

POLITECHNIKA WROCŁAWSKA
WYDZIAŁ INFORMATYKI I TELEKOMUNIKACJI

PROJEKT Z INTERNETOWYCH BAZ
DANYCH

Bookshop - księgarnia internetowa

Termin zajęć: Poniedziałek, 12:15–14:30

Autor/Autorzy:

Kamil Dywan

Indeks: 254049

Paweł Kuźma

Indeks: 252778

Mateusz Urbańczyk

Indeks: 252808

Prowadzący zajęcia:

dr inż. Roman Ptak, W4N

Wrocław, 2022 r.

1. Wstęp

1.1. Cel projektu

Projekt, implementacja oraz wdrożenie księgarni internetowej składającej się z bazy danych, interfejsu użytkownika i serwera dla firmy „Bookshop” znajdującej się na terenie Wrocławia oraz zajmującej się stacjonarną sprzedażą książek.

1.2. Plan projektu:

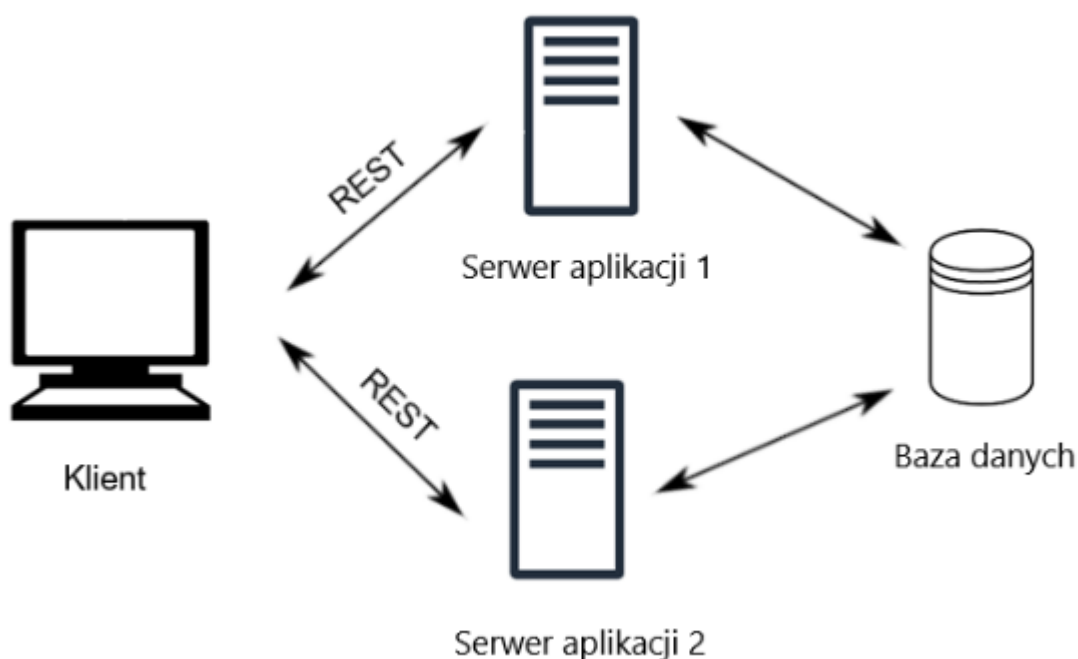
- Projektowanie:
 - Analiza SWOT
 - Dobór narzędzi
 - Wymagania funkcjonalne i нефункционалне
 - Diagramy UML
 - Diagramy ERD
 - Prototypy interfejsu użytkownika
- Implementacja:
 - Relacyjna baza danych
 - Backend
 - Aplikacja kliencka (frontend)
- Testowanie:
 - Testy integracyjne backendu
 - Testy jednostkowe
 - Testy manualne interfejsu użytkownika
 - Testy automatyczne interfejsu użytkownika
- Wdrożenie

1.3. Zakres projektu:

Zakres projektu obejmuje stworzenie i wdrożenie strony internetowej, która ma za zadanie ułatwić pracownikom zarządzanie zasobami księgarni (np. określenie listy dostępnych książek oraz ustalenie ich cen) oraz umożliwić klientom przeglądanie i zamawianie książek przez internet, które zamówione będą mogły być zakupione w sklepie stacjonarnym. Powinna być również opcja założenia konta przez klienta, które to konto będzie umożliwiało użytkownikowi przeglądanie listy dokonanych zamówień, sprawdzenie statusu zamówień, czy wycofanie zamówień, które nie są jeszcze w trakcie realizacji.

1.3. Architektura aplikacji

Realizowany system jest rozproszonym serwisem webowym, który w dużym uogólnieniu można opisać jako system typu klient-serwer. Klient pobiera źródło strony z serwera aplikacji nr 2 i w zależności od wybranych funkcji łączy się z jednym z serwerów aplikacji. Serwery odpowiednio przetwarzają otrzymane żądanie, komunikują się z bazą danych i zwracają klientowi odpowiedź, którą to później odpowiedź klient interpretuje i przedstawia użytkownikowi (w tym przypadku jest to GUI interfejsu webowego).



Zdecydowano się na zaimplementowanie systemu w postaci serwisu webowego, gdyż taka forma aplikacji zapewnia dużą wygodę użytkownika z powodu braku konieczności jej zainstalowania przez użytkownika, a z powodu tego, że założono, iż system powinien być dostępny dla każdego, kto posiada dostęp do Internetu, to nie ma wymogu instalowania aplikacji. Następnym powodem wybrania tego typu aplikacji jest to, że aplikacje webowe są obecnie najpopularniejszym typem systemu i implementacja takiej aplikacji pozwala na zdobycie wartościowego doświadczenia.

1.3.1 Baza danych

Baza danych to warstwa systemu odpowiedzialna za przechowywanie danych. Założono, że w systemie zostanie wykorzystana relacyjna baza danych (SQL). Jako bazę danych wybrano bazę relacyjną, gdyż bazy te charakteryzują się tym, że zapewniają integralność danych poprzez tworzenie relacji np. poprzez powiązanie tabel kluczem obcym,

a w realizowanym systemie wykryto wiele przypadków, w których można by było utworzyć relacje np. klient jest powiązaniem między encjami użytkownika i zamówień.

1.3.2. Serwer aplikacji - Backend

Backend jest odpowiedzialny za przyjmowanie żądań od klienta, odpowiednie przetwarzanie tych żądań, wykonywanie pewnych operacji na danych przechowywanych w bazie danych na podstawie otrzymanych przez klienta danych i przekazywanie klientowi adekwatnej odpowiedzi. Backend udostępnia klientowi punkty końcowe (ang. API endpoints) do których klient może przesłać żądanie. Punkty końcowe posiadają adresy URL np. `http://localhost:9000/user`, które składają się nazwy protokołu (`http`), nazwy hosta docelowego (`localhost`), portu docelowego (`9000`) oraz reszty, która identyfikuje punkt końcowy oraz jego przeznaczenie (dla przykładu punkt końcowy o końcówce adresu `user` związany jest z użytkownikami i klient może np. wysłać żądanie na ten endpoint do pobrania danych użytkownika o danym id). Ważnym elementem w przypadku backendu jest również wysłanie klientowi odpowiedniego statusu odpowiedzi oraz ewentualne dołączenie informacji zwrotnej do wiadomości, tak aby klient mógł odpowiednio zareagować na otrzymaną odpowiedź (np. w przypadku otrzymania negatywnej odpowiedzi, klient będzie mógł wyświetlić użytkownikowi komunikat o błędzie, a w przypadku otrzymania odpowiedzi negatywnej, klient będzie mógł wyświetlić komunikat o sukcesie). Warstwa ta jest w ścisłym powiązaniu z warstwą bazy danych.

1.3.2.1 Serwer aplikacji 1

Warstwa ta będzie odpowiedzialna za obsługę głównej części funkcji, które w założeniu ma oferować aplikacja webowa. Do skorzystania z usług dostarczanych przez ten serwer wymagane jest zalogowanie się użytkownika.

1.3.2.1 Serwer aplikacji 2

Warstwa ta będzie odpowiedzialna za przeglądanie listy dostępnych książek, szczegółów książki, wyszukiwanie książek po kryteriach oraz zwrócenie strony głównej.

1.3.3. Klient - Frontend

Frontend jest odpowiedzialny za wysyłanie żądań do warstwy backendowej i następnie odpowiednie przetwarzanie oraz wyświetlanie danych otrzymanych w odpowiedzi od backendu. W systemie tym klientem jest strona internetowa renderowana po stronie użytkownika. Klient jest aplikacją typu SPA (ang. Single Page Application), co oznacza, że aplikacja składa się z jednej strony HTML, a jej zawartość jest zmieniana dynamicznie poprzez JavaScript. Zdecydowano się na aplikację typu SPA, a nie MPA (ang. Multi Page Application) głównie dlatego, że aplikacje SPA działają szybciej z powodu tylko jednej

strony HTML, co jest w przypadku tego systemu konieczne, gdyż przewidziano, że aplikacja będzie się składać z wielu podstron. Kolejnymi kryteriami na korzyść SPA były łatwiejsze wdrożenie, gdyż wystarczą jedynie 3 pliki (HTML, CSS i JavaScript), łatwiejsze wprowadzenie dynamicznej zawartości oraz ułatwione tworzenie aplikacji.

1.3.4 REST

Komunikacja w systemie między frontendem i backendem, frontendem i serwerem bezpieczeństwa oraz backendem i serwerem bezpieczeństwa odbywa się za pomocą REST. REST jest to sposób i format w jaki komunikuje się klient z serwerem. Serwer udostępnia klientowi punkty końcowe (end-pointy), do których klient może wysłać żądania HTTP przesyłając przy tym jakieś dane np. tytuł wyszukiwanego artykułu. W skrócie komunikacja REST odznacza się następującymi cechami:

- bezstanowość (nie przechowuje informacji o poprzednich wymianach wiadomości),
- architektura klient-serwer,
- jednolity interfejs komunikacyjny – dzięki temu możliwe jest np. komunikowanie się systemów zaimplementowanych w różnych językach programowania, czy zainstalowanych na różnych platformach (mobilna, czy aplikacja webowa),
- wykorzystywanie protokołu HTTP.

1.3.5 HTTP

Protokół warstwy aplikacji opisujący zasady przesyłania zawartości w Internecie. Protokół ten jest oparty na komunikacji typu klient-serwer, przy czym klientem jest najczęściej przeglądarka. Klientem wysyła żądanie do serwera, a serwer wysyła do klienta odpowiedź. Zawartość wiadomości HTTP można przede wszystkim podzielić na nagłówek (ang. Header) oraz ładunek (ang. Body). W nagłówku znajdują się pola typu klucz-wartość. Nagłówek żądania różni się od nagłówka odpowiedzi poprzez dostępne opcje, ale mają one również wspólne pola np. informacja o hoście docelowym, czy format wiadomości. W nagłówku żądania można wyróżnić jeszcze np. nagłówek Authorization, w którym umieszczane są poświadczenia uwierzytelnienia. W nagłówku odpowiedzi można za to wyróżnić m.in. status odpowiedzi, który np. informuje o tym, czy żądanie zostało wykonane pomyślnie (np. status 200), czy też wystąpił błąd, a jeśli wystąpił, to z jakiego powodu (np. status 401 oznaczający nieudane uwierzytelnienie użytkownika). Ładunek jest opcjonalnym polem, które stanowi treść wiadomości HTTP i np. w przypadku odpowiedzi może zawierać listę wyszukiwanych produktów.

1.3.6 OAuth 2.0

1.4. Techniczna wykonalność systemu

1.4.1. Projektowanie

a) Visual Paradigm:

- Zakres: diagramy UML i ERD
- Wersja: 17.0
- Licencja: Academic license
- Koszt: 0 zł

b) Figma:

- Zakres: projektowanie makiet interfejsu użytkownika
- Licencja: Freeware
- Koszt: 0 zł

1.4.2. Implementacja

a) MySQL:

- Zakres: stworzenie relacyjnej bazy danych
- Wersja: 8.0.31
- Licencja: GPL license
- Koszt: 0 zł

b) Java:

- Zakres: aplikacja serwerowa 1
- Wersja: 17
- Licencja: Oracle No-Fee Terms and Conditions
- Koszt: 0 zł

c) Apache Maven:

- Zakres: aplikacji serwerowa 1 - budowanie projektów Javowych
- Wersja: 3.8.6
- Licencja: Apache License 2.0
- Koszt: 0 zł

d) Lombok:

- Zakres: aplikacji serwerowa 1 - generowanie getterów, setterów, konstruktorów itp. dla projektów Javowych
- Wersja: 1.18.24
- Licencja: MIT
- Koszt: 0 zł

e) Mapstruct:

- Zakres: aplikacji serwerowa 1 - generowanie mapowań obiektów dla projektów Javowych
- Wersja: 1.15.3
- Licencja: Apache License 2.0
- Koszt: 0 zł

f) Spring Boot:

- Zakres: framework Javy dla aplikacji serwerowej 1
- Wersja: 2.7.5
- Licencja: Apache License 2.0
- Koszt: 0 zł

g) Python:

- Zakres: aplikacja serwerowa 2
- Wersja: 3.10.8
- Licencja: Open Source
- Koszt: 0 zł

h) Django:

- Zakres: framework Pythona dla aplikacji serwerowej 2
- Wersja: 4.1.2
- Licencja: 3-clause BSD license
- Koszt: 0 zł

i) HTML:

- Zakres: aplikacja kliencka - zawartość stron
- Wersja: 5.3
- Licencja: Royalty-free license
- Koszt: 0 zł

j) CSS:

- Zakres: aplikacja kliencka - wygląd strony
- Wersja: 3
- Licencja: MIT
- Koszt: 0 zł

k) JavaScript:

- Zakres: aplikacja kliencka - logika strony
- Wersja: ECMAScript 2021
- Licencja:
- Koszt: 0 zł

l) React:

- Zakres: biblioteka JavaScript dla aplikacji klienckiej
- Wersja: 18.2.0
- Licencja: MIT
- Koszt: 0 zł

ł) IntelliJ IDEA Ultimate:

- Zakres: środowisko programistyczne dla aplikacji serwerowej 1
- Wersja: 2022.2.3
- Licencja: Trialware (dla celów edukacyjnych)
- Koszt: 0 zł

m) Visual Studio Code:

- Zakres: środowisko programistyczne do tworzenia frontendu
- Wersja: 1.72.2
- Licencja: MIT
- Koszt: 0 zł

n) MySQL Workbench:

- Zakres: zarządzanie MySQL
- Wersja: 8.0.22
- Licencja: GPL
- Koszt: 0 zł

o) GitHub:

- Zakres: repozytorium kodu
- Licencja: Free
- Koszt: 0 zł

1.4.3. Testowanie

a) JUnit:

- Zakres: testy integracyjne aplikacji serwerowej 1
- Wersja: 5.9.1
- Licencja: Eclipse Public License 2.0
- Koszt: 0 zł

b) REST-assured:

- Zakres: testowanie RESTowych serwisów aplikacji serwerowej 1
- Wersja: 5.2.0
- Licencja: Apache License 2.0
- Koszt: 0 zł

c) Postman:

- Zakres: testowanie żądań HTTP dla aplikacji serwerowych
- Wersja: 9.4
- Licencja: OSS
- Koszt: 0 zł

d) Selenium:

- Zakres: automatyczne testy akceptacyjne aplikacji klienckiej
- Wersja: 4.5.0
- Licencja: Apache Software License (Apache 2.0)
- Koszt: 0 zł

1.4.4. Wdrożenie

a) Microsoft Azure:

- Zakres: Wdrożenie całej aplikacji
- Licencja: Academic license
- Koszt: 0 zł

Wykorzystując własne umiejętności, posiadając wiedzę na temat tworzenia aplikacji webowych oraz posiadając odpowiednie narzędzia do wytwarzania oprogramowania, jesteśmy w stanie wykonać tę aplikację dla klienta w przeciągu 10 tygodni, które mamy. Wszystkie aktualnie wykorzystywane technologie są dla nas dostępne na licencjach darmowych lub akademickich.

1.5. Analiza SWOT

1.5.1 Identyfikacja czynników

	Pozytywne	Negatywne
Wewnętrzne	<ul style="list-style-type: none"> - znajomość wykorzystywanych technologii - możliwość pracy zdalnej - łatwa współpraca dzięki niewielkiemu rozmiarowi grupy 	<ul style="list-style-type: none"> - brak renomy firmy - duża złożoność systemu - ograniczony czas na realizację
Zewnętrzne	<ul style="list-style-type: none"> - nawiązanie współpracy z innymi firmami w celu dalszego rozszerzania możliwości aplikacji - pojawienie się nowych technologii na rynku - potencjalne rozszerzenie asortymentu 	<ul style="list-style-type: none"> - ataki hackerów - zmiana przepisów prawnych - spadek popytu na papierowe książki

1.5.2 Ważność czynników

Mocne strony	Waga	Słabe strony	Waga
znaj. techn.	0,15	brak renomy	0,10
praca zdalna	0,35	złoż. systemu	0,45
niewiel. rozm. grupy	0,50	czas realizacji	0,45
	1,00		1,00
Szanse	Waga	Zagrożenia	Waga
współ. z inn. firm.	0,35	ataki hackerów	0,35
nowe technologie	0,35	zmiana przepisów	0,20
rozszerz. asort.	0,30	spadek popytu	0,45
	1,00		1,00

1.5.3 Analiza powiązań SWOT

Czy określona mocna strona pozwala wykorzystać daną szansę?

Szanse/ Mocne strony	[O1]	[O2]	[O3]	Waga	Liczba interakcji	Iloczyn wag i interakcji	Ranga
[S1]	0	0	1	0,15	1	0,15	3
[S2]	1	0	0	0,35	1	0,35	2
[S3]	1	0	0	0,5	1	0,5	1
Waga	0,35	0,35	0,3				
Liczba interakcji	2	0	1				
Iloczyn wag i interakcji	0,7	0	0,3				
Ranga	1	3	2				
Suma interakcji					6		
Suma iloczynó w						2	

Czy określona mocna strona pozwala ograniczyć dane zagrożenie?

Zagrożenia / Mocne strony	[T1]	[T2]	[T3]	Waga	Liczba interakcji	Iloczyn wag i interakcji	Ranga
[S1]	1	0	0	0,15	1	0,15	2
[S2]	0	0	0	0,35	0	0	3
[S3]	1	1	0	0,5	2	1	1
Waga	0,35	0,35	0,3				
Liczba interakcji	2	1	0				
Iloczyn wag i interakcji	0,7	0,35	0				
Ranga	1	2	3				
Suma interakcji					6		
Suma iloczynów						2,2	

Czy określona słaba strona ogranicza możliwość wykorzystania danej szansy?

Szanse/ Słabe strony	[O1]	[O2]	[O3]	Waga	Liczba interakcji	Iloczyn wag i interakcj i	Ranga
[W1]	1	0	0	0,15	1	0,15	3
[W2]	0	1	1	0,35	2	0,7	2
[W3]	1	1	1	0,5	3	1,5	1
Waga	0,35	0,35	0,3				
Liczba interakcji	2	2	2				
Iloczyn wag i interakcji	0,7	0,7	0,6				
Ranga	1	1	3				
Suma interakcji					12		
Suma iloczynó w						4,35	

Czy określona słaba strona potęguje dane zagrożenie?

Zagrożenie / Słabe strony	[T1]	[T2]	[T3]	Waga	Liczba interakcji	Iloczyn wag i interakcji	Ranga
[W1]	0	0	1	0,15	1	0,15	3
[W2]	1	0	0	0,35	1	0,35	2
[W3]	1	1	0	0,5	2	1	1
Waga	0,35	0,35	0,3				
Liczba interakcji	2	1	1				
Iloczyn wag i interakcji	0,7	0,35	0,3				
Ranga	1	3	2				
Suma interakcji					8		
Suma iloczynów						2,85	

1.5.4 Analiza powiązań TOWS

Czy określona szansa potęguje daną silną stronę?

Szanse/ Mocne strony	[S1]	[S2]	[S3]	Waga	Liczba interakcji	Iloczyn wag i interakcj i	Ranga
[O1]	0	1	0	0,15	1	0,15	3
[O2]	1	0	0	0,35	1	0,35	2
[O3]	1	0	0	0,5	1	0,5	1
Waga	0,35	0,35	0,3				
Liczba interakcji	2	1	0				
Iloczyn wag i interakcji	0,7	0,35	0				
Ranga	1	2	3				
Suma interakcji					6		
Suma iloczynó w						2,05	

Czy określone zagrożenie ogranicza daną silną stronę?

Zagrożenia / Mocne strony	[S1]	[S2]	[S3]	Waga	Liczba interakcji	Iloczyn wag i interakcji	Ranga
[T1]	0	0	0	0,15	0	0	1
[T2]	0	0	0	0,35	0	0	1
[T3]	0	0	0	0,5	0	0	1
Waga	0,35	0,35	0,3				
Liczba interakcji	0	0	0				
Iloczyn wag i interakcji	0	0	0				
Ranga	1	1	1				
Suma interakcji					0		
Suma iloczynów						0	

Czy określona szansa pozwala osłabić (przezwyciężyć) daną słabą stronę?

Szanse/ Słabe strony	[W1]	[W2]	[W3]	Waga	Liczba interakcji	Iloczyn wag i interakcj i	Ranga
[O1]	1	1	1	0,15	3	0,45	2
[O2]	0	0	0	0,35	0	0	3
[O3]	1	0	0	0,5	1	0,5	1
Waga	0,35	0,35	0,3				
Liczba interakcji	2	1	1				
Iloczyn wag i interakcji	0,7	0,35	0,3				
Ranga	1	2	3				
Suma interakcji					8		
Suma iloczynów						2,3	

Czy określone zagrożenie wzmacnia daną słabą stronę?

Zagrożenie / Słabe strony	[W1]	[W2]	[W3]	Waga	Liczba interakcji	Iloczyn wag i interakcji	Ranga
[T1]	0	1	1	0,15	2	0,3	3
[T2]	0	1	1	0,35	2	0,7	1
[T3]	1	0	0	0,5	1	0,5	2
Waga	0,35	0,35	0,3				
Liczba interakcji	1	2	2				
Iloczyn wag i interakcji	0,35	0,7	0,6				
Ranga	3	1	2				
Suma interakcji					10		
Suma iloczynów						3,15	

1.5.5 Wnioski oraz wybór strategii i uzasadnienie

Wyniki zbiorcze analizy SWOT/TOWS

Kombinacja	Wyniki analizy SWOT		Wyniki analizy TOWS		Zestawienie zbiorcze SWOT/TOWS	
	Suma interakcji	Suma iloczynów	Suma interakcji	Suma iloczynów	Suma interakcji	Suma iloczynów
Mocne strony [S]/ Szanse [O]	6	2	6	2,05	12	4,05
Mocne strony [S]/ Zagrożenia [T]	6	2,2	0	0	6	2,2
Słabe strony [W]/ Szanse [O]	12	4,35	8	2,3	20	6,65
Słabe strony [W]/ Zagrożenia [T]	8	2,85	10	3,15	18	6

Wyniki analizy strategicznej i wybór strategii

	Szanse	Zagrożenia
Mocne strony	Strategia agresywna	Strategia konserwatywna
	Liczba interakcji	Liczba interakcji
	12	6
	Ważona liczba interakcji	Ważona liczba interakcji
	4,05	2,2
Słabe strony	Strategia konkurencyjna	Strategia defensywna
	Liczba interakcji	Liczba interakcji
	20	18
	Ważona liczba interakcji	Ważona liczba interakcji
	6,65	6

Z powodu przeważających słabych stron i powiązanych z nimi szansami, zamierzamy przyjąć strategię konkurencyjną. Chcemy wykorzystać zależność pomiędzy naszymi słabymi stronami a szansami, jakie daje otoczenie, czyli przezwyciężyć szansami słabe strony.

Powinniśmy skupić się na wykorzystaniu szansy w postaci możliwości współpracy z innymi firmami w celu przezwyciężenia słabych stron w postaci braku renomy wśród klientów, dużej złożoności systemu czy ograniczonego czasu na wykonanie projektu.

Szansą dla kręgielni mogą być także pojawienie się nowych technologii oraz poszerzenie asortymentu o ebooki czy audiobooki.

1.6. Kosztorys

- oprogramowanie - 0 zł
- wynagrodzenie pracowników - 3 prac. * 60 h * 50 zł/h = 9000 zł
- utrzymanie serwerów - 0 zł
- zakup domeny - 48,00 zł za pierwszy rok (bookshop.sklep.pl)
- hosting - 6520,23 zł (Azure w planie B1)

Całkowity, obliczony koszt wykonania i utrzymania systemu przez rok wynosi 15577,32 zł. Wartość ta mieści się w posiadanym budżecie, równym 20000 zł.

1.7. Harmonogram

Przygotowanie i uzasadnienie potrzeby realizacji. Analiza problemu.	24.10
Faza projektowa	07.11
Implementacja, testowanie i wdrożenie aplikacji	21.11
Prezentacja na forum publicznym i zaliczenie całości	05.12

1.8. Podział obowiązków

Kamil Dywan	Projektowanie, Baza danych, Aplikacja serwerowa 1, Frontend, Wdrożenie
Paweł Kuźma	Projektowanie, Baza danych, Aplikacja serwerowa 1, Frontend
Mateusz Urbańczyk	Projektowanie, Baza danych, Aplikacja serwerowa 2

2. Projektowanie

2.1 Role użytkowników

- a) Użytkownik - rola przypisywana każdemu użytkownikowi w systemie. Użytkownik może przede wszystkim przeglądać ofertę księgarni.
 - i) Niezalogowany użytkownik - rola, którą posiada każdy niezalogowany użytkownik. Użytkownik może założyć konto oraz się zalogować.
 - ii) Zalogowany użytkownik - rola przypisywana każdemu użytkownikowi, który się zalogował. Użytkownik może przede wszystkim zmienić swoje dane osobowe oraz wylogować się.
 - 1) Klient - rola przypisywana każdemu użytkownikowi, który zalogował się na konto klienta. Klient może przede wszystkim zarządzać swoim koszykiem i własnymi zamówieniami.
 - 2) Pracownik - rola przypisywana każdemu użytkownikowi, który zalogował się na konto pracownika. Założono, że w systemie będą już utworzone konta pracowników. Pracownik może przede wszystkim zarządzać zasobami sklepu oraz zamówieniami.

2.2. Wymagania funkcjonalne

- a) Wymagania związane ogólnie z użytkownikami:
 - przeglądanie listy dostępnych książek.
- b) Wymagania związane wyłącznie z niezalogowanym użytkownikiem:
 - założenie konta,
 - zalogowanie się.
- c) Wymagania związane wyłącznie z zalogowanym użytkownikiem:
 - wylogowanie się.
- d) Wymagania związane wyłącznie z klientem:
 - modyfikacja danych osobowych,
 - dodanie i usunięcie książki z koszyka,
 - zamówienie książek do odbioru w sklepie,
 - sprawdzenie statusu własnego zamówienia,
 - wycofanie własnego zamówienia niebędącego jeszcze w trakcie realizacji.

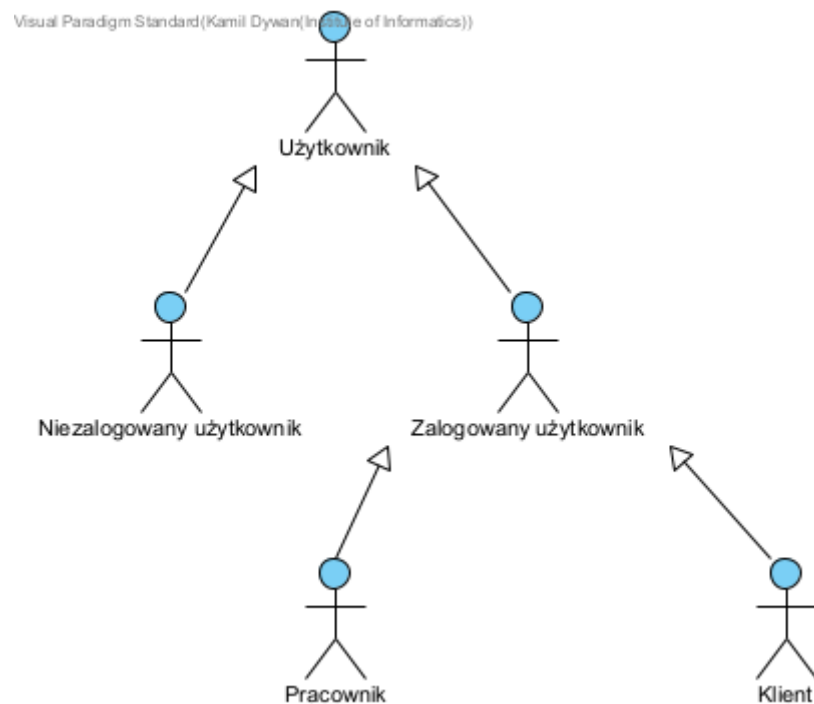
- e) Wymagania związane wyłącznie z pracownikiem:
- przeglądanie listy książek,
 - dodawanie, usuwanie i edytowanie parametrów książek będących w systemie,
 - dodawanie i usuwanie książek z oferty dostępnej dla klientów,
 - przeglądanie listy zamówień,
 - zmiana statusu zamówienia.

2.3. Wymagania niefunkcjonalne

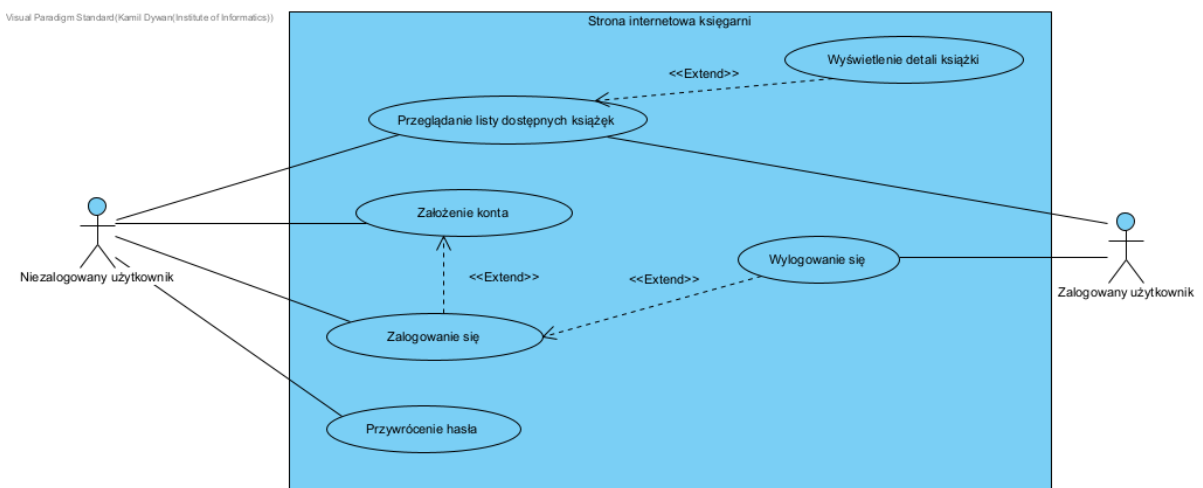
- Wymagania dotyczące bezpieczeństwa systemu:
 - tworzone hasła powinny spełniać następujące kryteria:
 - minimalna długość - 8 znaków,
 - co najmniej jedna mała litera,
 - co najmniej jedna duża litera,
 - co najmniej jedna cyfra.
 - bezpieczeństwo dostępu do systemu powinno być zapewnione poprzez użycie tokenów. Użytkownik nie będzie mógł skorzystać z danego endpointu albo wejść na daną podstronę, jeśli nie posiada wymaganej roli.
- Wymagania technologiczne:
 - komunikacja pomiędzy klientem i serwerami powinna odbywać się przy pomocy REST API,
 - dostęp do systemu będzie zabezpieczony poprzez protokół OAuth 2.0,
 - oprogramowanie serwera aplikacji 1 powinno być stworzone przy użyciu języka Java i frameworka Spring,
 - oprogramowanie serwera aplikacji 2 powinno być stworzone przy użyciu języka Python i frameworka django,
 - frontend aplikacji powinien być stworzony przy pomocy technologii HTML 5, CSS3 i Javascript z frameworkiem React,
 - baza danych powinna być stworzona przy użyciu technologii MySQL,
 - aplikacja powinna zostać przetestowana przy pomocy narzędzi JUnit, REST-assured, Postman, Selenium,
 - wdrożenie aplikacji powinno być wykonane w usłudze Google Cloud.

2.4. Diagramy przypadków użycia

Aktorzy:



Diagramy przypadków użycia dla aktorów:



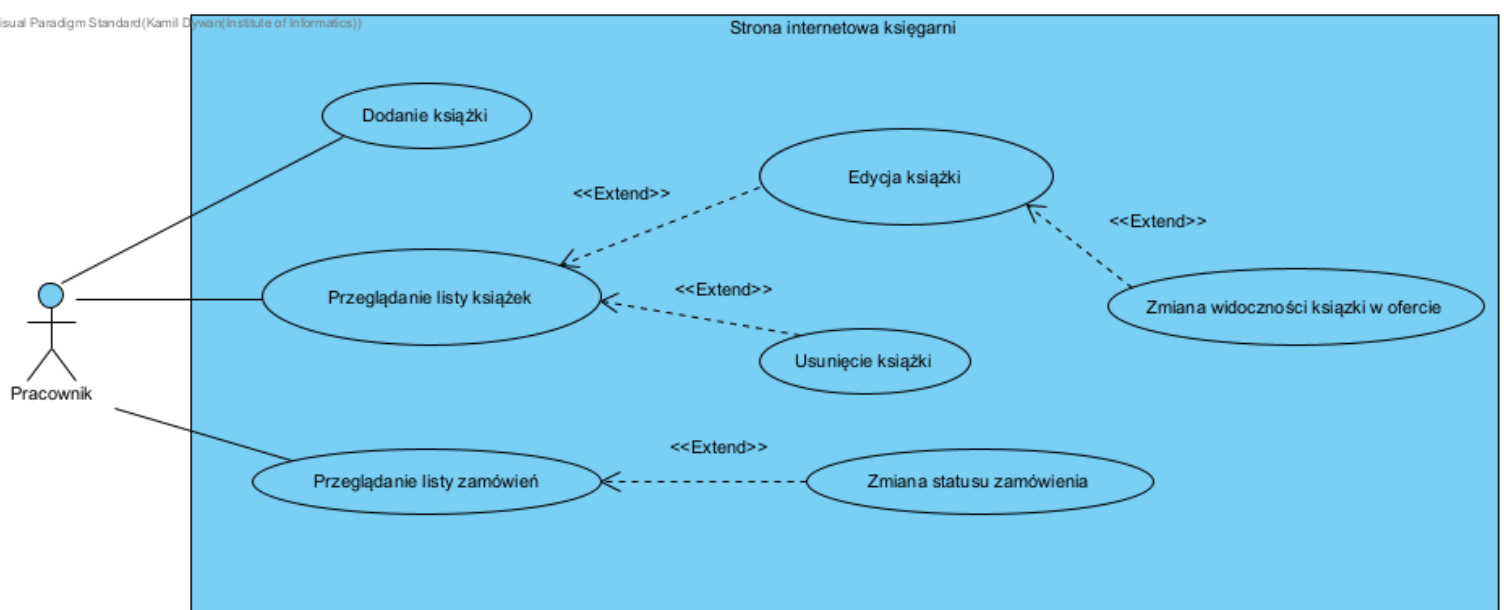
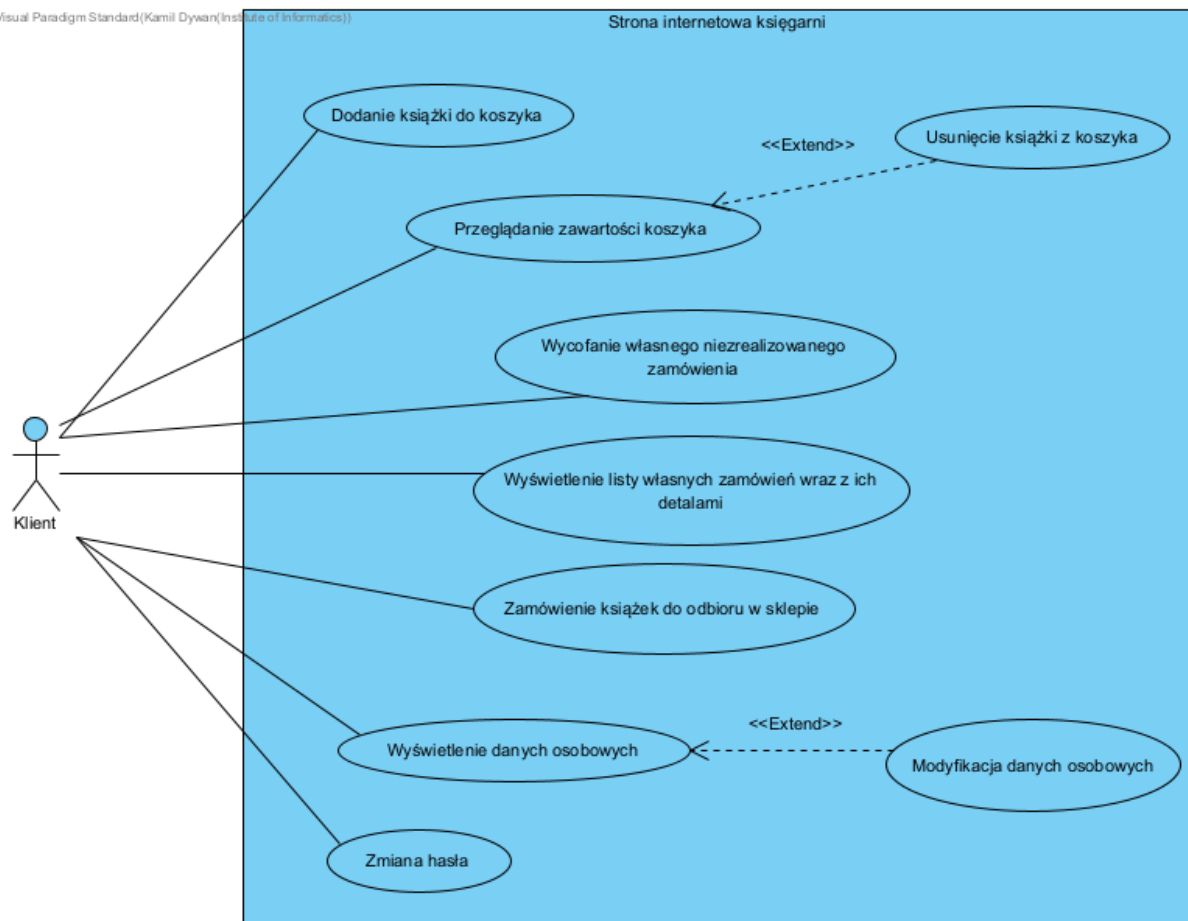
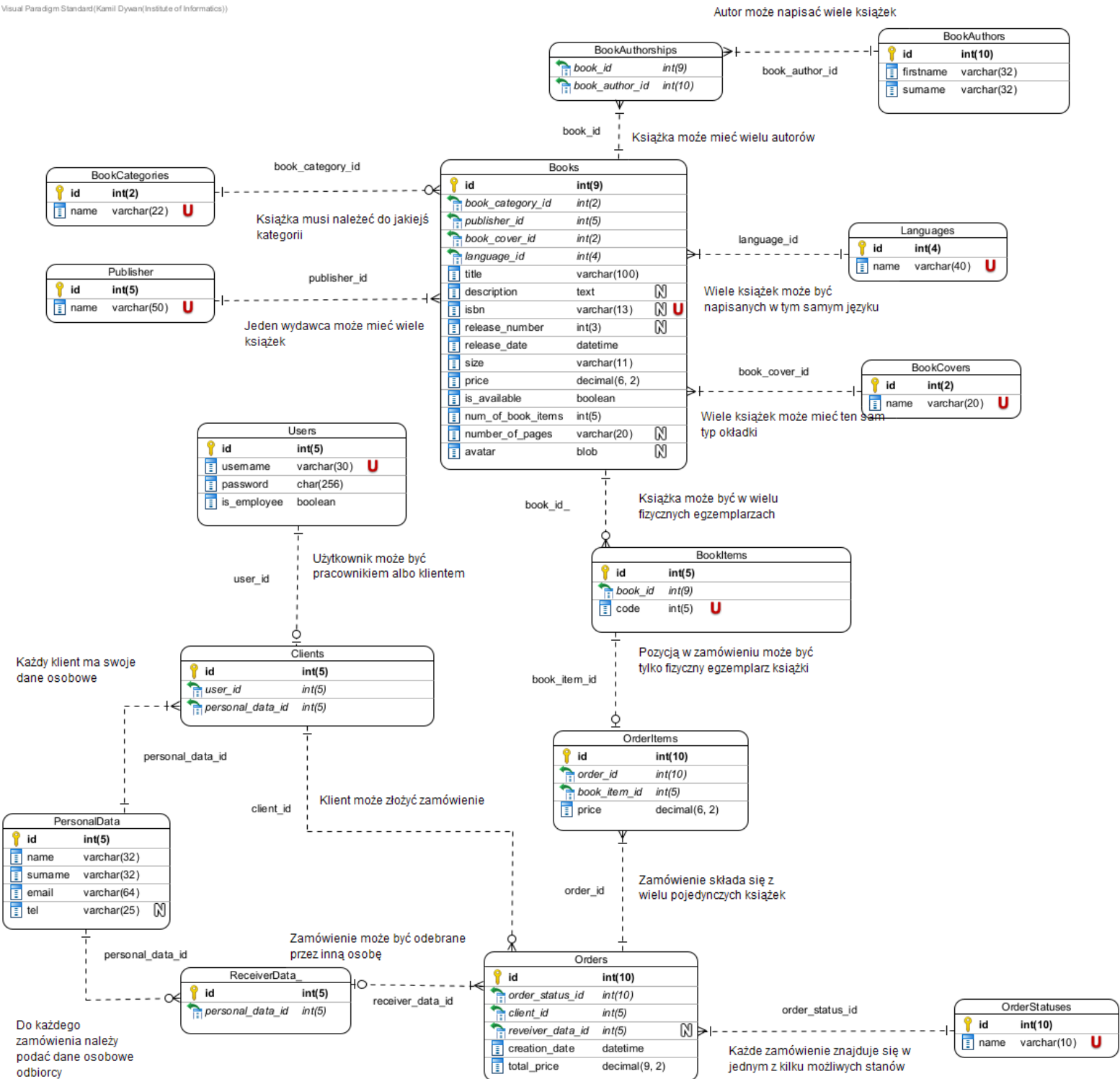


Diagram ERD

Visual Paradigm Standard(Kamil Dywan(Institute of Informatics))



2.5. Endpointy serwerów

a) Endpointy dla serwera aplikacji 1:

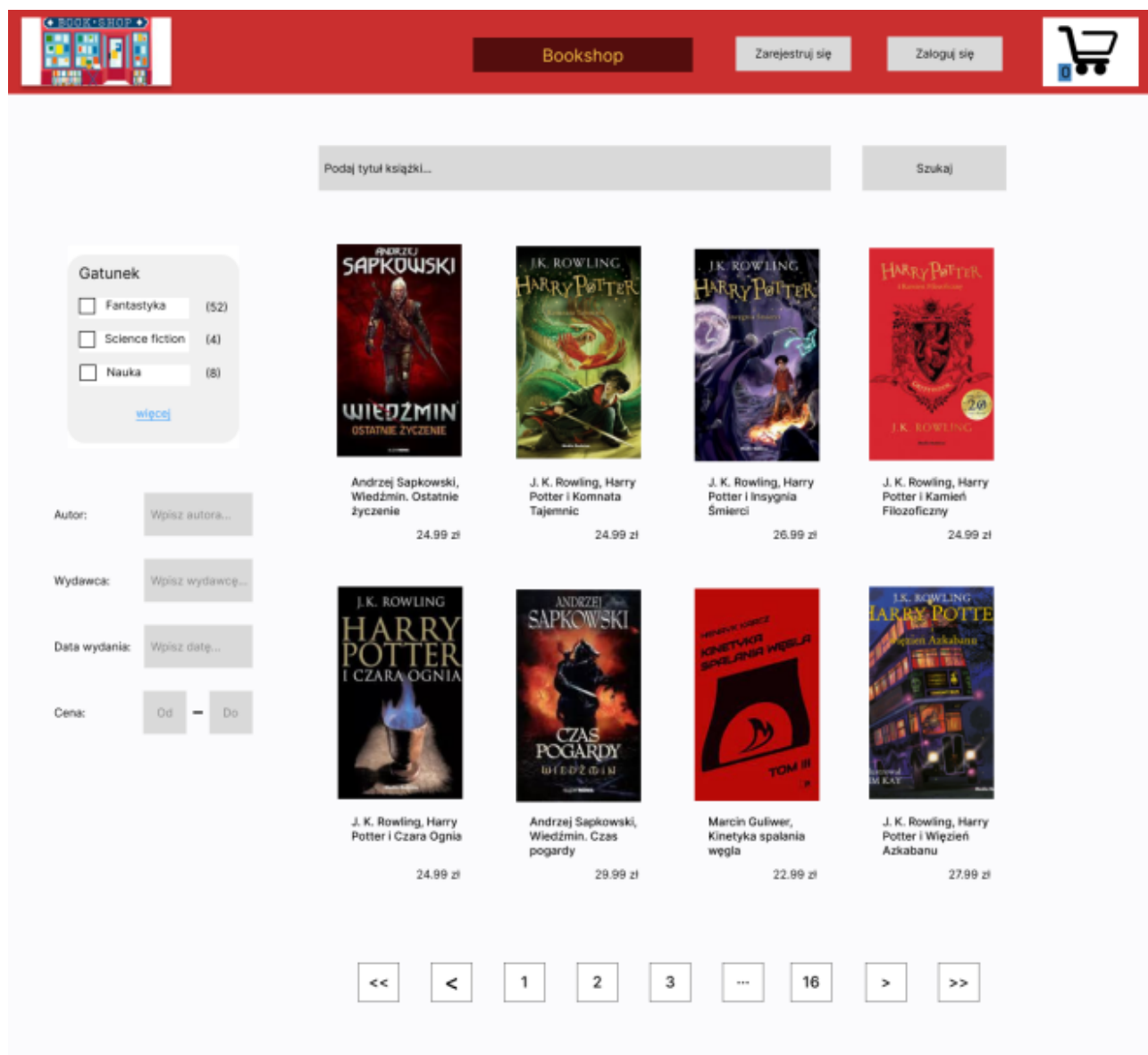
- GET /orders - wyszukanie zamówień po kryteriach oraz paginacji,
- GET /orders/{clientId} - zwrócenie zamówień klienta o podanym id,
- POST /order - zamówienie książek znajdujących się w koszyku,
- PUT /order/{orderId} - edycja statusu zamówienia o podanym id przez pracownika,
- PUT /order/{orderId}/cancel - wycofanie zamówienia o podanym id,
- GET /books/find - wyszukiwanie książek po liście parametrów oraz paginacji,
- POST /book - dodanie nowej książki,
- PUT /book/{bookId} - edycja parametrów książki o podanym id,
- DELETE /book/{bookId} - usunięcie książki o podanym id,
- PUT /user/{userId} - edycja danych użytkownika,
- PUT /client/{clientId} - edycja danych personalnych,
- POST /register - zarejestrowanie się,
- POST /token - aktualizowanie tokena dostępu,
- POST /login - zalogowanie się,
- DELETE /logout/{sessionId} - wylogowanie się.

b) Endpointy dla serwera aplikacji 2:


- GET / - zwrócenie źródła strony
- GET /books/all - pobranie danych wszystkich książek oraz kategorii książek dostępnych w sklepie
- GET /book/{bookId} - wyszukanie książki po id,
- GET /books/find/ - wyszukiwanie dostępnych książek według ustalonych kryteriów (kryteria jako parametry - params)

2.6. Wygląd interfejsu użytkownika

a) Strona główna



b) Strona główna po zalogowaniu




Bookshop

Moje zamówienia

Moje dane

Wyloguj się



Podaj tytuł książki...

Szukaj

Gatunek

☐ Fantastyka (52)

☐ Science fiction (4)

☐ Nauka (8)


więcej

Autor:

Wydawca:


Data wydania:

Cena:




Andrzej Sapkowski,
Wiedźmin. Ostatnie
życzenie

24.99 zł



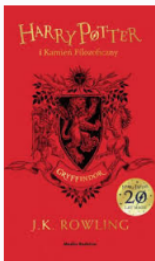
J. K. Rowling, Harry
Potter i Komnata
Tajemnic

24.99 zł




J. K. Rowling, Harry
Potter i Insignia
Śmierci

26.99 zł




J. K. Rowling, Harry
Potter i Kamień
Filozoficzny

24.99 zł




J. K. Rowling, Harry
Potter i Czara Ognia

24.99 zł




Andrzej Sapkowski,
Wiedźmin. Czas
pogardy

29.99 zł



Marcin Guliwer,
Kinetyka spalania
węgla

22.99 zł



J. K. Rowling, Harry
Potter i Więzień
Azkabanu

27.99 zł

<<

<

1

2

3


...

16

>

>>


c) Szczegóły książki




Bookshop

Zarejestruj się

Zaloguj się





Wiedźmin. Ostatnie życzenie

Autor Andrzej Sapkowski

Wydawnictwo SUPERNOWA-Niezależna Oficyna
Wydawnicza NOWA

Data wydania: 2014-10-06


Liczba stron 332

Dodaj do koszyka


24.99 zł

Zbiór opowiadań Andrzeja Sapkowskiego stanowiący wstęp do cyklu wiedźmińskiego opowiadającego o perypetiach Geralta z Rivii.

[Więcej szczegółów](#)

 Wróć do przeglądania

d) Wszystkie szczegóły książki



Wiedźmin. Ostatnie życzenie

Autor: Andrzej Sapkowski

Wydawnictwo: SUPERNOWA-Niezależna Oficyna Wydawnicza NOWA

Data wydania: 2014-10-06


Liczba stron: 332

Dodaj do koszyka 24.99 zł


Zbiór opowiadań Andrzeja Sapkowskiego stanowiący wstęp do cyklu wiedźmińskiego opowiadającego o perypetiach Geralta z Rivii.

[Mniej szczegółów](#)

ISBN	978-83-7578-063-5
Numer wydania	1
Język	polski
Oprawa	miękka
Wymiary	195×24×125

 [Wróć do przeglądania](#)


e) Koszyk



Bookshop

Zarejestruj sięZaloguj się


Koszyk (1)



Wiedźmin. Ostatnie życzenie
Andrzej Sapkowski


- 1 +

24.99 zł




Łączna kwota: 24.99 zł


Złóż zamówienie

 Wróć do strony głównej

f) Składanie zamówienia



Bookshop



Zarejestruj sięZaloguj się

Imię:

Bazy

Nazwisko:

Danych

E-mail:

bazy@mail.pl

Numer telefonu:


012 345 678

Łączna kwota: 24.99 zł


Złóż zamówienie

Użyj swoich danych


Wyczyść formularz

 Wróć do strony głównej

g) Logowanie



Bookshop



Logowanie

Nazwa użytkownika:


Hasło:

[Zapomniałem hasła](#)


[Nie masz jeszcze konta? Zarejestruj się](#)

[← Wróć do strony głównej](#)

h) Rejestracja



Bookshop




Rejestracja

Imię:	<input type="text" value="Bazy"/>	Nazwisko:	<input type="text" value="Danych"/>
Nazwa użytkownika:	<input type="text" value="bazy_danych"/>	E-mail:	<input type="text" value="bazy@mail.pl"/>
Hasło:	<input type="password" value="*****"/>	Numer telefonu:	<input type="text" value="012 345 678"/>
Powtórz hasło:	<input type="password" value="*****"/>		

[Masz już konto? Zaloguj się](#)

[← Wróć do strony głównej](#)


i) Dane klienta



Bookshop

Moje zamówienia


Wyloguj się


0


Twoje dane

Imię:	Bazy	Nazwisko:	Danych
Nazwa użytkownika:	bazy_danych	E-mail:	bazy@mail.pl
Hasło:	*****	Numer telefonu	012 345 678
Powtórz hasło:	*****		

Edytuj

 Wróć do strony głównej


j) Modyfikacja własnych danych przez klienta



Bookshop


Moje zamówienia

Wyloguj się




Twoje dane

Imię:	<input type="text" value="Bazy"/>	Nazwisko:	<input type="text" value="Danych"/>
Nazwa użytkownika:	<input type="text" value="bazy_danych"/>	E-mail:	<input type="text" value="bazy@mail.pl"/>
Hasło:	<input type="password" value="*****"/>	Numer telefonu	<input type="text" value="012 345 678"/>
Powtórz hasło:	<input type="password" value="*****"/>		

 Wróć do strony głównej


k) Zamówienia klienta



Bookshop

Moje daneWyloguj się

Zamówienie nr 1



Wiedźmin. Ostatnie życzenie

Andrzej Sapkowski

x

1

24.99 zł

Data utworzenia: 06.11.2022


Łączna kwota: 24.99 zł

Status zamówienia:

przyjęte

Wycofaj

Zamówienie nr 2



Harry Potter i Komnata Tajemnic

J.K. Rowling

x

1

24.99 zł

Data utworzenia: 06.11.2022

Łączna kwota: 24.99 zł

Status zamówienia:


w trakcie realizacji

brak możliwości wycofania

←

Wróć do strony głównej

1) Strona główna pracownika



[Bookshop](#)
[Zarządzaj zamówieniami](#)
[Zarządzaj magazynem](#)
[Wyloguj się](#)

Gatunek

☐ Fantastyka (52)

☐ Science fiction (4)

☐ Nauka (8)

[więcej](#)

Autor:









Wydawca:

Data wydania:


Cena:

Dostępne: ☒

Liczba sztuk:

 <p>Andrzej Sapkowski, Wiedźmin. Ostateczne Życzenie</p> <p>24.99 zł</p>	 <p>J. K. Rowling, Harry Potter i Komnata Tajemnic</p> <p>24.99 zł</p>	 <p>J. K. Rowling, Harry Potter i Insygnia Śmierci</p> <p>26.99 zł</p>	 <p>J. K. Rowling, Harry Potter i Kamień Filozoficzny</p> <p>24.99 zł</p>
 <p>J. K. Rowling, Harry Potter i Czara Ognia</p> <p>24.99 zł</p>	 <p>Andrzej Sapkowski, Wiedźmin. Czas pogardy</p> <p>29.99 zł</p>	 <p>Marcin Guliwer, Kinetyka spalania węgla</p> <p>22.99 zł</p>	 <p>J. K. Rowling, Harry Potter i Więzień Azkabanu</p> <p>27.99 zł</p>


m) Modyfikacja zamówienia przez pracownika



Bookshop

Wyloguj się

Zamówienie nr 1



Wiedźmin. Ostatnie życzenie

Andrzej Sapkowski

x

1

24.99 zł

Data utworzenia: 06.11.2022

Łączna kwota: 24.99 zł

Status zamówienia:

w trakcie realizacji

▼


Dane klienta

Imię	Baza
Nazwisko	Danych
E-mail	baza@mail.pl
Numer telefonu	012 345 678

←

Wróć do przeglądania

n) Modyfikacja danych książki przez pracownika



Edytuj zdjęcie

Wiedźmin. Ostatnie życzenie

Autor:

Wydawnictwo:

Data wydania:

Liczba stron:

Cena:

Dostępne: ☒

Opis

Zbiór opowiadań Andrzeja Sapkowskiego stanowiący wstęp do cyklu wiedeńskiego opowiadającego o przygodach Geralt z Rivii.

ISBN:

Numer wydania:

Język: ▼

Oprawa: ▼


Wymiary:

Liczba sztuk:

← Wróć do przeglądania

Zatwierdź zmiany

o) Modyfikacja zamówienia przez pracownika



Bookshop

Wyloguj się

Wyszukiwanie zamówień

Dane klienta

Imię

Wpisz imię...

Nazwisko

Wpisz nazwisko...

E-mail

Wpisz imię...

Numer telefonu

Wpisz nr. tel.

Dane zamówienia

Data utworzenia

Od

Do

Status

Status

▼

Łączna kwota

Od

Do

Szukaj

	A	B	C	D	E	F	G
1	Imię	Nazwisko	Data złożenia	Liczba produktów	Cena	Status	Email
2	Jan	Kowalski	03.03.2022	14	227.99	W przygotowaniu	jan.kowalski@wp.pl
3	Adam	Nowak	03.03.2022	12	204.99	Do odbioru	adam.nowak@gmail.com

<<

<

1

2

3

...

16

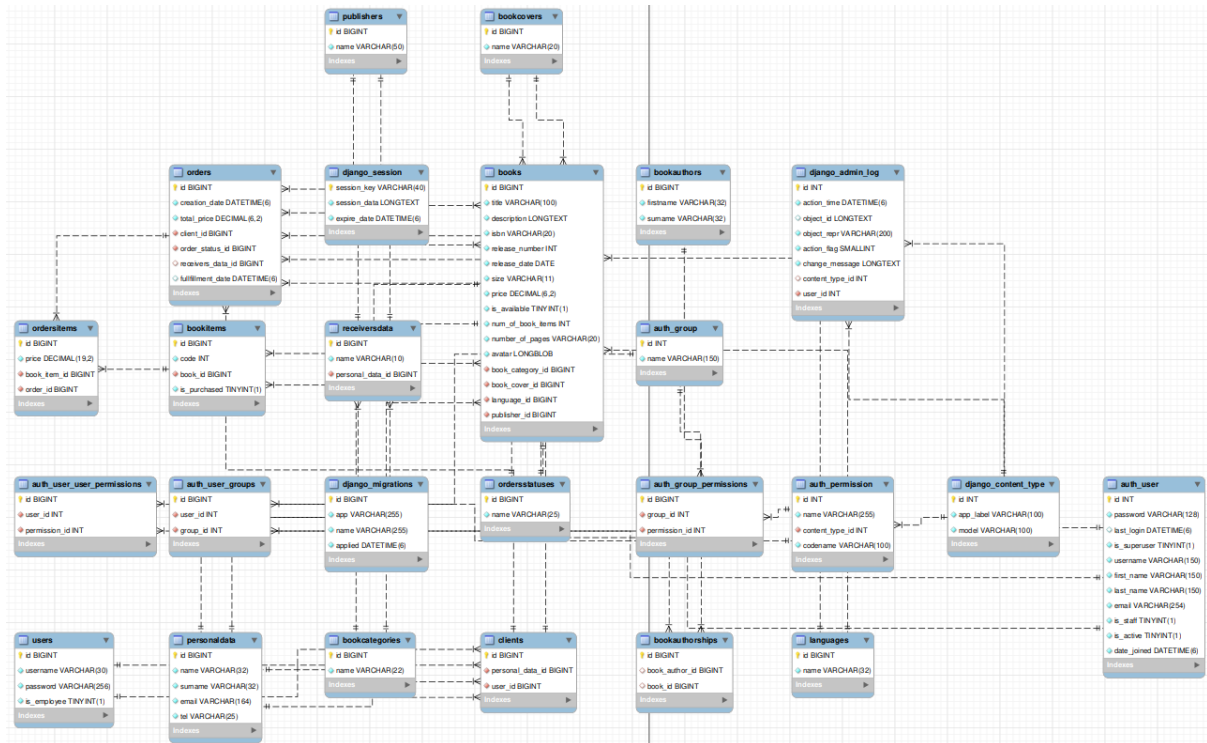
>

>>

3. Implementacja

3.1. Baza danych

Bazę danych stworzyliśmy automatycznie przy pomocy mapowania obiektowo-relacyjnego. Schemat bazy danych po implementacji:



W celu zapewnienia bezpieczeństwa dostępu do bazy przez serwer nr 2 został stworzony użytkownik, który ma uprawnienia tylko do wykonywania zapytań SELECT na 7 tabelach z budowanego systemu oraz pozostałych tabelach stworzonych automatycznie przez framework django. Użytkownik został stworzony przy pomocy komendy create user 'unlogged_person' identified by '5Dn%sfr%Jth6'. Uprawnienia nadane użytkownikowi ilustrują poniższe komendy:

```
mysql> GRANT SELECT ON BOOKS TO unlogged_person;
Query OK, 0 rows affected (0,14 sec)

mysql> GRANT SELECT ON BOOKAUTHORSHIPS TO unlogged_person;
Query OK, 0 rows affected (0,13 sec)

mysql> GRANT SELECT ON BOOKAUTHORS TO unlogged_person;
Query OK, 0 rows affected (0,12 sec)

mysql> GRANT SELECT ON LANGUAGES TO unlogged_person;
Query OK, 0 rows affected (0,17 sec)

mysql> GRANT SELECT ON BOOKCOVERS TO unlogged_person;
Query OK, 0 rows affected (0,24 sec)
```

```
mysql> GRANT SELECT ON PUBLISHERS TO unsigned_person;  
Query OK, 0 rows affected (0,09 sec)
```

```
mysql> GRANT SELECT ON BOOKCATEGORIES TO unsigned_person;  
Query OK, 0 rows affected (0,07 sec)
```

```
mysql> GRANT ALL ON auth_group TO UNLOGGED_USER;  
ERROR 1410 (42000): You are not allowed to create a user with GRANT  
mysql> GRANT ALL ON auth_group TO UNLOGGED_PERSON;  
ERROR 1410 (42000): You are not allowed to create a user with GRANT  
mysql> GRANT ALL ON auth_group TO unsigned_person;  
Query OK, 0 rows affected (0,10 sec)
```

```
mysql> GRANT ALL ON auth_group_permissions TO unsigned_person;  
Query OK, 0 rows affected (0,17 sec)
```

```
mysql> GRANT ALL ON auth_permission TO unsigned_person;  
Query OK, 0 rows affected (0,13 sec)
```

```
mysql> GRANT ALL ON auth_user TO unsigned_person;  
Query OK, 0 rows affected (0,08 sec)
```

```
mysql> GRANT ALL ON auth_user_groups TO unsigned_person;  
Query OK, 0 rows affected (0,21 sec)
```

```
mysql> GRANT ALL ON auth_user_user_permissions TO unsigned_person;  
Query OK, 0 rows affected (0,10 sec)
```

```
mysql> GRANT ALL ON django_admin_log TO unsigned_person;  
Query OK, 0 rows affected (0,06 sec)
```

```
mysql> GRANT ALL ON django_content_type TO unsigned_person;  
Query OK, 0 rows affected (0,12 sec)
```

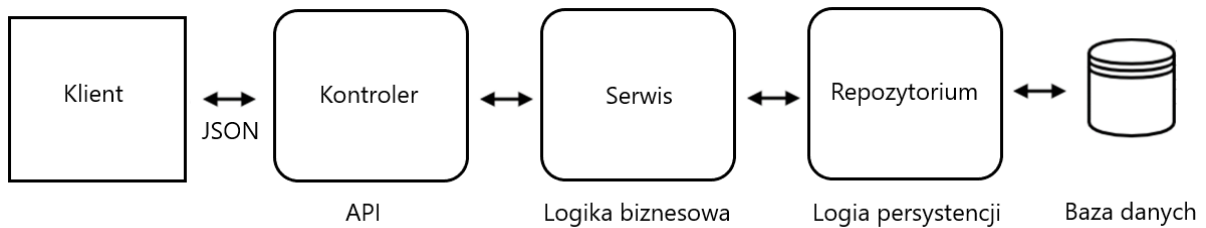
```
mysql> GRANT ALL ON django_migrations TO unsigned_person;  
Query OK, 0 rows affected (0,18 sec)
```

```
mysql> GRANT ALL ON django_session TO unsigned_person;  
Query OK, 0 rows affected (0,12 sec)
```

3.2. Aplikacja serwerowa 1

Aplikacja serwerowa 1 zajmuje się obsługą większości funkcji systemu.

Do skorzystania z tych funkcji wymagane jest zalogowanie się użytkownika. Aplikacja korzysta z mapowania obiektowo-relacyjnego (ORM). Architektura aplikacji jest następująca:



Baza danych jest to warstwa odpowiedzialna za przechowywanie i zarządzanie danymi.

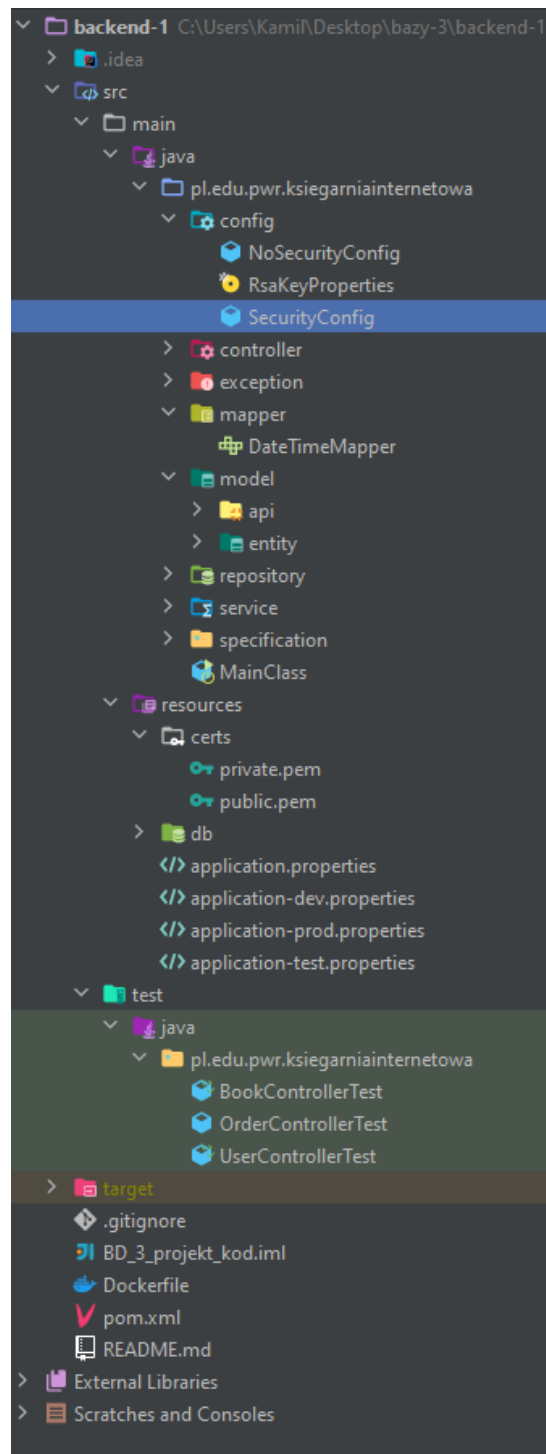
Repozytorium jest ściśle powiązane z bazą danych. Repozytorium operuje na encjach (modelach), które są odpowiednio oznaczonymi klasami Javy będącymi obiektowo-relacyjną reprezentacją tabel w bazie danych. Repozytorium nakłada na bazę danych abstrakcję (logikę persystencji), dzięki której programista może operować na obiektach Javy zamiast ręcznego pisania zapytań do bazy danych. Wykorzystany ORM po stronie Javy pozwala nawet na generowanie metod repozytoriów na podstawie jedynie deklaracji tych metod i jest to możliwe poprzez odpowiednie nazywanie tych metod np. metoda `findByName()` zwróci encję o podanej nazwie.

Serwisy są ściśle powiązane z repozytoriami. Warstwa ta stanowi logikę biznesową aplikacji. Serwisy są odpowiednio oznaczonymi klasami Javy, których metody realizują poszczególne funkcje systemu od strony backendu.

Warstwa kontrolera odpowiada za udostępnianie klientowi usług (np. złożenie zamówienie) dostarczanych przez serwisy. Kontroler udostępnia grupę endpointów, z których każdy z nich dostarcza jedną z usług. Klient może skorzystać z danej usługi poprzez wysłanie żądania na endpoint powiązany z tą usługą.

Warstwa kliencka jest to aplikacja kliencka, która wysyła żądania do kontrolera, w celu skorzystania z udostępnionych przez niego usług.

Struktura aplikacji jest zgodna z przedstawioną architekturą warstwową.



Przykładowa encja książek:

```
@Builder
@Getter
@Setter
@Entity
@NoArgsConstructor
@AllArgsConstructor
```

```

@Table(name = "BOOKS")
public class BookEntity {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(name = "title", length = 100, nullable = false)
    private String title;

    @Lob
    @Column(name = "description", columnDefinition = "LONGTEXT")
    private String description;

    @Column(name = "isbn", length = 20, unique = true)
    private String isbn;

    ...

    @Lob
    @Type(type="org.hibernate.type.ImageType")
    @Column(name = "avatar")
    private byte[] avatar;

    @JsonIgnoreProperties(value = {"id", "bookEntity"})
    @OneToMany(mappedBy = "bookEntity")
    private Set<BookAutorshipEntity> authors = new HashSet<>();

    ...

    @JsonIgnoreProperties("id")
    @ManyToOne
    @JoinColumn(name = "book_category_id")
    private BookCategoryEntity category;
}

```

Użycie adnotacji `@Column` pozwala na wygenerowanie encji o atrybutach odpowiadających wskazanym kolumnom tabeli po stronie bazy danych.

Przykładowe repozytorium książek:

```
@Repository
public interface BookRepository extends JpaRepository<BookEntity, Long>
{
}
```

Repozytoria zawierają powiązanie encji z jej kluczem podstawowym (w tym przypadku jest to powiązanie encji książki z jej kluczem podstawowym typu `Long`). Dziedziczenie interfejsu po interfejsie `JpaRepository` pozwala na automatyczne wygenerowanie kilku podstawowych metod do zarządzania danymi np. `findAll`, `findById`, `save`, czy `deleteById`. Możliwe jest również zadeklarowanie własnych metod i nawet po spełnieniu konwencji nazewniczej wygenerowanie ciał metod na podstawie jedynie nazwy zadeklarowanej metody.

Przykładowy serwis książek:

```
@Service
@RequiredArgsConstructor
public class BookServiceImpl implements BookService {

    private final BookRepository bookRepository;

    @Override
    public boolean existsById(Long bookId) {

        return bookRepository.existsById(bookId);
    }

    @Override
    public BookEntity getBookById(Long bookId) {
        Optional<BookEntity> bookEntityOpt =
bookRepository.findById(bookId);

        if(bookEntityOpt.isEmpty()){
            throw new EntityNotFoundException("Nie istnieje książka o
takim id");
        }
        // Optional, zeby mozna bylo ewentualnie wyrzucac wyjatki

        return bookEntityOpt.get();
    }
}
```

```

@Override
public List<BookEntity> getByIdList(List<Long> ids) {

    return bookRepository.findAllById(ids);
}
}

```

Zaimplementowano w warstwie serwisów obsługę błędów. W przypadku otrzymania błędnych danych, rzucane są odpowiednie wyjątki z wiadomościami zwrotnymi, które są później obsługiwane przez kontrolery.

Przykładowy kontroler książek:

```

@RestController
@CrossOrigin(origins = {"http://localhost:3000",
"http://localhost:8000", "https://booksshop-dr.azurewebsites.net"})
@RequestMapping(value = "/v1")
@RequiredArgsConstructor
public class BookController {

    private final BookService bookService;

    @GetMapping(value = "/book/{id}")
    public ResponseEntity getBookById(@PathVariable("id") String
bookIdStr) {

        Long bookId;

        try{
            bookId = Long.valueOf(bookIdStr);
        }
        catch(NumberFormatException e){
            return
ResponseEntity.status(HttpStatus.BAD_REQUEST).body("Podano
nieprawidłowe id książki");
        }

        BookEntity bookEntity;

        try{
            bookEntity = bookService.getBookById(bookId);
        }
        catch(EntityNotFoundException e){

```

```

        return
        ResponseEntity.status(HttpStatus.NOT_FOUND).body(e.getMessage());
    }

    return ResponseEntity.ok(bookEntity);
}

@GetMapping(value = "/books/by-Ids")
public ResponseEntity<List<BookEntity>>
getBooksByIds(@RequestParam("bookIds") List<Long> bookIds){

    List<BookEntity> foundBooks = bookService.getByIdList(bookIds);

    return ResponseEntity.ok(foundBooks);
}
}

```

Założono, że dostęp do endpointów wszystkich kontrolerów będzie możliwy jedynie dla klienta lokalnego i produkcyjnego. W warstwie kontrolerów została zaimplementowana obsługa błędów (wyjątków) rzucanych przez serwisy. W przypadku otrzymania błędu zwracany jest status http odpowiadający temu błędowi wraz z wiadomością otrzymaną z serwisu.

3.3. Aplikacja serwerowa 2

Aplikacja serwerowa 2 zajmuje się obsługą pięciu endpointów, które nie wymagają zalogowania się użytkowników. Jednym z endpointów jest zwrócenie zawartości całej strony użytkownikowi (endpoint nr 1). Poniżej przedstawione są wszystkie endpointy aplikacji.

```

urlpatterns = [
    path('', views.return_page),
    path('books/all', views.get_start_page_data),
    path('books/find/', views.find_books),
    path('book/<int:id>', views.get_book),
    path('<path:resource>', views.return_page2),
]

```

Endpoint nr 2 służy do pobrania zawartości strony głównej, czyli informacji o wszystkich dostępnych książkach oraz wszystkich kategoriach książek, które są dostępne w sklepie. Przez parametry zapytania podawane są parametry wyszukiwania, aby po odświeżeniu strony można było powrócić do poprzednich wyników wyszukiwania. Endpoint nr 4 służy do pobrania danych szczegółowych książki. W ścieżce podawane jest id książki. Ostatni

endpoint zwraca stronę główną w przypadku, gdy podana ścieżka nie pasuje do poprzednich endpointów.

Do wykonywania wszelkich zapytań do bazy danych wykorzystywane jest mapowanie obiektowo-relacyjne. Django chroni przed wstrzykiwaniem SQL, używając parametryzacji zapytań. W warstwie ORM Django definiuje zapytania SQL oddzielone od parametrów zapytania, a sterownik bazy danych jest odpowiedzialny za unikanie każdego z parametrów. Połączenie z bazą danych odbywa się przy pomocy protokołu TLS z użyciem certyfikatu. Aplikacja serwerowa 2 może tylko wykonywać zapytanie SELECT na kilku tabelach (stworzony użytkownik w bazie danych). Poniżej znajduje się kod funkcji odpowiedzialnej za wyszukiwanie książek według kryteriów.

```
def find_books(title1: str, genrel: list, author_firstname: str,
author_surname:str, publisher1: str, release1: str, min1: str, max1:
str):
    all_entries = None
    dict1 = {}
    dict1['is_available'] = 1
    if title1 is not None:
        dict1['title__icontains'] = title1
    if release1 is not None:
        format = '%Y-%m-%d'
        try:
            date = datetime.datetime.strptime(release1, format).date()
        except ValueError:
            return ""
        dict1['release_date__exact'] = date
    if min1 is not None:
        min2 = float(min1)
        dict1['price__gte'] = min2
    if max1 is not None:
        max2 = float(max1)
        dict1['price__lte'] = max2

    all_entries = Books.objects.filter(**dict1).select_related('book_category','publisher')

    list1 = []
    for i in all_entries:
        good = True
        if genrel is not None:
            if i.book_category.name not in genrel:
                good = False
        if good is True and publisher1 is not None:
            if publisher1 != i.publisher.name:
```

```

        good = False
    if good is True:
        if author_firstname is not None:
            authors = BookAuthorships.objects.select_related('book_author').filter(book = i.id)

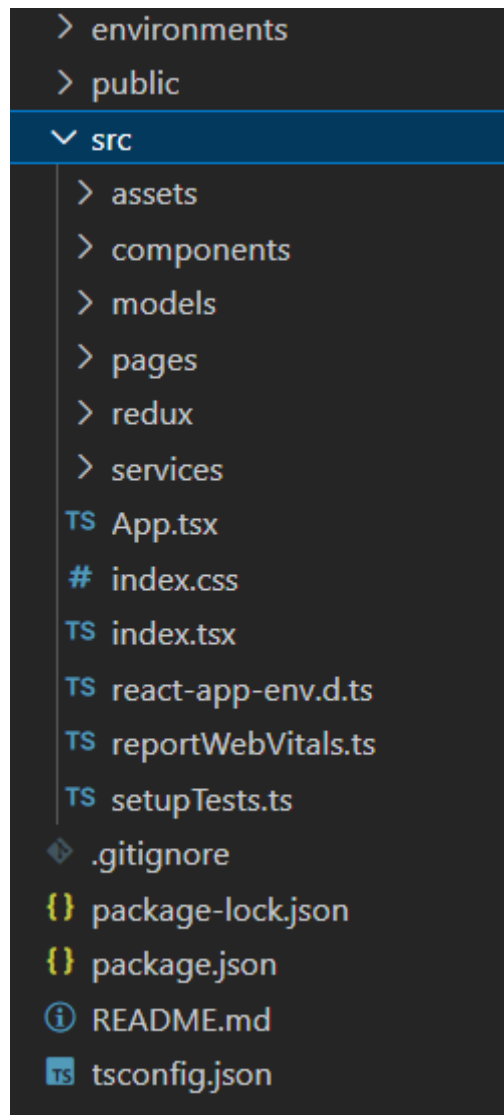
            contains = False
            for j in authors:
                if j.book_author.firstname == author_firstname and j.book_author.surname == author_surname:
                    contains = True
                    break
            if contains is True:
                serializer = BooksSerializer(i)
                list1.append(serializer.data)
            else:
                serializer = BooksSerializer(i)
                list1.append(serializer.data)
    return json.dumps(list1)

```

W funkcji tej parametry wejściowe mają określony typ (str). Parametry te są walidowane. Zapytania do bazy wykonywane są przy użyciu ORM, z wykorzystaniem funkcji takich jak `select_related()` oraz `filter()`. Rekordy otrzymane z bazy danych są serializowane przy pomocy stworzonych serializerów. Lista zserializowanych obiektów jest konwertowana do formatu json.

3.4. Frontend strony

Struktura aplikacji frontendowej jest następująca:



Struktura ta ma następujące cechy:

- assety to obrazki,
- serwisy odpowiadają za wysyłanie żądań do backendu,
- modele to interfejsy odpowiadające otrzymanych przez backend danych,
- komponenty określają widok i logikę strony,
- pages to główne komponenty stanowiące osobne podstrony,
- w redux znajdują się slice'y, które pozwalają na przechowywanie globalnego stanu aplikacji.

4. Testowanie

4.1. Baza danych

Sprawdziliśmy poprawność nadanych uprawnień, zgodność z wymaganiami oraz integralność bazy danych.

4.2. Aplikacja serwerowa 1

Dla aplikacji serwerowej 1 przeprowadzono akceptacyjne testy integracyjne. Testy te polegały na wysyłaniu odpowiednio spreparowanych żądań na poszczególne endpointy kontrolerów i następnie sprawdzaniu otrzymanych odpowiedzi. Przeprowadzono zarówno testy sprawdzające, czy dane funkcjonalności działają, jak i sprawdzające, czy zostanie zwrócony odpowiedni komunikat błędu w przypadku podania niepoprawnych danych.

Test sprawdzający, czy można pobrać książkę po jej identyfikatorze:

```
@Test
public void shouldBookById() {

    BookEntity book = bookEntities.get(1);

    Response response = given()
        .when()
        .get(url + "/book/" + book.getId());

    response.then().statusCode(HttpStatus.OK.value());

    String responseStr = response.asString();

    //BookEntity gotBook = response.as(BookEntity.class);

    assertTrue(responseStr.contains("\"id\":\"" + book.getId()));
}
```

Test sprawdzający, czy niemożliwe jest pobranie książki po nieistniejącym identyfikatorze:

```
@Test
public void shouldNotGetBookByNotExistingId() {

    Response response = given()
        .when()
        .get(url + "/book/20");

    response.then().statusCode(HttpStatus.NOT_FOUND.value());
}
```

Test sprawdzający, czy niemożliwe jest pobranie książki po niepoprawnym identyfikatorze:

```

@Test
public void shouldNotGetBookByInvalidId() {

    Response response = given()
        .when()
        .get(url + "/book/a");

    response.then().statusCode(HttpStatus.BAD_REQUEST.value());
}

```

Test sprawdzający, czy możliwe jest pobranie listy książek po liście ich identyfikatorów:

```

@Test
public void shouldGetBooksWhichIdIsInIds() {

    Response response = given()
        .queryParams("bookIds", List.of(1, 2, 3))
        .when()
        .get(url + "/books/by-Ids");

    String strBooks = response.asString();

    System.out.println(strBooks);

    response.then().statusCode(HttpStatus.OK.value());

    assertTrue(strBooks.contains("\"id\":1"));
    assertTrue(strBooks.contains("\"id\":2"));
    assertTrue(strBooks.contains("\"id\":3"));
}

```

Test sprawdzający, czy możliwe jest zarejestrowanie się użytkownika:

```

@Test
public void shouldRegisterUser() {

    Response response = given()
        .when()
        .contentType(MediaType.APPLICATION_JSON.toString())
        .body(
            RegistrationData.builder()

```

```

        .username("kamil")
        .password(("dywan").toCharArray())
        .name("Kamil")
        .surname("Dywan")
        .email("kamil@mail.com")
        .tel("000111222")
        .build()
    )
    .post(url + "/register");

assertTrue(userRepository.findById(7L).get().getUsername().equals("kamil"));

assertTrue(personalDataRepository.findById(1L).get().getName().equals("Kamil"));

assertTrue(clientRepository.findById(1L).get().getUserEntity().getUsername().equals("kamil"));
}

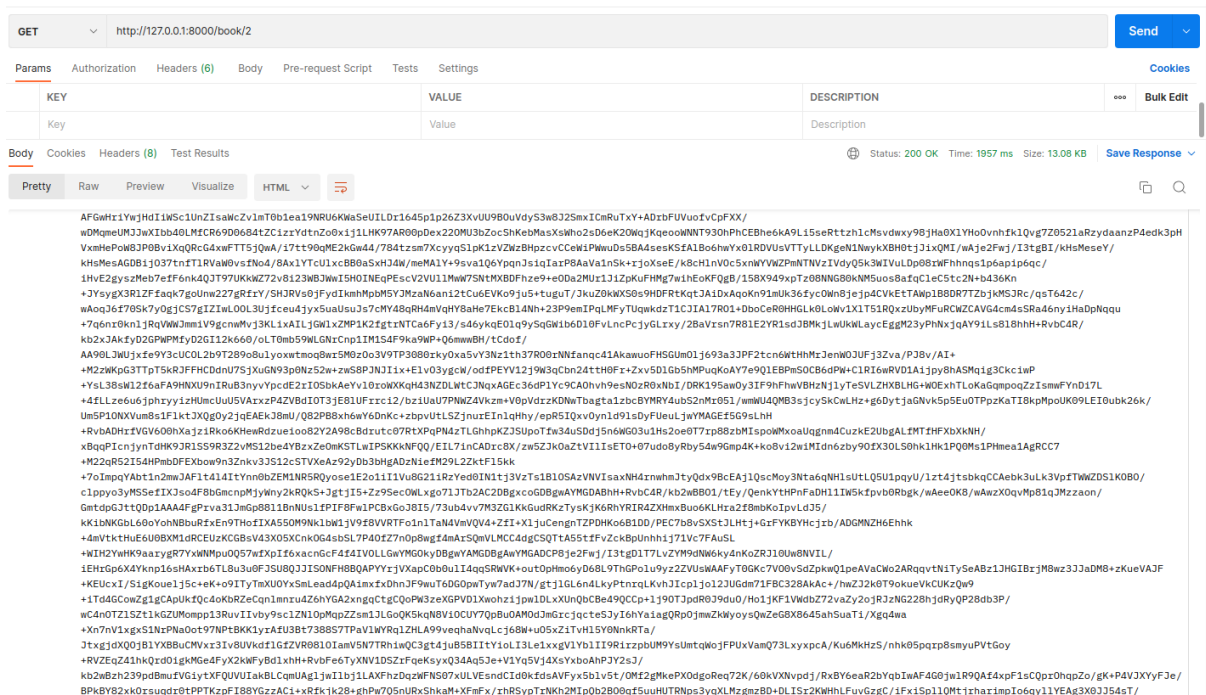
```

Wszystkie testy zakończyły się powodzeniem.

4.3. Aplikacja serwerowa 2

Aplikacja serwerowa nr 2 została przetestowana przy pomocy programu Postman. Zostało wykonane kilkanaście testów dla każdego endpointu. Poniżej znajdują się zrzuty ekranu z wyników niektórych przypadków testowych. Wszystkie testy wykazały poprawność działania aplikacji.

Wyszukiwanie książki - test



4.4. Frontend

Frontend aplikacji przetestowaliśmy przy pomocy testów manualnych.

5. Wdrożenie

Do wdrożenia wszystkich elementów systemu wykorzystano platformę Azure. Zarówno aplikację serwerową 1, jak i aplikację serwerową 2 wdrożono przy użyciu kontenerów oraz App Service z poziomu Azure.

5.1. Baza danych

Nazwa serwera	: bookshop.mysql.database.azure.com
Nazwa logowania admini...	: bazy
Konfiguracja	: <u>Z możliwością zwiększania szybkości, B1ms, rdzenie virtualne: 1,...</u>
Wersja MySQL	: 8.0
Strefa dostępności	: 3
Utworzono w dniu	: 2022-11-15 16:44:14.0360091 UTC

5.2. Aplikacja serwerowa 1

Adres URL : <https://backend-1-api.azurewebsites.net>
Plan usługi App Service : [bazy3-front \(B1: 1\)](#)
System operacyjny : Linux
Sprawdzanie kondycji : [Nieskonfigurowane](#)

5.3. Aplikacja serwerowa 2

Adres URL : <https://booksshop-dr.azurewebsites.net>
Plan usługi App Service : [bazy3-front \(B1: 1\)](#)
System operacyjny : Linux
Sprawdzanie kondycji : [Nieskonfigurowane](#)

6. Podsumowanie

Aplikacja została zrealizowana w 90 procentach. Większość przypadków użycia udało się wykonać. Nie udało się w całości wykonać logowania (obecnie logowanie jest zmockowane) oraz niektórych przypadków użycia pracownika. Aplikacja w całości została wdrożona w serwisie Azure. Działa pod linkiem <https://booksshop-dr.azurewebsites.net/>. Aplikację przetestowaliśmy przy pomocy testów jednostkowych i akceptacyjnych. Wszystkie testy wykazały poprawność działania aplikacji.

7. Bibliografia

- <https://www.stackhawk.com/blog/sql-injection-prevention-django/>
- <https://www.djangoproject.com/>