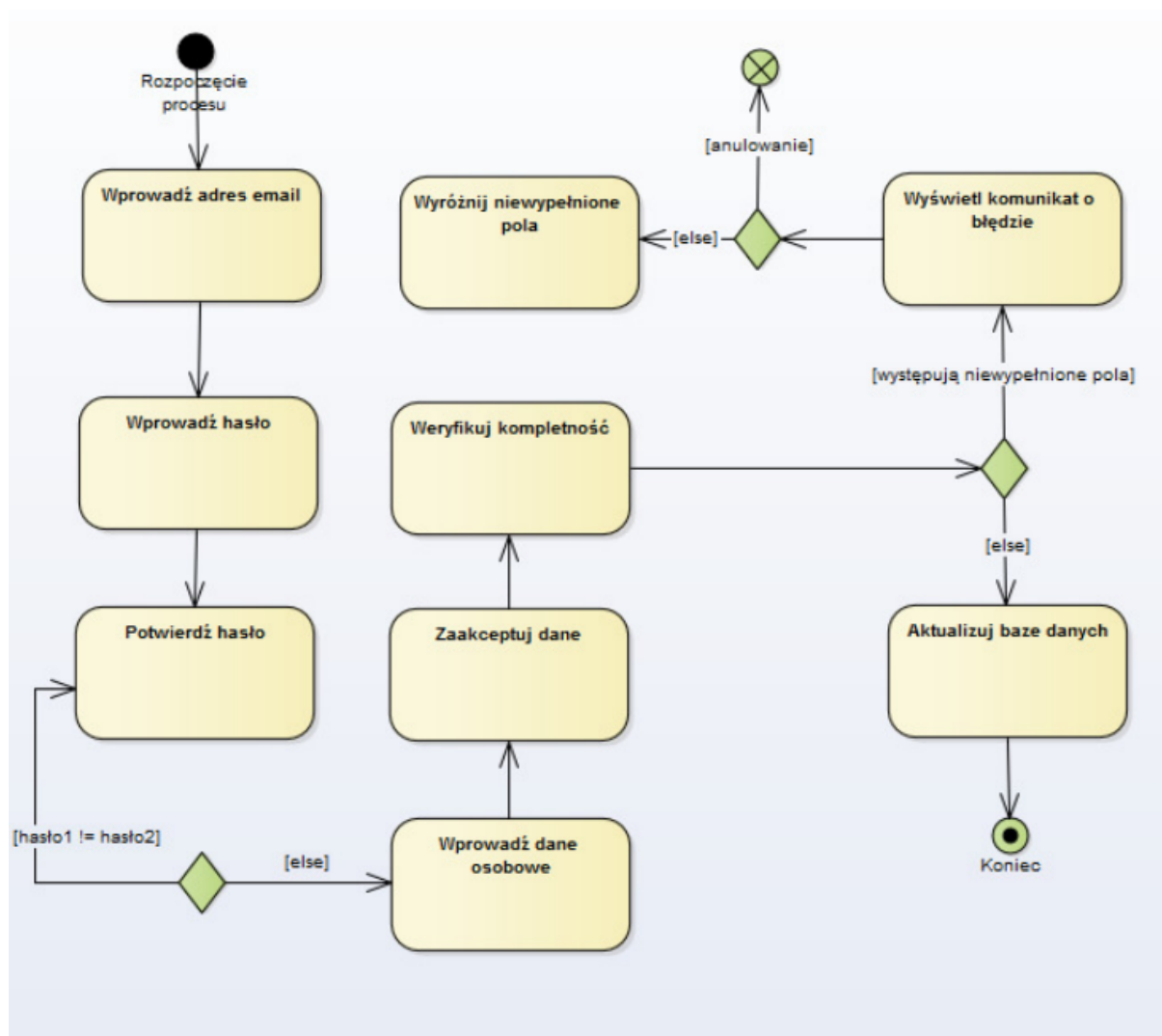
9. Diagram klas



10. Diagram sekwencji



11. Diagram aktywności - Logowanie



12. Wzorce projektowe - opis

- **Metoda szablonowa** – inaczej wzorzec behawioralny. Zadaniem metody szablonowej jest definicja szkieletu algorytmu, w którym istnieje możliwość inicjowania zmian w poszczególnych częściach szkieletu. Szablon jest zbudowany z niezmiennych operacji, które są zdefiniowane w klasie bazowej oraz z metod zmiennych, które przyjmują implementacje zależną od klas pochodnych. Dzięki temu zmniejsza się nadmiarowość kodu, a dokonanie w nim zmian staje się mniej kosztowne. Można powiedzieć, że metoda szablonowa dostarcza pewien standard realizacji tego samego zadania na różne sposoby.
- **Budowniczy** - używany w programowaniu obiektowym. Zalicza się on do rodziny wzorców konstrukcyjnych. Dzięki użyciu budowniczego oddzielamy proces tworzenia obiektu od jego reprezentacji. Jest to dość prosty wzorzec, który jednak sprawia problemy, ze względu na różne warianty w jakich występuje.
- **Adapter** - strukturalny wzorzec projektowy, którego celem jest umożliwienie współpracy dwóm klasom o niekompatybilnych interfejsach. Adapter przekształca interfejs jednej z klas na interfejs drugiej klasy. Innym zadaniem omawianego wzorca jest opakowanie istniejącego interfejsu w nowy. Wzorzec adaptera stosowany jest najczęściej w przypadku, gdy wykorzystanie istniejącej klasy jest niemożliwe ze względu na jej niekompatybilny interfejs. Drugim powodem użycia może być chęć stworzenia klasy, która będzie współpracowała z klasami o nieokreślonych interfejsach
- **Singleton** – jest szczególnie przydatny gdy chcemy zapewnić kontrolę dostępu do dzielonego zasobu takiego jak baza danych lub plik. Dzięki temu wzorcowi projektowemu, istnieje możliwość przydzielenia klasie zadania śledzenia swojego jedyne go egzemplarza, a także zagwarantowania i zapewnienia, że nie powstanie żaden inny jej egzemplarz. Wzorzec singleton jest szczególnie przydatny gdy musi istnieć dokładnie jeden egzemplarz(obiekt) klasy dostępny klientom w znanym miejscu, a także kiedy potrzebna jest możliwość rozszerzania jedyne go egzemplarza przez tworzenie podklas, a klient może korzystać z ulepszego egzemplarza bez konieczności wprowadzania zmian w ich kodzie.

- **Odwiedzający** - Wzorzec projektowy Odwiedzający proponuje umieszczenie nowych obowiązków w osobnej klasie zwanej odwiedzającym, zamiast próbować zintegrować je z istniejącymi klasami. Pierwotny obiekt, który miał wykonywać te obowiązki, teraz jest przekazywany do jednej z metod odwiedzającego w charakterze argumentu. Daje to metodzie dostęp do wszystkich potrzebnych danych znajdujących się w obiekcie.

Zastosowanie wzorca Odwiedzający można odnaleźć, gdy zaistnieje potrzeba wykonywania jakiegoś działania na wszystkich elementach złożonej struktury obiektów (np.. Drzewo obiektów). Ponadto stosowanie odwiedzającego pozwala uprzętnąć logikę biznesową

czynności pomocniczych. Warto stosować ten wzorzec gdy jakieś zachowanie ma sens tylko w kontekście niektórych klas wchodzących w skład hierarchii klas, lecz nie wszystkich.

- **Dekorator** – jest to wzorzec projektowy, który “opakowuje” oryginalną klasę w nową klasę dekorującą. Oryginalny obiekt przekazany jest jako parametr konstruktora dekoratora. Metody z których zbudowany jest dekorator wywołują metody oryginalnego obiektu i dodatkowo implementują nową funkcjonalność. Dekorator jest alternatywą dla dziedziczenia – rozszerza zachowanie klasy w trakcie działania programu.

Wzorzec projektowy dekorator jest szczególnie przydatny, gdy zaistnieje potrzeba dynamicznego i przezroczystego sposobu dodawania pojedynczych obiektów, a także w wypadku zobowiązań, które mogą być cofnięte.

- **Fasada** - wzorzec projektowy należący do grupy wzorców strukturalnych. Służy do ujednolicenia dostępu do złożonego systemu poprzez wystawienie uproszczonego, uporządkowanego interfejsu programistycznego, który ułatwia jego użycie. Fasada to klasa stanowiąca prosty interfejs dla złożonego podsystemu, zawierającego mnóstwo ruchomych części. Fasada może dawać ograniczoną funkcjonalność, w porównaniu z korzystaniem z elementów podsystemu bezpośrednio, ale za to eksponuje tylko te możliwości, których klient naprawdę potrzebuje.

Stworzenie fasady jest wygodnym sposobem integracji twojej aplikacji ze skomplikowaną biblioteką posiadającą wiele funkcji, gdy potrzebujesz tylko wąskiego zakresu jej funkcji.

- **Kompozyt** - pozwala komponować obiekty w struktury drzewiaste, a następnie traktować te struktury jakby były osobnymi obiektami. Wzorzec ten stosuje się, gdy wygodniej jest korzystać z pewnych operacji dla danego obiektu w ten sam sposób jak dla grupy obiektów, np. rysując na ekranie prymitywy lub obiekty złożone z prymitywów; zmieniając rozmiar zarówno pojedynczych prymitywów jak i obiektów złożonych z prymitywów (z zachowaniem proporcji).
- **Obserwator** - Używany jest do powiadamiania zainteresowane obiekty o zmianie stanu pewnego innego obiektu. W programowaniu obiektowym obiekty posiadają pewien stan, tj. zbiór aktualnych wartości pól obiektu, który w wyniku wykonywania na nich operacji może ulegać zmianie. Od bieżącego stanu mogą być zależne inne obiekty, dlatego musi istnieć możliwość ich powiadomienia o jego zmianie tak, aby mogły one się do niej dostosować. Możemy także żądać, aby inne obiekty były powiadamiane o tym, że inny obiekt próbuje wykonać konkretną czynność, np. ponownie nawiązywać utracone połączenie z bazą danych. Pragniemy zaimplementować ogólny mechanizm, który umożliwi nam osiągnięcie tych celów.

- **MVC** - Model-View-Controller (pol. Model-Widok-Kontroler) – wzorzec architektoniczny służący do organizowania struktury aplikacji posiadających graficzne interfejsy użytkownika. Wiele prac traktuje go jako pojedynczy wzorzec, lecz może on być także traktowany jako złożony wzorzec wykorzystujący idee wzorców prostych, takich jak Obserwator, Strategia czy Kompozyt. Oba te podejścia nie wykluczają się. MVC nie był traktowany jako samodzielny wzorzec również w pracy „Design Patterns: Elements of Reusable Object-Oriented Software” autorstwa „Bandy Czworga”. Model-View-Controller zakłada podział aplikacji na trzy główne części: Zmiana danych przez użytkownika odbywa się za pomocą kontrolera, ten modyfikuje model, model odświeża widok, a informacja z widoku dociera do użytkownika. Model – jest pewną reprezentacją problemu bądź logiki aplikacji. Widok – opisuje, jak wyświetlić pewną część modelu w ramach interfejsu użytkownika. Może składać się z podwidoków odpowiedzialnych za mniejsze części interfejsu. Kontroler – przyjmuje dane wejściowe od użytkownika i reaguje na jego poczynania, zarządzając aktualizacje modelu oraz odświeżenie widoków.

13. Implementacja

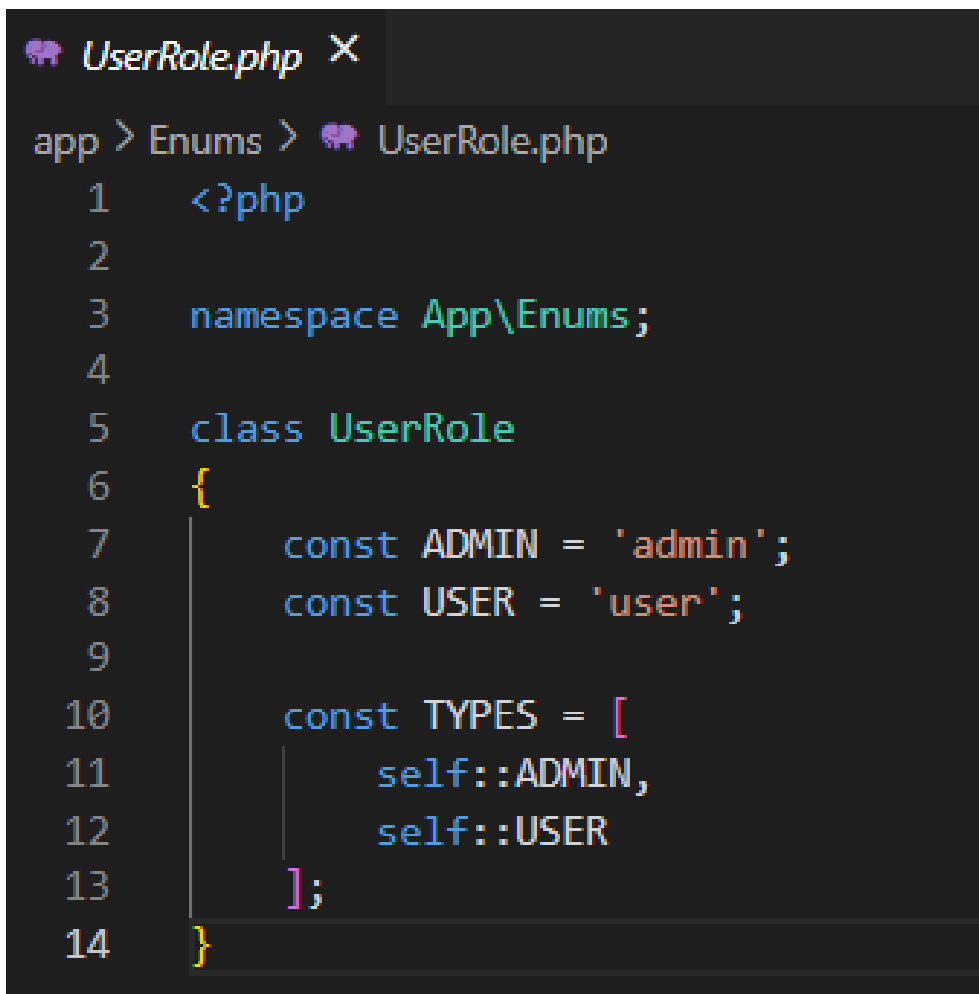
- Logowanie:

```
<form method="POST" action="{{ route('login') }}">
  @csrf
  <div class="form-group">
    <input id="email" type="email" class="form-control form-control-user @error('email') is-invalid @enderror" name="email" value="{{ old('email') }}"
    @error('email')
      <span class="invalid-feedback" role="alert">
        <strong>{{ $message }}</strong>
      </span>
    @enderror
  </div>
  <div class="form-group">
    <input id="password" type="password" class="form-control form-control-user @error('password') is-invalid @enderror" name="password" required=""
    @error('password')
      <span class="invalid-feedback" role="alert">
        <strong>{{ $message }}</strong>
      </span>
    @enderror
  </div>
  <div class="form-group">
    <div class="custom-control custom-checkbox">
      <input class="custom-control-input" type="checkbox" name="remember" id="customCheck" {{ old('remember') ? 'checked' : '' }}>
      <label class="custom-control-label" for="customCheck">Zapamiętaj mnie!</label>
    </div>
  </div>
</form>
```

```
protected $fillable = [
    'id',
    'name',
    'email',
    'role',
    'password',
];

/**
 * The attributes that should be hidden for serialization.
 *
 * @var array<int, string>
 */
protected $hidden = [
    'password',
    'remember_token',
];
```

- Rozróżnienie użytkowników:



```
app > Enums > UserRole.php
1  <?php
2
3  namespace App\Enums;
4
5  class UserRole
6  {
7      const ADMIN = 'admin';
8      const USER = 'user';
9
10     const TYPES = [
11         self::ADMIN,
12         self::USER
13     ];
14 }
```

- Usuwanie:



```
public function delete($id)
{
    $users = User::find($id);

    DB::connection('mysql')->delete(DB::raw('DELETE FROM users WHERE id='.$id.''));

    return redirect('users_list');
}
```

- Wyszukiwanie:

```
public function search(Request $request)
{
    $search = $request->get('search');

    $books = DB::table('book')->where(function($query) use ($request){
        $query->where('tytul', 'like', '%'.$request->search . "%");
        $query->orWhere('autor', 'like', '%'.$request->search . "%");
    })->get();

    return view('book_list', ['books' => $books]);
}
```

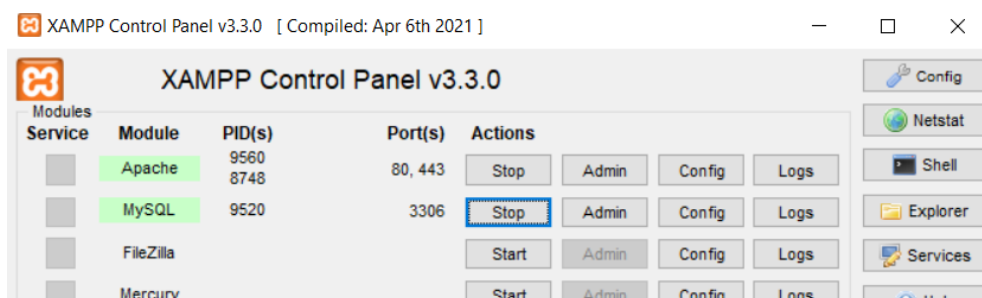
14. Uruchomienie oraz działanie aplikacji

Do poprawnego działania należy uruchomić server poleceniem **php artisan serve** oraz program xampp - serwer bazodanowy.

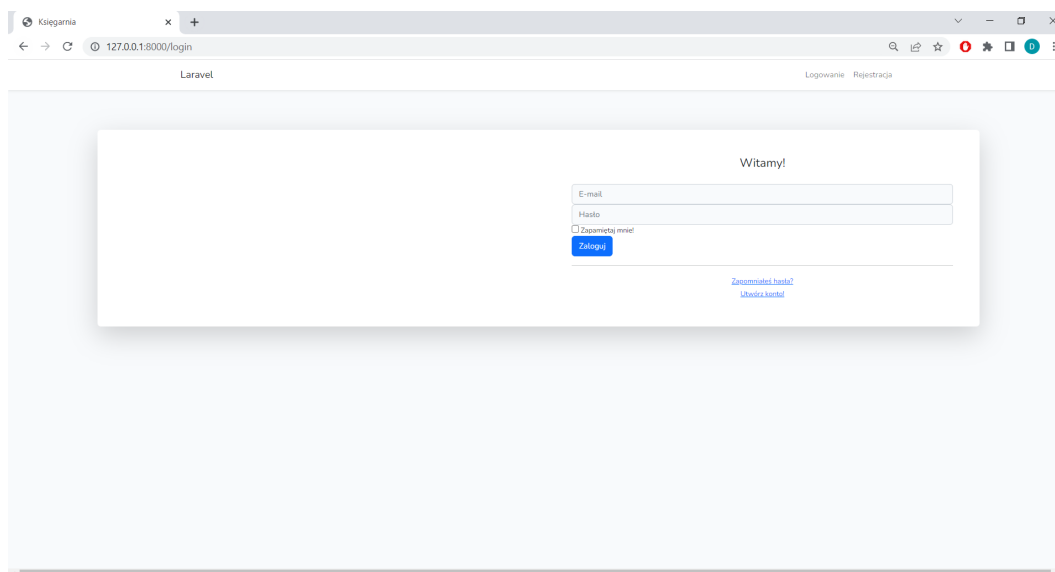
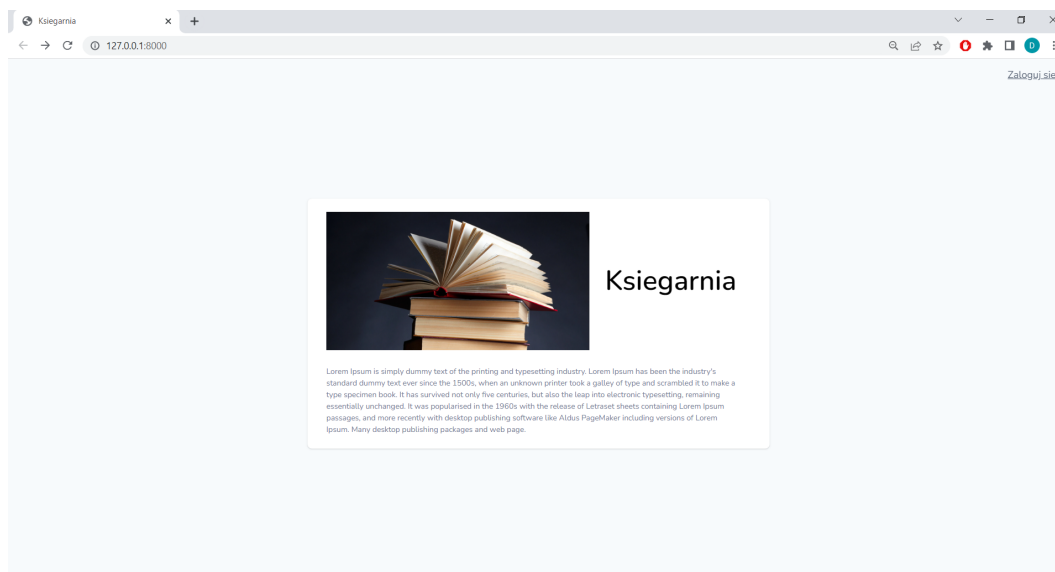
```
PS C:\ProjektBookshop\I04_P1\Ksiegarnia> php artisan serve

INFO Server running on [http://127.0.0.1:8000].

Press Ctrl+C to stop the server
```




Po uruchomieniu strony w przeglądarce ukazuje nam się okno startowe z możliwością zalogowania się.



Witamy!

☐ Zapamiętaj

 Uwzględnij znak „@” w adresie e-mail. W adresie „zle_dane” brakuje znaku „@”.

[Zapomniałeś hasła?](#)

[Utwórz konto!](#)

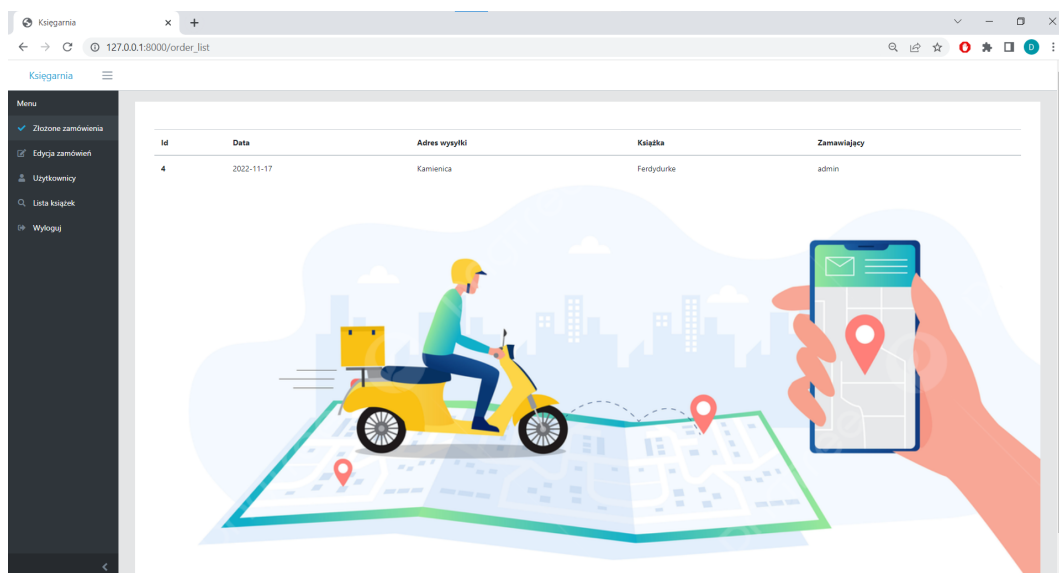
Witamy!

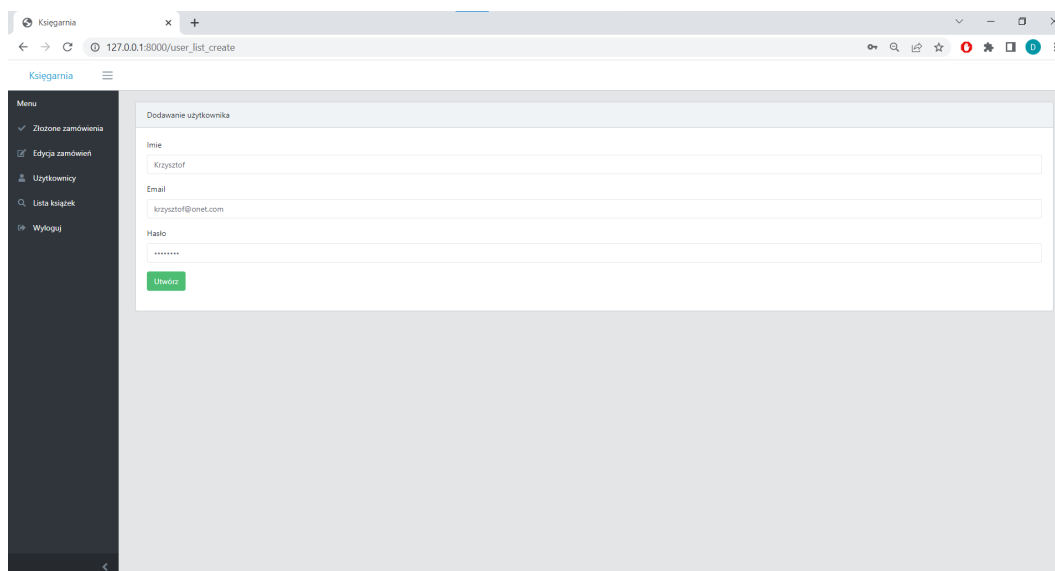
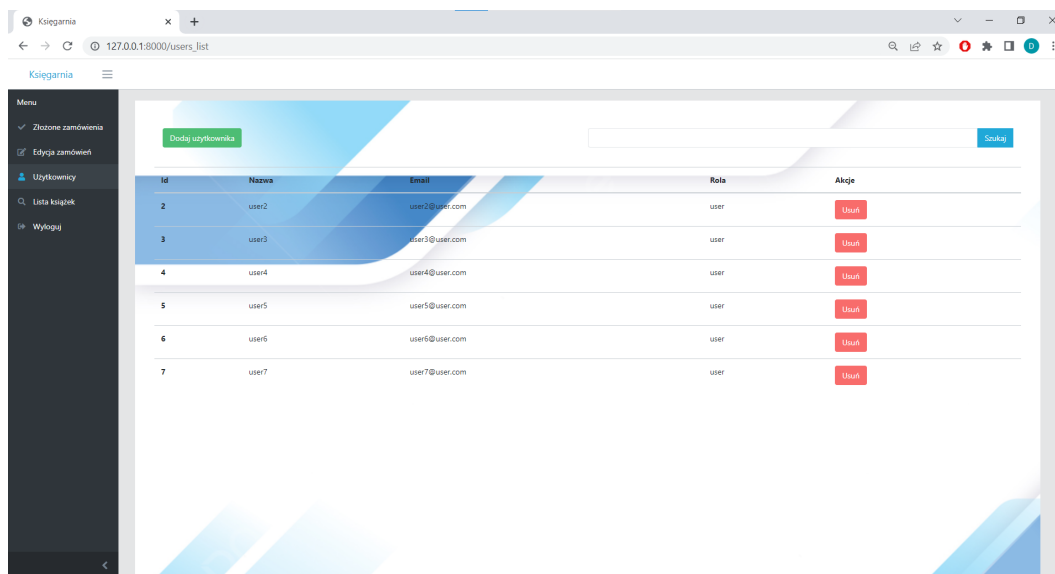
These credentials do not match our records.

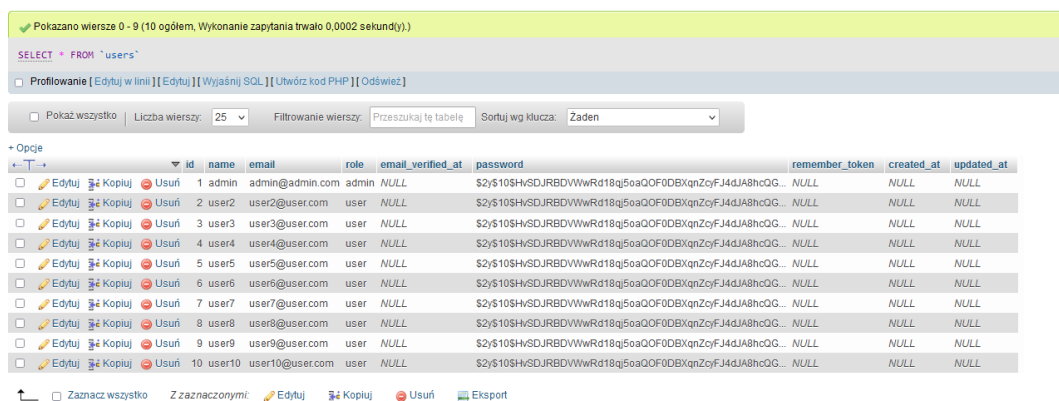
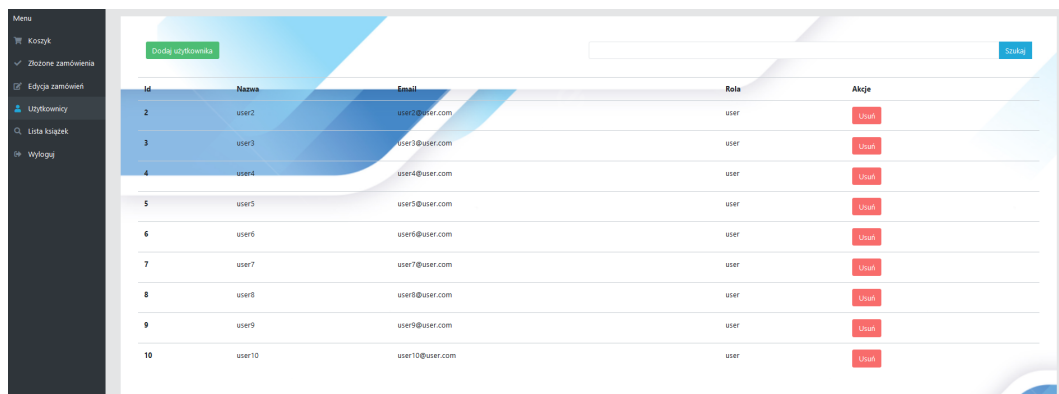
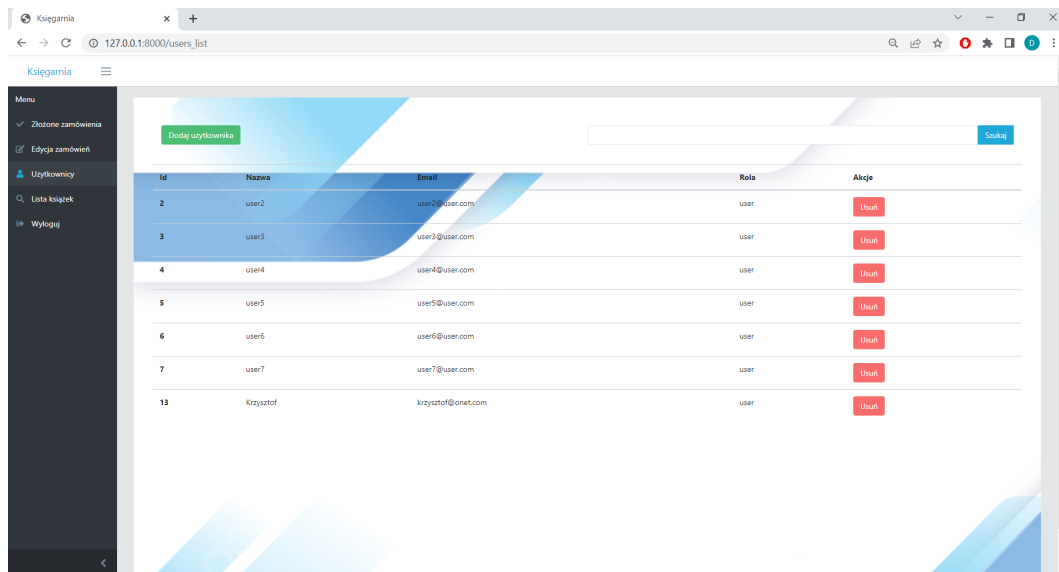
☐ Zapamiętaj mnie!

[Zapomniałeś hasła?](#)

[Utwórz konto!](#)







Dodawanie książki

Tytuł:

Cena:

Ilość stron:

Ilość sztuk:

Wydawnictwo:

Autor:

id	Tytuł	Cena	Ilość stron	Ilość sztuk	Wydawnictwo	Autor
1	Pan Tadeusz	49	300	200	GREG	Adam Mickiewicz
2	Ferdynand	30	250	50	OPERON	Witold Gombrowicz
3	Książka 3	80	890	15	GREG	Juliusz Słowacki

☐ Pokaż wszystko | Liczba wierszy: 25 | Filtrowanie wierszy: Przeszukaj tę tabelę | Sortuj wg klucza: Żaden

+ Opcje

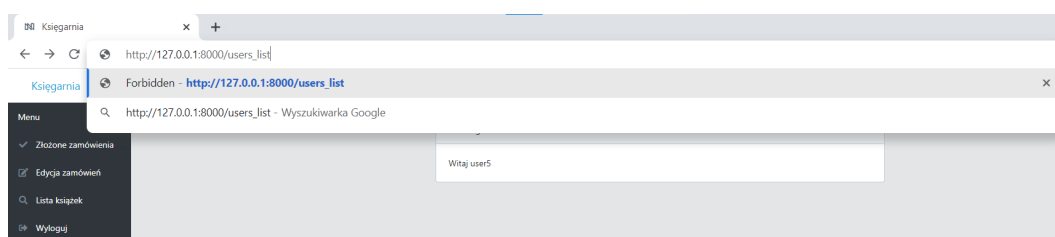
	id	tytuł	cena	ilosc_stron	ilosc_sztuk	wydawnictwo	autor
<input type="checkbox"/> Edytuj <input type="button" value="Kopiuj"/> <input type="button" value="Usuń"/>	1	Pan Tadeusz	49	300	200	GREG	Adam Mickiewicz
<input type="checkbox"/> Edytuj <input type="button" value="Kopiuj"/> <input type="button" value="Usuń"/>	2	Ferdynand	30	250	50	OPERON	Witold Gombrowicz
<input type="checkbox"/> Edytuj <input type="button" value="Kopiuj"/> <input type="button" value="Usuń"/>	3	Książka 3	80	890	15	GREG	Juliusz Słowacki

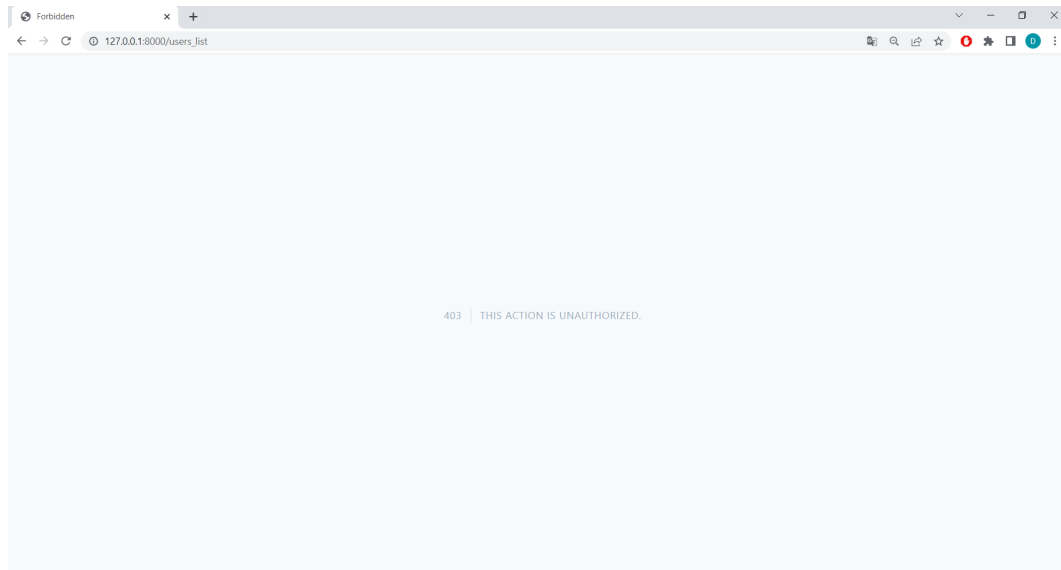
Z zaznaczonymi:

☐ Pokaż wszystko | Liczba wierszy: 25 | Filtrowanie wierszy: Przeszukaj tę tabelę | Sortuj wg klucza: Żaden

Operacja na wynikach zapytania

W przypadku gdy zwykły użytkownik będzie próbował dostać się do stron z funkcjonalności admina zostanie odmówiony.





15. Testowanie

1. **Testy funkcjonalne** - Aplikacja została zweryfikowana bez wnikania w kod źródłowy pod kątem poprawności wyświetlanych informacji, komunikacja między witryną i serwerem przebiega pomyślnie, nie generując przy tym żadnych błędów.
2. **Testy niefunkcjonalne** - Celem tego rodzaju testów było uzyskanie informacji o konkretnych właściwościach systemu i określonych modułów.
3. **Testy użyteczności** - Aplikacja została przetestowana pod kątem spełnienia założonych celów i wymagań.
4. **Testy wydajnościowe** - Działanie witryny zostało sprawdzone pod dużym obciążeniem.
5. **Testy przeciążeniowe** - Zweryfikowano działanie aplikacji, działającej na granicy swoich zasobów takich jak pamięć czy procesor.
6. **Testy obciążeniowe** - Weryfikacja dotycząca oprogramowania, podczas gdy potęguje się jego obciążenie. Przykładem może być test, w którym zwiększamy liczbę zalogowanych do aplikacji użytkowników – sprawdzając jak reaguje aplikacja.
7. **Testy przenaszalności** - Weryfikacja tego, w jaki sposób i z jakim poziomem trudności można przenieść system na inne środowisko. Nasza aplikacja działa praktycznie w każdym serwerem, ograniczeniem jest jedynie wymagana baza danych mySQL.
8. **Testy bezpieczeństwa** - Weryfikacja systemu pod kątem osłony i mechanizmów bezpieczeństwa, które mają chronić system przez niepożądanymi zachowaniami osób z wewnątrz i z zewnątrz. Dzięki konkretnym funkcjonalnościom Laravela, mamy możliwość łatwego implementowania autoryzacji. Dzięki temu nieautoryzowane osoby nie uzyskają dostępu do chronionych treści. Oprócz tego Laravel jest wyposażony w systemy chroniące przed atakami XSS (Cross-site Scripting), CSRF (Cross-site Request Forgery) oraz SQL, które stanowiły ponad połowę wszystkich ataków zarejestrowanych w roku 2017.

9. **Testy pielęgnowalności** - Weryfikacja pielęgnowalności, czyli tego jak łatwo można dostosować system do nowych wymagań oraz czy jest możliwość jego prostej modyfikacji do planowanych i nieplanowanych zmian. Nasza witryna dzięki wykorzystanemu frameworkowi jest skalowalna. Ideą modelu MVC jest ułatwienie zrozumienia działania aplikacji nawet przez developera, który dostał zlecenie dodania funkcjonalności, bez wcześniejszego poznania struktury i działania aplikacji.

16. Podsumowanie projektu i wnioski

- **Podsumowanie**

Celem naszego projektu było utworzenie witryny internetowej o tematyce księgarni w dowolnym języku. My wybraliśmy język PHP i JavaScript oraz framework Laravel. Główną rolę posiada administrator który jest w stanie np. usunąć książkę, użytkownika bądź też dodać książkę. Widok użytkownika oraz administratora różni się poprzez nadane prawa. Niestety, nie wszystkie funkcjonalności zostały zaimplementowane. Zabrakło m.in. koszyka do zakupu książki. Dzięki zrealizowaniu projektu zapoznaliśmy się ze strukturą baz danych, frameworku, języka oraz sposobu programowania całej działalności projektowej.

- **Wnioski**

Projekt utworzyliśmy w programie Visual Studio Code, przy wykorzystaniu frameworka Laravel, korzystaliśmy również z Githuba, który w ogromnym stopniu przyczynił się do ułatwienia oraz sprężenia wykonania pracy projektowej. Naszym zadaniem było utworzenie serwisu księgarni, gdzie użytkownik może utworzyć swoje konto, wyszukać interesujący go tytuł po nazwie książki bądź autorze oraz dokonać kupna towaru. Administrator systemu może zarządzać całą bazą danych systemu czyli dodawać użytkowników, edytować ich dane bądź usuwać. Może również dodawać, edytować lub usuwać książki dostępne w księgarni. Udało nam się wykonać większość zakładanych celów projektu. Został prawidłowo utworzony system wraz z wszelkimi dodatkami wizualnymi, wszelkie funkcjonalności działają bez zarzutów, nie zdołaliśmy jedynie wykonać systemu zamówień. Wykonanie projektu zaowocowało powiększeniem się naszego zakresu wiedzy na temat aplikacji webowych, struktur systemowych, baz danych oraz języków programowania. Dzięki wykonywanym diagramom byliśmy w stanie lepiej oszacować schemat wykonywania prac projektowych oraz ich zastosowanie. Nie można również zapominać o systemie Github, z którym mieliśmy do czynienia nieustannie od samego początku implementacji projektu. Podsumowując, wykonanie aplikacji webowej wymaga odpowiedniego czasu, przygotowania, wiedzy oraz zaangażowania. Staraliśmy się jak najlepiej wykonać aplikację poświęcając na nią czas oraz zasoby. Wykonaliśmy zdecydowaną większość zakładanych celów oraz poszerzyliśmy nasz zasób wiedzy tematycznej.

- **Link do repozytorium <https://github.com/kapizwa/IO4p1>**

17. Bibliografia

- [*https://getbootstrap.com/*](https://getbootstrap.com/)
- [*https://laravel.com/docs/9.x*](https://laravel.com/docs/9.x)
- [*https://dev.mysql.com/doc/*](https://dev.mysql.com/doc/)
- [*https://developer.mozilla.org/en-US/docs/Web/CSS*](https://developer.mozilla.org/en-US/docs/Web/CSS)
- [*https://online.visual-paradigm.com/*](https://online.visual-paradigm.com/)