****

**KOLEGIUM INFORMATYKI STOSOWANEJ**

**Kierunek: INFORMATYKA**

**Specjalność: Technologie internetowe i mobilne**

Kinga Pszeniczna

Numer albumu studenta: 57122

***Internetowy zbiór książek***

Promotor: dr inż. Łukasz Piątek

**PRACA DYPLOMOWA INŻYNIERSKA**

**Rzeszów 2020**

Ja niżej podpisany/a oświadczam, że składana przeze mnie praca dyplomowa pt. „Internetowy zbiór książek” została przygotowana samodzielnie.

Oświadczam również, że praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu zawodowego w wyższej uczelni.

Oświadczam ponadto, że niniejsza wersja pracy jest identyczna ze złożoną wersją elektroniczną.

.........................................................

data czytelny podpis autora

Oświadczam, że niniejsza praca została przygotowana pod moim kierunkiem i stwierdzam, że spełnia ona warunki do przedstawienia jej w postępowaniu o nadanie tytułu zawodowego.

................................................................

data czytelny podpis promotora

Spis treści

[Wstęp 5](#_Toc34042984)

[Cel pracy 5](#_Toc34042985)

[Zakres pracy 5](#_Toc34042986)

[Aktualny stan wiedzy 5](#_Toc34042987)

[1. Wprowadzenie do aplikacji internetowych 7](#_Toc34042988)

[1.1. Wprowadzenie 7](#_Toc34042989)

[1.2. Rest API 9](#_Toc34042990)

[1.3. Entity Framework 9](#_Toc34042991)

[2. Wybrane technologie do tworzenia serwisów internetowych 11](#_Toc34042992)

[2.1. Języki programowania 11](#_Toc34042993)

[2.1.1. C# 11](#_Toc34042994)

[2.1.2. Java 11](#_Toc34042995)

[2.1.3. JavaScript – REACT oraz Angular 11](#_Toc34042996)

[2.1.4. PHP 12](#_Toc34042997)

[2.2. Środowiska programistyczne 12](#_Toc34042998)

[2.2.1. Visual Studio 12](#_Toc34042999)

[2.2.2. Visual Studio Code 13](#_Toc34043000)

[2.2.3. Eclipse 13](#_Toc34043001)

[2.3. Systemy baz danych 14](#_Toc34043002)

[2.3.1. MS SQL Server 14](#_Toc34043003)

[2.3.2. MySQL 14](#_Toc34043004)

[2.3.3. Bazy nierelacyjne 15](#_Toc34043005)

[3. Aktualny stan wiedzy 17](#_Toc34043006)

[3.1. Wprowadzenie 17](#_Toc34043007)

[3.2. Wykorzystane technologie 18](#_Toc34043008)

[3.3. Wybrane aplikacje internetowe związane z tematyką pracy 19](#_Toc34043009)

[4. Specyfikacja systemu 21](#_Toc34043010)

[4.1. Wymagania funkcjonalne i niefunkcjonalne 21](#_Toc34043011)

[4.2. Opis poszczególnych paneli systemu 21](#_Toc34043012)

[5. Wybrane funkcjonalności Internetowego zbioru książek 23](#_Toc34043013)

[5.1. Model 23](#_Toc34043014)

[5.2. Widok 25](#_Toc34043015)

[5.3. Kontroler 28](#_Toc34043016)

[6. System Internetowego zbioru książek 30](#_Toc34043017)

[6.1. Panel administracyjny 30](#_Toc34043018)

[6.2. Panel użytkownika 31](#_Toc34043019)

[Streszczenie 38](#_Toc34043020)

[Literatura 39](#_Toc34043021)

[Spis stron WWW 39](#_Toc34043022)

[Spis rysunków 40](#_Toc34043023)

[Załączniki 42](#_Toc34043024)

# **Wstęp**

Aktualnie popularność czytania książek jest wypierana przez rozrywkę cyfrową, taką jak gry, filmy, seriale. W Polsce potencjalny czytelnik woli zasięgnąć do artykułów w Internecie czy telewizji. Jeśli natomiast czytelnik sięgnie po książkę, to coraz częściej jest to wersja elektroniczna.

# **Cel pracy**

Celem niniejszej pracy dyplomowej będzie opracowanie, a następnie implementacja systemu Internetowego zbioru książek. Witryna internetowa umożliwia zalogowanym użytkownikom, przeglądanie informacji o autorach i książkach, oraz daje możliwość pobierania książki w formacie \*.pdf. Z utworzonej aplikacji można korzystać za pomocą przeglądarki internetowej na komputerze, bądź na smartfonie. Uruchomienie aplikacji na smartfonie jest możliwe dzięki responsywności aplikacji. W stworzonej aplikacji zostały umieszczone książki wolne od praw autorskich.

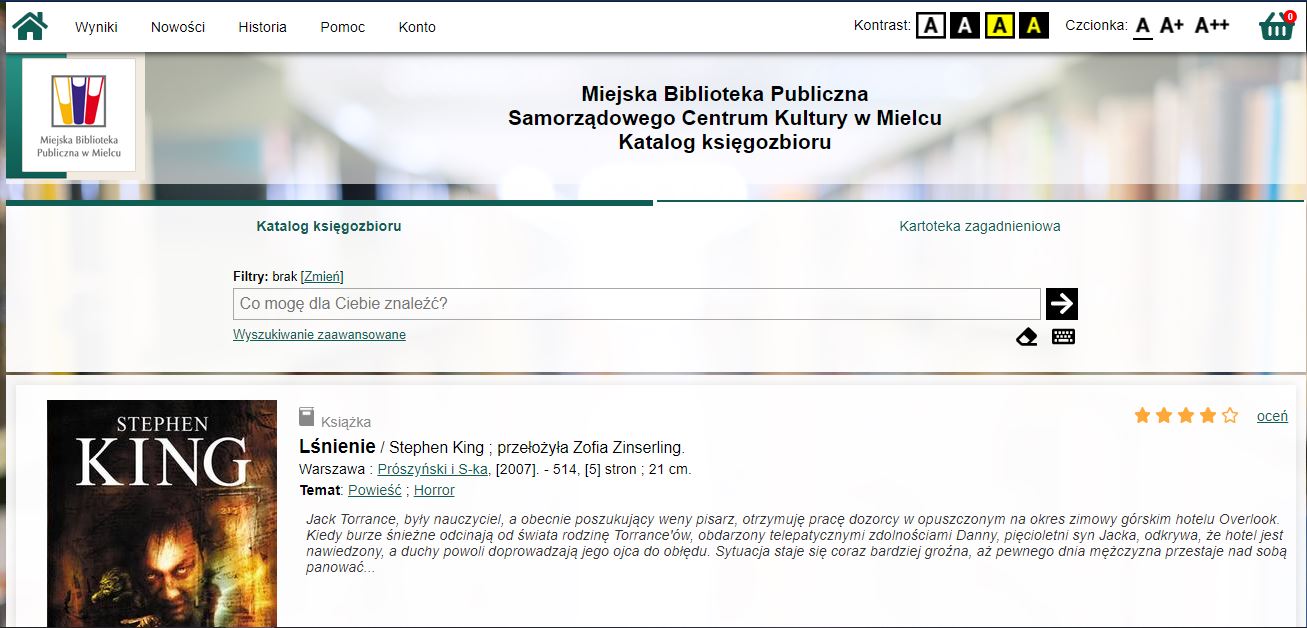
# **Zakres pracy**

Aplikacja internetowa została stworzona z wykorzystaniem technologii ***ASP .NET MVC 5***, w połączeniu z ***Entity Framework***. Technologia ***ASP .NET*** przeznaczona jest do tworzenia aplikacji internetowych. Wykorzystywane są w tej technologii języki ***C#***, ***HTML***, ***JavaScript***, ***CSS***. Aby strona była responsywna, czyli aby można ją było uruchomić w przeglądarce na smartfonie, wykorzystany został ***Bootstrap***. Działa on na zasadzie siatki, która dopasowuje widok strony do ekranu, na którym jest wyświetlona strona. Wzorzec ***MVC*** wykorzystany podczas tworzenia systemu oznacza ***M***odel – ***V***iew – ***C***ontroller. Wzorzec ten pozwala na organizowanie struktury tworzonej aplikacji, która posiada graficzne interfejsy użytkownika. Wykorzystany do stworzenia systemu ***Entity Framework*** to bardzo popularny ***ORM*** (ang. ***O***bject-***R***elational ***M***apping). Wykorzystuje się go dopisania aplikacji, które zawierają bazę danych. ***Entity Framework*** pozwala patrzeć na bazę danych jak na obiekt klasy. Do przechowywania danych została wykorzystana baza danych ***SQL Server Express***. Jest to system zarządzania bazą danych w wersji darmowej. Oznacza to, że nie jest wersją pełną. Aby zarządzać bazą danych stworzonego systemu, wykorzystane zostało narzędzie ***SQL Server Management Studio 2017***.

Poza wykonaną implementacją systemu, praca zawiera część pisemną podzieloną na dwie części: część teoretyczną oraz praktyczną. Część teoretyczna zawiera cztery rozdziały. W rozdziale pierwszym zawarte jest wprowadzenie do aplikacji internetowych. Rozdział ten zawiera podstawowe informacje o technologiach takich jak ***HTTP, HTML, CSS,*** ***REST API*** oraz ***Entity Framework***. Rozdział drugi zawiera opis wybranych technologii, które umożliwiają tworzenie różnych aplikacji. Przedstawione zostały języki programowania takie jak: ***C#, Java, JavaScript, REACT, Angular*** oraz ***PHP***. Następnie opisane zostały najpopularniejsze środowiska programistyczne, do których należą: ***Visual Studio, Visual Studio Code, Eclipse***. Opisane zostały również systemy bazodanowe, takie, jak: MS ***SQL Server, MySQL*** oraz nie relacyjne bazy danych. Następny rozdział zawiera informacje o aktualnym stanie wiedzy. Rozdział ten zawiera krótkie wprowadzenie, w którym zawarta jest charakterystyka problemu, wykorzystane technologie do napisania aplikacji oraz przedstawione zostały przykładowe aplikacje o tematyce książek. Ostatnim rozdziałem części teoretycznej jest specyfikacja systemu. Zawiera ona wymagania funkcjonalne i niefunkcjonalne utworzonego systemu oraz opis poszczególnych paneli aplikacji. Część praktyczna zawiera dwa rozdziały. Pierwszy rozdział tej części przedstawia wybrane funkcjonalności utworzonej aplikacji z podziałem na Model, Widok i Kontroler. Drugim rozdziałem jest przedstawienie działania utworzonego systemu. Rozdział został podzielony na panel administracyjny oraz panel użytkownika.

# **Aktualny stan wiedzy**

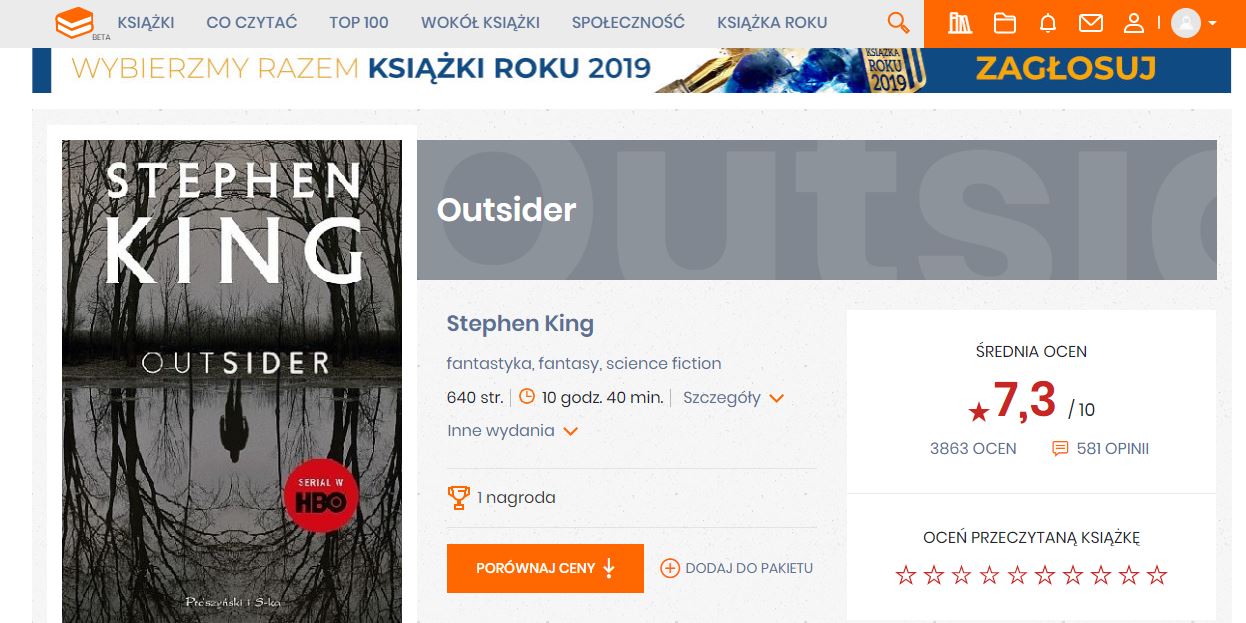
Pomimo wzrastającej liczby książek na rynku, ich czytelność maleje. Ma na to wpływ rozwój technologii. W obecnych czasach ludzie wolą spędzić wolny czas przed ekranem komputera. Aby nadążyć za rozwojem technologii oraz zwrócić uwagę czytelnika, powstało wiele aplikacji, w których można znaleźć różnego rodzaju książki. Takimi aplikacjami mogą być katalogi online bibliotek, strony z możliwością zakupu książek w wersji elektronicznej. Dużo czytelników korzysta również z elektronicznej wersji książki, czyli z ebooków. Poniższy rysunek (Rys. 1) przedstawia widok katalogu online ***Biblioteki Miejskiej w Mielcu***.



Rys. 1 Katalog online Biblioteki Miejskiej w Mielcu

Źródło: opracowanie własne

Rysunek (Rys. 2) poniżej przedstawia widok serwisu ***lubimyczytać.pl***, w którym można wyszukać wybraną książkę oraz sprawdzić ceny w różnych sklepach.



Rys. 2 Widok wybranej książki w serwisie lubimyczytać.pl

Źródło: opracowanie własne

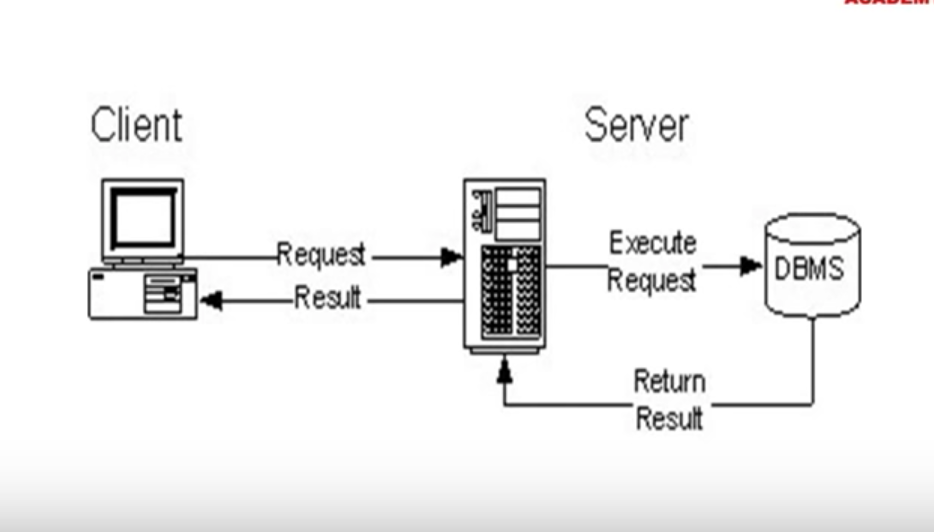
# **Wprowadzenie do aplikacji internetowych**

W niniejszym rozdziale zamieszczone zostaną skrótowe informacje związane z wprowadzeniem do aplikacji internetowych. Przedstawione zostaną takie zagadnienia jak:

* ***HTTP,***
* ***HTML,***
* ***CSS,***
* ***Rest Api***,
* ***Entity Framework***.
  1. **Wprowadzenie**

Protokół ***HTTP*** (ang. ***H***yper***T***ext ***T***ransfer ***P***rotocol) jest używany w przeglądarkach, pozwala on na komunikację klient-serwer. Określa on, w jaki sposób przekazywane i modyfikowane są dane oraz jak ma je odczytać serwer. Protokół ***HTTP*** jest bezstanowy, tzn., że nie przechowuje żadnych danych. Protokół ten jest mało bezpieczny, używane są w nim pliki cookies, które śledzą użytkownika danego serwisu internetowego. Zaleca się używanie protokołu ***HTTPS***, który jest protokołem bezpiecznym.

Poniższy rysunek (Rys. 3) przedstawia schemat działania protokołu ***HTTP***.



Rys. 3 Schemat działania protokołu HTTP

Źródło: <https://medium.com/@rutr1ka/http-https-protocols-http-methods-api-c0d61845e46e>(28.02.2020)

Aby wiedzieć, czym tak naprawdę jest ***ASP .NET***, trzeba wiedzieć też, czym jest ***HTML*** (ang. ***H***yper***T***ext ***M***arkup ***L***anguage). Jest to język opisowy, który za pomocą znaczników określa zawartość strony. ***HTML*** wykorzystuje hiperłącza, obrazki, pola tekstowe, tabele, przyciski, nagłówki, stopki, itp. Określany jest również tytuł strony, opis, skrypty, meta dane (np. język znaków), czy też arkusze stylów (***CSS***). Rysunek poniżej (Rys. 4) przedstawia fragment kodu, w którym znajduje się podstawowa struktura języka ***HTML***.



Rys. 4 Podstawowa struktura kodu języka HTML

Źródło: <https://developer.mozilla.org/pl/docs/Learn/Getting_started_with_the_web/HTML_basics>(28.02.2020)

* <!DOCTYPE html> - oznacza jaki typ przyjmuje pisany dokument
* <html></html> - zawiera całą treść strony, jest elementem bazowym, wskazuje gdzie zaczyna się i kończy kod
* <head></head> - oznacza nagłówek strony, treść zawarta w tym znaczniku nie wyświetla się użytkownikowi, można tutaj zamieścić takie informacje jak: style ***CSS***, meta dane, używane skrypty
* <meta charset=”utf-8”> - ten element oznacza jaki zestaw znaków ma znajdować się na stronie, UTF-8 zawiera znaki z większości języków pisanych
* <title></title> - dzięki temu znacznikowi wyświetlamy tytuł jaki ma się pojawić w karcie przeglądarki
* <body></body> - zawiera całą zawartość strony, która ma być wyświetlona użytkownikowi.

Poniżej znajduje się rysunek (Rys. 5), na którym został przedstawiony znacznik <p>



Rys. 5 Znacznik <p>

Źródło: opracowanie własne

Na rysunku (Rys. 6) poniżej przedstawiony został rezultat uruchomienia strony z tekstem umieszczonym w znaczniku <p>. Znacznik <p> służy do oznaczania tekstu, który chcemy wyświetlić na stronie.

przyklad na wyjsciu

Rys. 6 Rezultat uruchomienia strony z tekstem w znaczniku <p>

Źródło: opracowanie własne

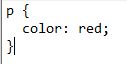
***CSS*** (ang. ***C***ascading ***S***tyle ***S***heets) to kaskadowe arkusze stylów, które odpowiadają za odpowiedni layout strony, która wyświetla się na ekranie. Z pomocą ***CSS*** można ustawić kolor, rozmiar oraz styl czcionki na stronie, tło, wielkość wyświetlanych obrazków, itp. Kaskadowe arkusze stylów można umieszczać w strukturze kodu ***HTML*** w różny sposób. Można to zrobić przy każdym znaczniku z osobna, można umieścić wszystkie selektory w znaczniku <head>, lub można stworzyć nowy plik o rozszerzeniu \*.css. W strukturze kodu ***HTML*** umieszczamy tylko odnośnik do pliku, w którym zawarte będą wszystkie selektory oraz ich parametry. Poniższy rysunek (Rys. 7) przedstawia odnośnik do pliku style.css.



Rys. 7 Odnośnik do pliku style.css

Źródło: opracowanie własne

Na rysunku (Rys. 8) poniżej znajduje się treść pliku style.css.



Rys. 8 Treść pliku style.css

Źródło: opracowanie własne

W przedstawionym powyżej fragmencie kodu, selektor *p* przyjmuje parametr *color* o wartości *red*.

Rysunek (Rys. 9) poniżej przedstawia rezultat implementacji powyższej modyfikacji.



Rys. 9 Rezultat modyfikacji kodu

Źródło: opracowanie własne

Jedną z bibliotek ***CSS*** jest ***Bootstrap***. Dzięki niemu strona jest responsywna. Biblioteka ***Bootstrap*** została opisana w rozdziale 3.

Czym więc tak właściwie jest ***ASP .NET***? Jest to zbiór technologii, które oparte są na frameworkach firmy ***Microsoft***. Na podstawie ***ASP .NET*** można budować aplikacje internetowe, pracujące na serwerach, komunikujące się z użytkownikiem za pomocą wybranej przeglądarki. Technologia ***ASP .NET*** została szczegółowo opisania w rozdziale 3.

* 1. **Rest API**

***REST API*** (ang. ***Re***presentational ***S***tate ***T***ransfer) jest to styl architektury oprogramowania, który opiera się o zbiór reguł. Opisują one jak definiowane są zasoby oraz jaki jest do nich dostęp.

***API*** (ang. ***A***pplication ***P***rogramming ***I***nterface) jest to zbiór reguł, który definiuje komunikacje między programami komputerowymi.

Zadaniem ***REST API*** jest odseparowanie interfejsu użytkownika od operacji na serwerze. Klient za pomocą strony internetowej przekazuje polecenia do serwera, nie mając wpływu, co się tam dzieje. Serwer natomiast, przekazuje odpowiedź do klienta i nie ma wpływu na to, co klient wykonuje na stronie.

„***RESTful*** ***api*** oraz sposób jego implementacji zostało zdefiniowane przez ***Roy’a Fielding’a*** w jego rozprawie doktorskiej z 2000 roku. ***REST*** zostało zaprojektowane w taki sposób, aby w komunikacji pomiędzy urządzeniami korzystać z istniejących już protokołów.”[[1]](#footnote-1) ***REST*** jest bardzo elastycznym narzędziem, potrafi współpracować z wieloma typami danych, np.: ***XML, JSON***. ***Roy Fielding***, który opracował ***REST API***, zdefiniował 6 podstawowych elementów, którymi trzeba się kierować przy ocenie danego ***API***:

* architektura typu klient – serwer,
* buforowanie danych, tam gdzie jest to możliwe (najlepiej po stronie klienta),
* niezależność poszczególnych połączeń,
* oddzielenie implementacji po stronie klienta i serwera z użyciem jednolitego interfejsu,
* podział architektury ***REST*** na współdziałające ze sobą warstwy,
* pozwala na przesyłanie kolejnych fragmentów kodu za pomocą ***API***.
  1. **Entity Framework**

***Entity Framework*** jest narzędziem, które ułatwia programiście utworzenie bazy danych w środowisku ***.NET***. Generuje obiekty oraz encje zgodnie z tabelami bazy danych. ***Entity Framework*** jest wykorzystywany w wielu aplikacjach. Pozwala on na:

* wykonywanie podstawowych operacji ***CRUD*** (ang. ***C***reate, ***R***ead, ***U***pdate, ***D***elete),
* zarządzanie relacjami,
* tworzenie relacji dziedziczenia między encjami.

Aby korzystanie z ***Entity Framework*** było możliwe, trzeba go najpierw zainstalować. Zainstalowanie tego narzędzia jest możliwe dzięki konsoli ***NuGet Packages Manager***. Podczas tworzenia systemu Internetowego zbioru książek, ***Entity Framework*** zostało wykorzystane, aby zmapować utworzone klasy modelu bazy do tabel w bazie danych. Narzędzie to pozwoliło na utworzenie relacji pomiędzy tabelami.

Na poniższym rysunku (Rys. 10) znajduje się schemat pokazujący działanie ***Entity Framework***.



Rys. 10 Schemat działania Entity Framework

Źródło: <https://www.plukasiewicz.net/Artykuly/EntityFramework> [16.01.2020]

# **Wybrane technologie do tworzenia serwisów internetowych**

Niniejszy rozdział zawiera charakterystykę wybranych technologii do tworzenia serwisów internetowych. Rozdział został podzielony na:

* języki internetowe,
* środowiska programistyczne,
* systemy baz danych.
  1. **Języki programowania**

W niniejszym rozdziale zamieszczone zostały informacje o najpopularniejszych językach programowania.

* + 1. **C#**

***C#*** to obiektowy język programowania ściśle zintegrowany z platformą ***.NET***. ***C#*** uważany jest za język łatwy do nauki, dzięki wielu gotowym rozwiązaniom. Język ten posiada bardzo rozbudowaną bibliotekę ***BCL***. Pozwala ona na rozwijanie różnych aplikacji sieciowych, okienkowych, konsolowych, czy też bazodanowych. Do niedawna ***C#*** był stosowany do pisania aplikacji na system ***Windows***, lecz po rozwinięciu framework’a ***.NET***, aplikacje pisane w tym języku są dostępna również na systemach takich jak ***Linux*** i ***Mac***. ***C#*** jest używany przez technologię ***ASP .NET*** do tworzenia aplikacji webowych. ***C#*** to wieloplatformowy język, którego można wykorzystać do napisania niemalże każdej aplikacji, na dowolny system operacyjny. Język ***C#*** jest wykorzystywany również do tworzenia aplikacji mobilnych za pomocą narzędzia ***Xamarin***. W języku ***C#*** każda zmienna musi mieć zadeklarowany typ przechowywanych danych. W przeciwnym wypadku, program wychwyci i zwróci błąd. Dzięki temu program jest jednoznaczny oraz szybciej może wychwycić błąd.

* + 1. **Java**

„***Java*** to język programowania i platforma do tworzenia oprogramowania komputerowego wprowadzona przez firmę ***Sun Microsystems*** po raz pierwszy w 1995 r. Istnieje wiele aplikacji i serwisów internetowych, które nie będą poprawnie działać, jeśli w systemie użytkownika nie będzie zainstalowane oprogramowanie ***Java***, a ich liczba wciąż rośnie.”[[2]](#footnote-2)Język ten bez żadnych przeszkód można wykorzystywać do pisania aplikacji na różne systemy operacyjne. Składnia Javy jest bardzo podobna do języka ***C***, ***C++***. ***Java*** ma bardzo szerokie zastosowanie, ze względu na wydajność oraz łatwą eliminację błędów podczas tworzenia aplikacji. Język ***Java*** jest bardzo mocno ukierunkowany na programowanie obiektowe, wyjątkiem są typy proste, takie jak ***int***, ***float***. Wszystkie obiekty są instancjami klasy ***Object***, dzięki czemu, wszystkie obiekty maja wspólny podzbiór podstawowych funkcji. Kod, który powstaje jest niezależny od procesora i systemu operacyjnego, wykonywany jest przez maszynę wirtualna ***Javy***. Maszyna wirtualna jest tłumaczem między kodem uniwersalnym, a kodem dostosowanym do konkretnego procesora i systemu operacyjnego. Wadą kompilowania programu poprzez maszynę wirtualną jest wydłużony czas na ładowanie programu.

* + 1. **JavaScript – REACT oraz Angular**

***JavaScript*** jest językiem skryptowym, w którym funkcje są traktowane jak obiekty, które można przechowywać w zmiennych oraz przekazywać jak każdy inny obiekt. ***JavaScript*** jest używany nie tylko, jako język skryptowy dla stron internetowych. Używany jest przez wiele innych środowisk poza przeglądarką, np.: ***Adobe Acrobat***, ***Node.js***.

***JavaScript*** posiada bardzo popularną od niedawna bibliotekę – ***REACT***. Służy ona do budowania graficznych interfejsów aplikacji internetowych. ***REACT*** został stworzony przez programistę ***Facebooka***, ***Jordana Walke***. ***REACT*** wykorzystuje ***DOM*** (ang. ***D***ocument ***O***bject ***M***odel), czyli ***Obiektowy Model Dokumentu***. Polega to na tym, że ***REACT*** przechowuje ten model w pamięci, następnie po zmianie statusu, sprawdza różnice między wirtualnym a realnym modelem i wprowadza zmiany. ***REACT*** wykorzystuje także język ***JSX***. Jest to nakładka na ***JavaScript***, umożliwiająca wstawienie kodu ***HTML*** bezpośrednio w kodzie. ***REACT*** jest obecnie używany przez takie aplikacje internetowe jak ***Netflix***, ***PayPal***.

***Angular*** jest frameworkiem ***JavaScrpit*** utworzonym przez ***Google***. ***Angular*** zyskał popularność wśród projektantów front-endu. Dzięki dwukierunkowości tego frameworku, programiści mogą zmniejszyć ilość wprowadzanego kodu. ***Angular*** sam odnajduje zmiany w modelu i wprowadza odpowiednie zmiany w kodzie. Programiści mają możliwość wdrożenia wzorca ***MVC*** (ang. ***M***odel – ***V***iew - ***C***ontroller).

* + 1. **PHP**

***PHP*** jest zazwyczaj stosowany do tworzenia skryptów na serwerach ***WWW***. Dzięki modułowej budowie, używany jest również do tworzenia graficznych stron internetowych oraz aplikacji webowych. ***PHP*** to język wieloplatformowy. Daje możliwość współpracowania z różnymi rodzajami źródeł danych, takimi jak, ***DBMS*** (System Zarządzania Bazami Danych), dokumenty ***XML***, pliki tekstowe lub nawet serwisy ***WWW***. Obsługuje wiele popularnych baz danych: ***MySQL***, ***MS SQL***, ***PostgreSQL***, ***Oracle***. Skrypty pisane w języku ***PHP*** są umieszczone w pliku tekstowym, ale pozwala również na wykonanie skryptów w linii poleceń, podobnie jak ***Perl*** czy ***Python***. Funkcjonalność ***PHP*** zawarta jest w czterech zbiorach modułów:

* moduły jądra,
* moduły oficjalne,
* repozytorium PECL,
* repozytorium PEAR.

***PHP*** jest wykorzystywany w większości obecnie istniejących serwisach internetowych. Należą do nich m.in.: fora dyskusyjne, systemy zarządzania treścią, klient bazy danych, czy nawet aplikacje pocztowe.

* 1. **Środowiska programistyczne**

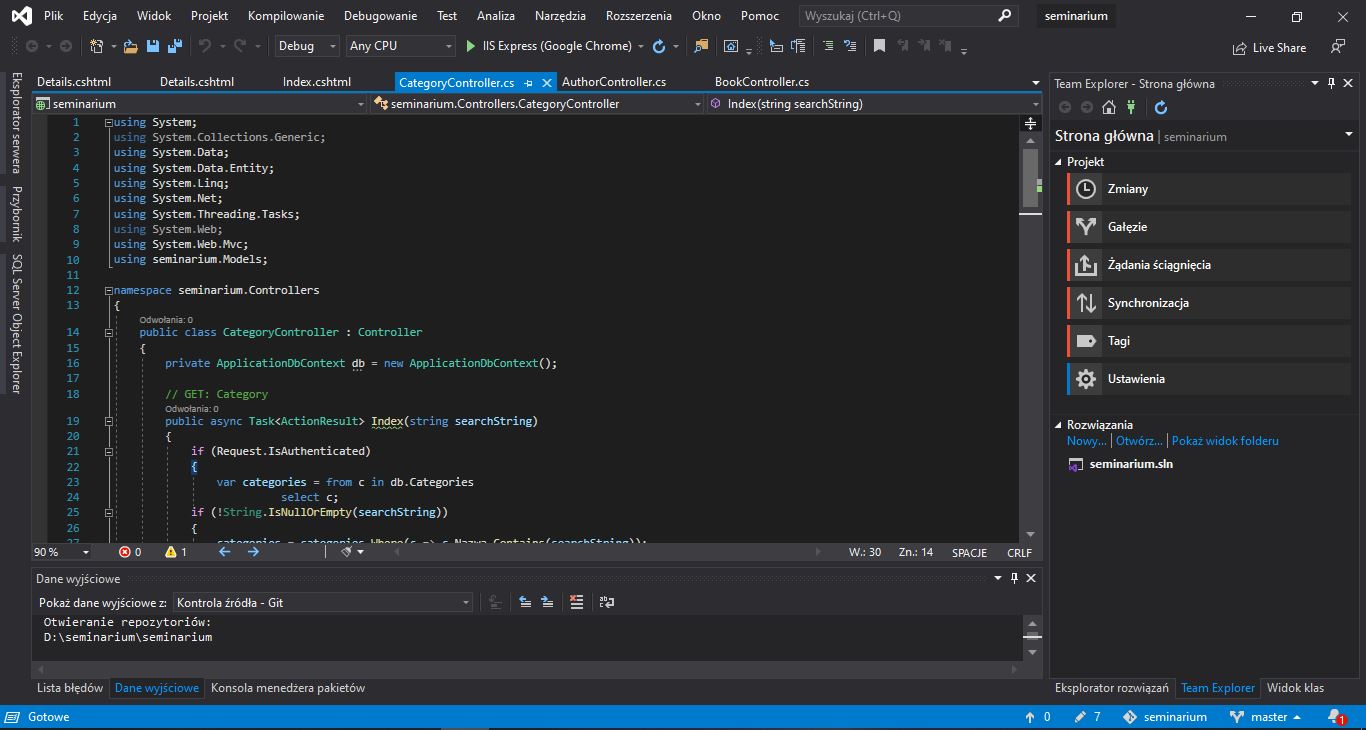
W niniejszym rozdziale przedstawione zostaną najpopularniejsze zintegrowane środowiska programistyczne, ***IDE*** (ang. ***I***ntegrated ***D***evelopment ***E***nvironment).

* + 1. **Visual Studio**

***Visual Studio 2019*** jest zintegrowanym środowiskiem, służącym do programowania, które zostało stworzone przez firmę ***Microsoft***. ***Visual Studio*** używa się do tworzenia aplikacji typu: ***Windows*** ***Forms***, ***WPF***, ***Web Applications***, ***Web Sites***. Aplikacje mogą być tworzone na różne platformy.

***Visual Studio*** zawiera edytor kodu oraz mechanizmy refaktoryzacji kodu. Debugger, który jest zintegrowany ze środowiskiem, działa na poziomie kodu źródłowego jak i maszyny. Narzędzia, jakie programista może znaleźć w ***Visual Studio*** to m.in.: narzędzie do tworzenia aplikacji ***WPF***, ***Windows Forms*** oraz ***Web***, narzędzie do tworzenia baz danych oraz klas. Poprzez pobranie odpowiednich dodatków programista może zwiększyć funkcjonalność oprogramowania.

Poniższy rysunek (Rys. 11) przedstawia wygląd opisanego środowiska programistycznego.



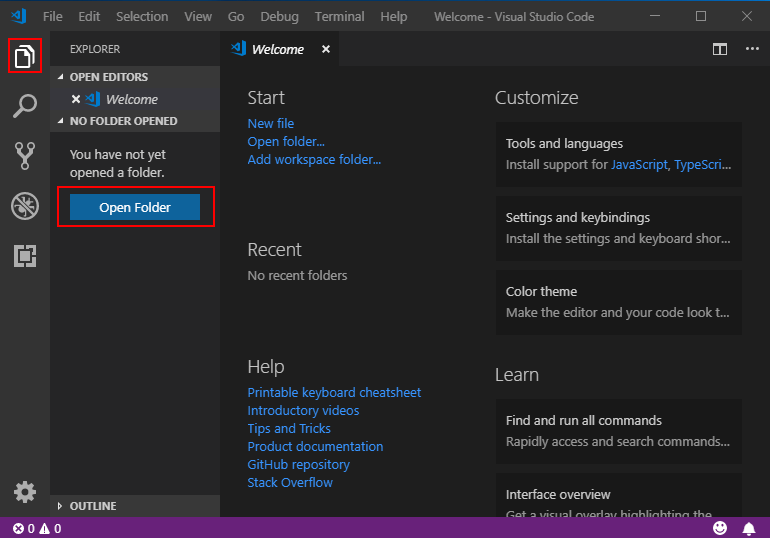
Rys. 11 Środowisko programistyczne Visual Studio 2019

Źródło: opracowanie własne

* + 1. **Visual Studio Code**

***Visual Studio Code*** jest to darmowy, desktopowy edytor kodów źródłowych. Pozwala on na kolorowanie składni wielu języków programowania. Umożliwia on również zarządzaniem wersjami kodu źródłowego za pomocą systemu kontroli wersji ***Git***.

Na Rys. 12 przedstawiony został wygląd środowiska ***Visual Studio Code***.



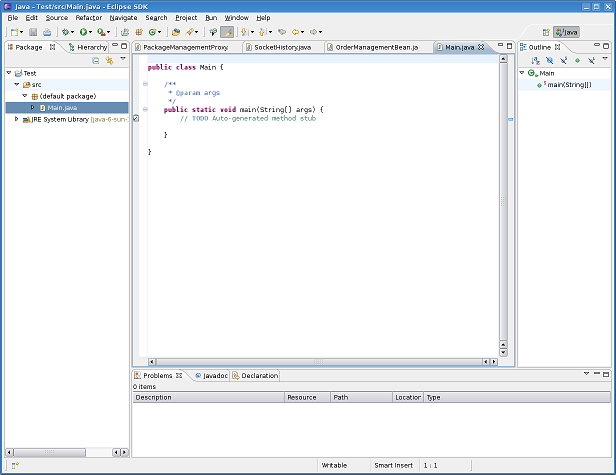
Rys. 12 Środowisko programistyczne Visual Studio Code

Źródło: <https://docs.microsoft.com/pl-pl/dotnet/core/tutorials/with-visual-studio-code>[17.02.2020]

* + 1. **Eclipse**

***Eclipse*** jest to środowisko programistyczne, umożliwiające pisanie aplikacji w języku ***Java***. Oprogramowanie jest udostępniane jako otwarte. ***Eclipse*** współpracuje z najpopularniejszymi systemami operacyjnymi.

Poniższy rysunek (Rys. 13) przedstawia środowisko ***Eclipse***.



Rys. 13 Środowisko programistyczne Eclipse

Źródło: <https://javastart.pl/baza-wiedzy/wprowadzenie/eclipse-ide-wstep-i-instalacja>[17.02.2020]

* 1. **Systemy baz danych**

W niniejszym rozdziale opisane zostaną najpopularniejsze narzędzia bazodanowe, używane do tworzenia aplikacji internetowych i nie tylko.

* + 1. **MS SQL Server**

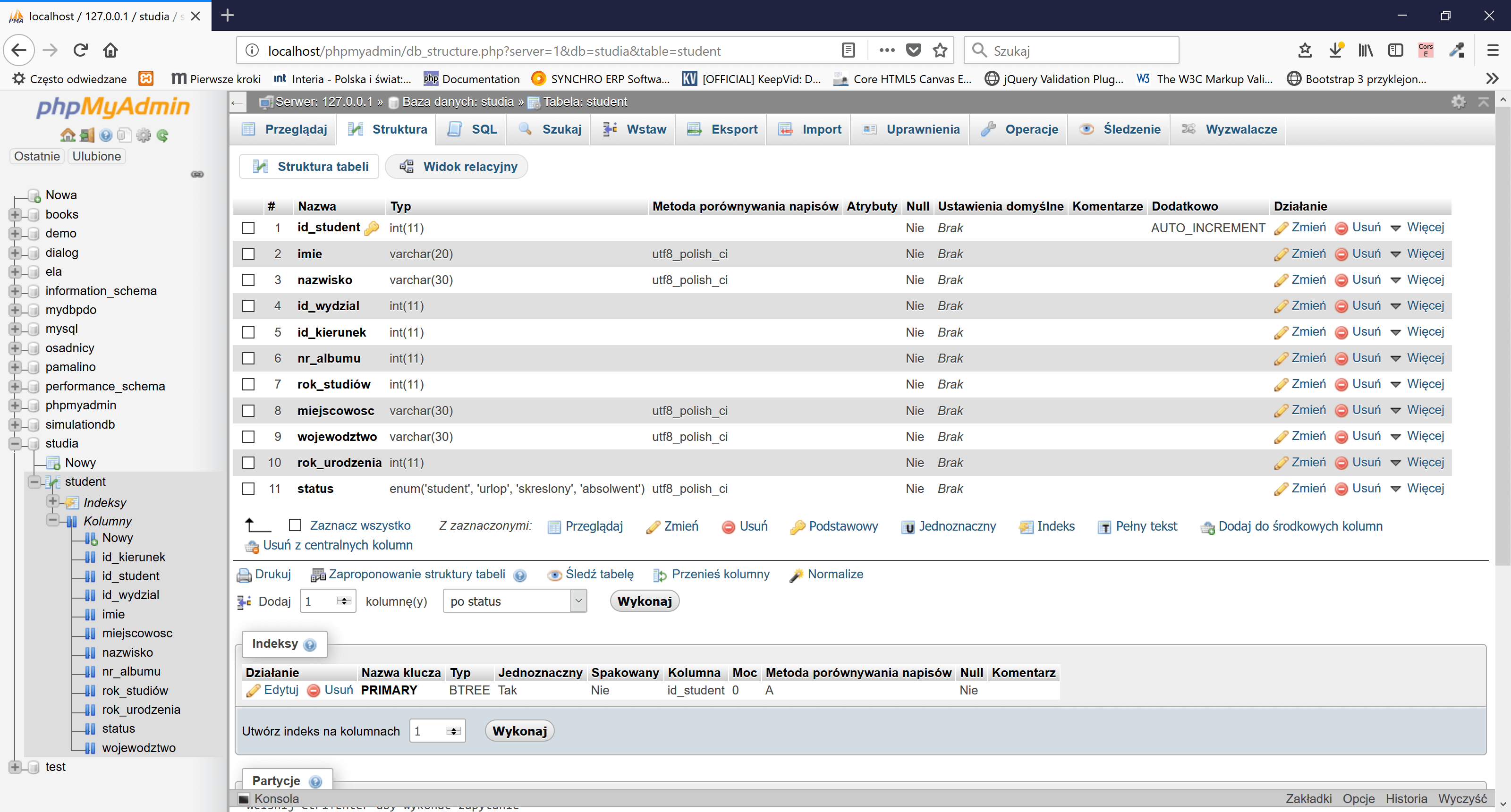
***MS SQL Server*** jest to ***S***ystem ***Z***arządzania ***B***azą ***D***anych (***SZBD***). Platforma ***SQL Server*** opiera się na usłudze serwera. W trakcie instalacji środowiska jest możliwość wybrania tych komponentów, które będą potrzebne do realizacji zadań. Najważniejszym komponentem ***MS SQL Server*** jest silnik bazy danych. Odpowiada on za przetwarzanie zapytań, zarządza składowaniem oraz ochroną danych. Środowisko posiada wiele innych komponentów ułatwiających pracę, są nimi m.in.:

* usługi analityczne,
* usługi raportujące,
* usługi integracji danych.

Podczas instalacji, ***MS SQL Server*** daje możliwość wyboru dodatkowych funkcjonalności oraz narzędzi:

* ***SQL Server Management Studio,***
* ***SQL Server Configuration Manager,***
* ***SQL Server Profiler,***
* ***SQLCMD***.
  + 1. **MySQL**

***MySQL*** to open source’owy system zarządzania bazą danych. Jego zastosowanie można znaleźć podczas tworzenia aplikacji w języku ***PHP. MySQL*** jest szybkim systemem oraz bardzo wydajnym. Środowisko to jest dostępne na wielu systemach operacyjnych, takich jak: ***Linux***, ***Windows***, ***OpenSolaris***, ***Symbian***. ***MySQL*** oferuje również ochronę danych poprzez autentykację użytkowników, wsparcie protokołów ***SSH, SSL***, enkrypcja i deskrypcja danych, rozdzielenie użytkowników, aby mieli możliwość zarządzania tylko tymi danymi, do których mają dostęp. Poniższy rysunek (Rys. 14) przedstawia przykładową tabelę w MySQL.



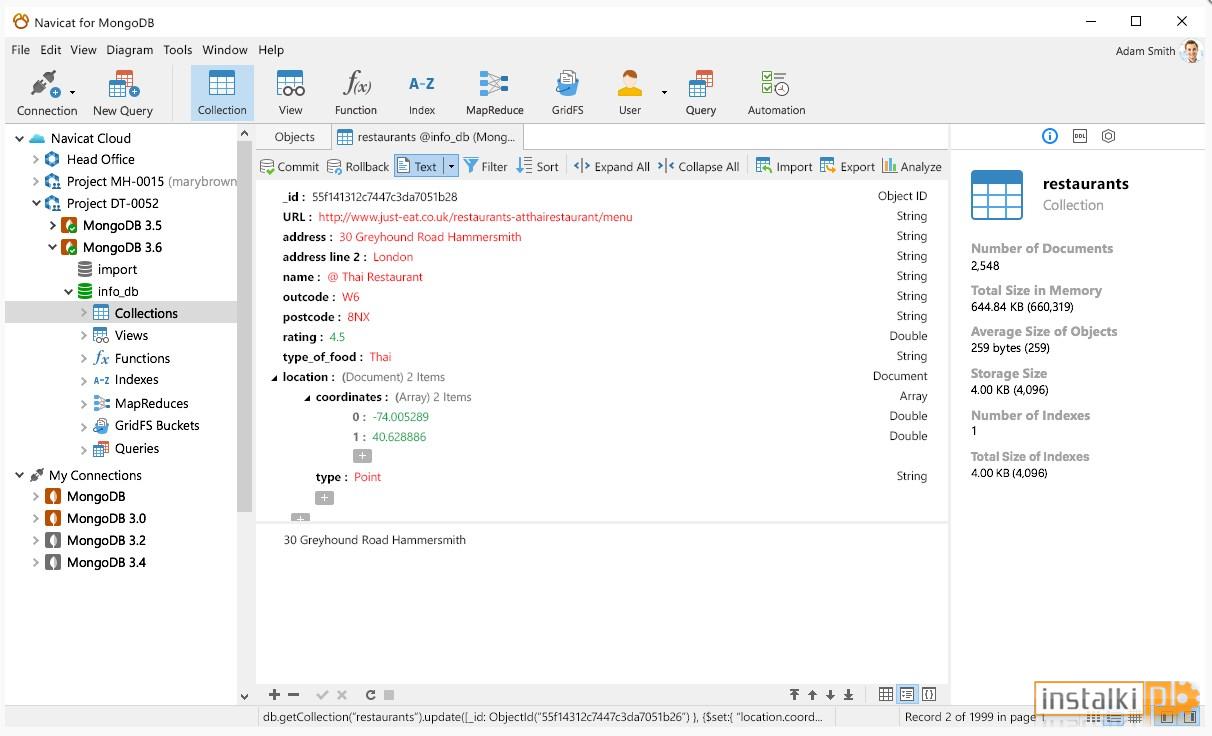
Rys. 14 Przykładowa tabela w bazie MySQL

Źródło: <http://www.galaxy.agh.edu.pl/~pamalino/programowanie/mysql/img/phpMyAdmin3.png>(28.02.2020)

* + 1. **Bazy nierelacyjne**

Nierelacyjne bazy danych zapewniają możliwość przechowywania, organizowania oraz wyszukiwania danych w inny sposób niż odbywa się to w relacyjnych bazach. Bazy nierelacyjne przydają się jeśli mamy do czynienia z bazą o dużym wolumenie, np.: ***Big Data***. W bazach nierelacyjnych rezygnuje się ze spójności na rzecz wydajności i tolerancji na partycje. Dane w bazach ***NoSQL*** przechowywane są jako kolekcje, w jednej kolekcji mogą być dane jednego typu. Przykładem takiej nierelacyjnej bazy jest np.: ***MongoDB***, ***Apache Cassandra***.

Bazy nierelacyjne nie mają predefiniowanych schematów tabel czy typów danych, co powoduje, że są bardziej elastyczne. Zaletą baz ***NoSQL*** jest ich niska cena oraz prostota w utrzymaniu. Minusem baz ***NoSQL*** jest to, że nie są tak znane jak bazy ***SQL***. Bazy nierelacyjne mogą być wykorzystane w celu podtrzymania pamięci podręcznej stron, czy aplikacji. Na rysunku (Rys. 15) poniżej przedstawiona została przykładowa tabela bazy ***MongoDB***.



Rys. 15 Widok kolekcji w bazie MongoDB

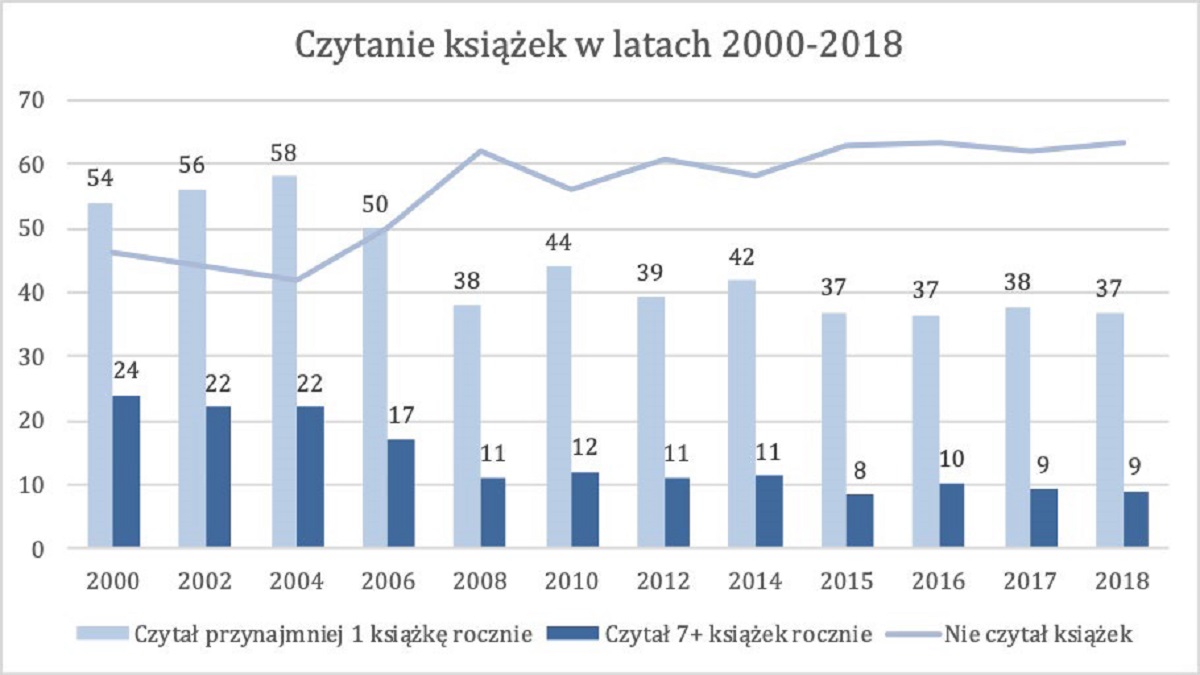
Źródło: <https://www.instalki.pl/programy/download/Windows/bazy_danych/Navicat_for_MongoDB.html>

# **Aktualny stan wiedzy**

Niniejszy rozdział opisuje charakterystykę problemu oraz cel wykonania pracy. Przedstawione zostaną również technologie jakie zostały wykorzystane do stworzenia aplikacji. Zostaną również skrótowo opisane aplikacje o tematyce zbliżonej do tematu pracy.

* 1. **Wprowadzenie**

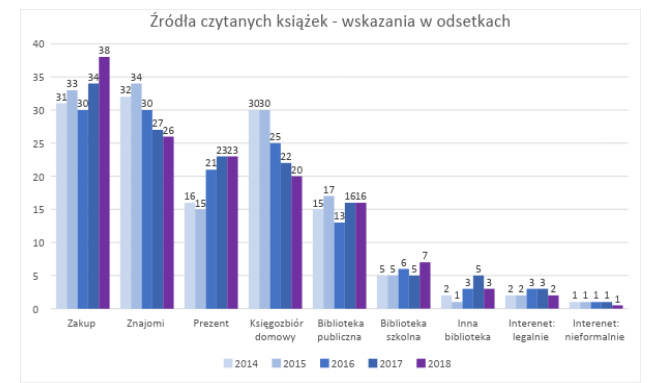
Od bardzo wielu lat można zauważyć, że ilość osób czytających regularnie książki maleje. Młodzież woli spędzić czas grając w gry komputerowe, które pochłaniają bardzo dużo czasu. Na dalszy plan schodzą nawet obowiązkowe lektury szkolne. Osoby pracujące zazwyczaj są pochłonięte życiem codziennym, w którym często brakuje czasu na przeczytanie czegoś dłuższego niż artykuł w gazecie, czy na popularnych stronach internetowych. Problemem jest nie tylko brak czasu na czytanie książek, lecz również postęp technologiczny. Sprawił on, że człowiek ma coraz więcej sposobów na spędzenie wolnego czasu. Można tutaj wymienić gry single player, multi player, wybijające się na rynku gry ***Virtual Reality***, serwisy z filmami i serialami, np.: ***Netflix***, ***HBO***. Dużo wolnego czasu poświęcane jest również na przeglądanie portali społecznościowych. Na poniższym rysunku (Rys. 16) został pokazany wykres, na którym należy zauważyć spadek ilości czytanych książek na przestrzeni lat 2000 do 2018.



Rys. 16 Wykres ilości czytanych książek w latach 2000 do 2018

Źródło: https://www.spidersweb.pl/rozrywka/2019/03/25/czytelnictwo-w-polsce-2018-raport-opinia/[13.01.2020]

Aby nadążyć za postępem technologicznym potencjalny czytelnik, mimo elektronicznych czytników książek, woli ekranizowaną wersję książki, która jest bardzo często okrojona ze szczegółów. Zaletą czytania książek jest wzbogacanie języka oraz wczuwanie się w atmosferę, co pobudza wyobraźnię czytelnika. Wiele ludzi odchodzi jednak od czytania ze względu na możliwość pozyskania wiedzy w innych źródłach. Zazwyczaj w statystykach biorą udział głównie dzieci i młodzież, które nie zawsze są zachęcane do czytania. Z tego względu obwinianie rozwoju technologii czy Internetu mija się z celem. Od kilku lat organizowane są akcje czytania książek dla najmłodszych, dzięki czemu czytelnictwo wśród tej grupy podniosło swój poziom. Popularność Internetu wzrosła na tyle, że czytelnik często przeszuka wszystkie dostępne zasoby aby znaleźć książkę, nie zawsze w legalnych źródłach. Na poniższym rysunku (Rys. 17) został przedstawiony wykres, z jakich źródeł korzystają czytelnicy, aby pozyskać książkę.



Rys. 17 Źródła pozyskiwania książek w latach 2014 do 2018

Źródło: <https://www.spidersweb.pl/rozrywka/2019/03/25/czytelnictwo-w-polsce-2018-raport-opinia/>[13.01.2020]

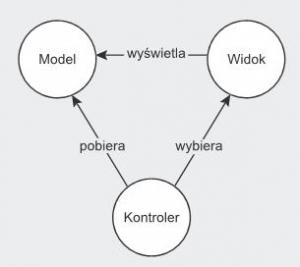
Celem utworzonej aplikacji jest zbliżenie potencjalnego czytelnika, aby skłonił się do przeczytania każdego rodzaju książki. Aplikacja została utworzona z myślą o postępie technologicznym, aby odnalezienie szukanej książki stało się łatwiejsze oraz przyjemniejsze. Istnieją strony z tego rodzaju materiałami, lecz im więcej takich źródeł, tym większa szansa, że czytelnictwo ponownie wzbije się w górę.

* 1. **Wykorzystane technologie**

Model tworzenia aplikacji ***ASP .NET MVC 5*** jest najpopularniejszym modelem, który wykorzystują programiści. Zapewnia on przejrzystą strukturę, intuicyjność zarządzania oraz odizolowaną logikę od interfejsu użytkownika. Skrót ***MVC*** oznacza ***M***odel-***V***iew-***C***ontroller, poniżej znajduje się opis składowych wzorca ***MVC***:

* ***M***odel – w tworzonym projekcie, w katalogu Model znajduje się model bazy danych. Pliki, które się znajdują w tym katalogu są klasami. Podczas tworzenia aplikacji z podejściem code first, pliki z modelem bazy będą tworzone przez programistę. Natomiast przy wyborze podejścia database first, pliki z modelem bazy zostaną automatycznie dodane do projektu.
* ***V***iew – pliki do projektu, umieszczone w katalogu Views, odpowiadają za prezentację danych, które poda kontroler. Ta część wzorca używa takich technologii jak, ***HTML***, ***CSS*** oraz innych języków, potrzebnych do zarządzania widokiem strony.
* ***C***ontroller – utworzone pliki w katalogu Controllers pozwalają na odbiór i przetwarzanie danych wprowadzonych przez użytkownika z poziomu strony, czy też tych pobranych z serwera bazodanowego. Kontroler może wywołać żądanie wprowadzenia danych, później wysłać je do modelu, z modelu do widoku, który wyświetli te dane użytkownikowi. Może zostać wywołany również inny kontroler.

Na poniższym rysunku (Rys. 18) znajduje się schemat, przedstawiający zasadę działania wzorca ***MVC***.



Rys. 18 Schemat wzorca MVC

Źródło: <http://www.php.pl/Wortal/Artykuly/Framework/Frameworki-dla-PHP-czyli-wydajne-tworzenie-aplikacji/MVC>[16.01.2020]

Podejścia, które można wyszczególnić do utworzenia aplikacji technologią ***ASP .NET*** to:

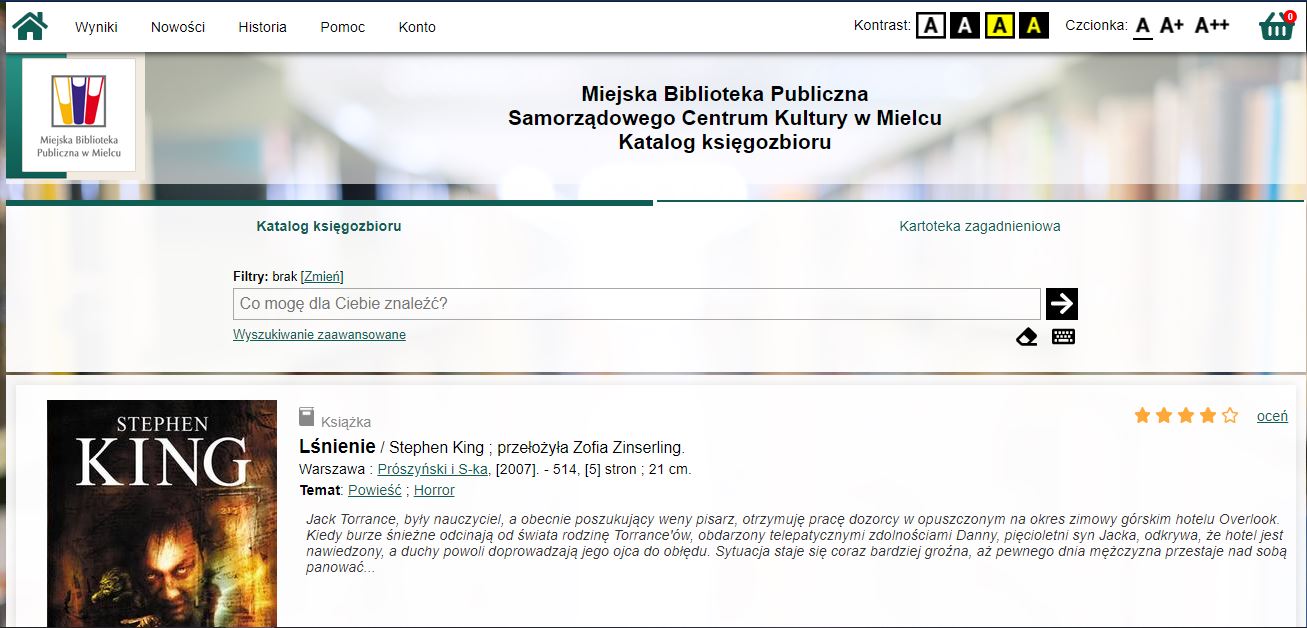
* code first – podejście to polega na tym, że nie jest tworzony schemat bazy wprost, lecz tworzony jest pewien szablon w języku ***C#***, który z pomocą ***Entity Framework*** generuje model bazy danych.
* database first – w tym podejściu trzeba najpierw utworzyć schemat bazy danych, następnie zimportować bazę do modelu, a ***Entity Framework*** wygeneruje odpowiednie klasy oraz je zmapuje.
* model first – to podejście pozwala na utworzenie modelu bazy w wizualnym designerze. Następnie tworzona jest na tej podstawie baza danych.

Aktualnie tworząc strony internetowe, czy aplikacje webowe, wykorzystuje się bibliotekę ***CSS***, która powoduje, że widok strony dopasowuje się do rozdzielczości ekranu, na którym jest wyświetlany, czyli jest responsywny. Ta biblioteka ***CSS*** to ***Bootstrap***. Działa on na zasadzie siatki, która dopasowuje się do wielkości ekranu i ustawia elementy na stronie według określonej szerokości oraz kolejności.

Baza danych, która została utworzona w ramach projektu znajduje się na zdalnym serwerze ***Microsoft*** ***SQL Server***. Baza jest zarządzana przez narzędzie ***SQL Server Management Studio 2017***.Aplikacja ta pozwala na zarządzanie wszystkimi komponentami, takimi jak, baza danych, usługi raportowe, analityczne, itp. Dzięki temu narzędziu można budować zapytania oraz skrypty. Aplikacja zawiera ***Object Explorer***, za pomocą którego możemy przeglądać, wybierać oraz wykonywać różne działania na obiektach serwera. Do utworzonej aplikacji wykorzystana została bezpłatna wersja Express, która jest ograniczona w porównaniu do wersji pełnej. Brak w tej wersji usług analitycznych, raportowych oraz integracji.

* 1. **Wybrane aplikacje internetowe związane z tematyką pracy**

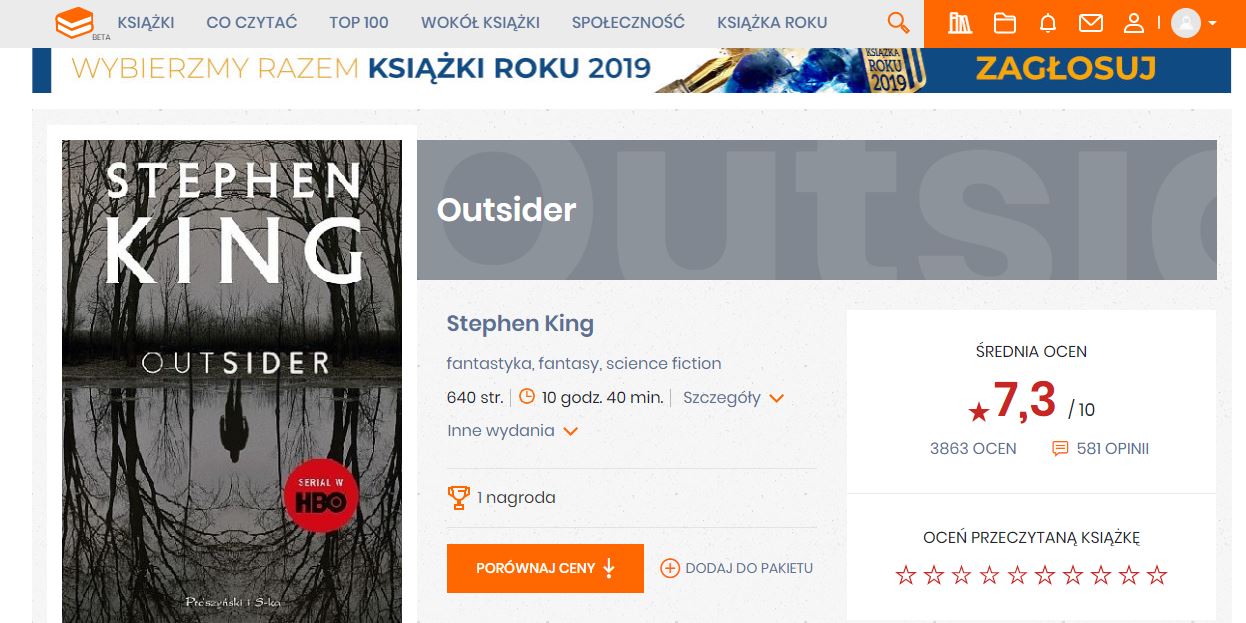
Utworzona aplikacja nie jest jedyną taką aplikacją. O tej tematyce istnieją już aplikacje, które są odwiedzane przez użytkowników oraz spełniają swoją rolę. Przykładem aplikacji może być chociażby, katalog online z książkami ***Biblioteki Miejskiej w Mielcu***. Przedstawiony na Rys. 19 katalog pozwala na wyszukanie interesującej czytelnika książki, przeglądnięcie informacji na jej temat oraz dodanie książki do koszyka. W koszyku natomiast możemy zdecydować czy książkę chcemy zarezerwować, aby ją wypożyczyć czy też nie. Niestety w katalogu nie ma możliwości pobrania czy też zakupienia książki.



Rys. 19 Przykład wyszukanej książki w katalogu Biblioteki Miejskiej w Mielcu

Źródło: opracowanie własne

Następnym serwisem, który można nazwać zbiorem książek jest strona ***lubimyczytać.pl*** (Rys. 20). Zalogowany użytkownik może dodawać do swojej wirtualnej biblioteczki książki wyszukane w bazie serwisu. Dodane książki można dzielić na różne półki, np.: do przeczytania, teraz czytane, przeczytane. Jeśli czytelnik zechce kupić wybraną książkę, wystarczy że wybierze opcje porównania cen. Serwis automatycznie porówna kilka sklepów i posortuje czytelnikowi ceny, od najniższej do najwyższej.



Rys. 20 Wyszukana książka w serwisie lubimyczytac.pl

Źródło: opracowanie własne

# **Specyfikacja systemu**

W tym rozdziale zostaną przedstawione wymagania funkcjonalne i niefunkcjonalne utworzonego systemu. Następnym podrozdziałem jest podział systemu na część administracyjną oraz na część użytkownika.

* 1. **Wymagania funkcjonalne i niefunkcjonalne**

Niniejszy rozdział przedstawia wymagania funkcjonalne, jakie ma zawierać system, aby spełnić podstawową funkcjonalność utworzonej aplikacji. W poniższej tabeli (Tabela 1) opisuje wspomniane wymagania.

Tabela 1 Wymagania funkcjonale

|  |  |
| --- | --- |
| **Wymaganie** | **Opis** |
| Logowanie i rejestracja | Użytkownik rejestruje się do systemu, podając odpowiednie dane (adres e-mail, hasło). Aby zalogować się do systemu trzeba podać adres e-mail oraz hasło |
| Wyszukiwanie wybranego autora | Wyszukiwanie odbywa się po zalogowaniu do systemu oraz przejściu do widoku listy autorów, następnie użytkownik może sortować listę bądź filtrować ją wpisując odpowiedni ciąg znaków. |
| Wyszukiwanie wybranej książki | Aby wyszukać dana książkę, użytkownik musi być zalogowany. Wyszukiwanie książki odbywa się za pomocą sortowania lub filtrowania. |
| Pobieranie książki | Zalogowany użytkownik po wybraniu książki i przejściu do szczegółów, może skopiować zamieszczony link i pobrać wybraną książkę. |
| Dodawanie danych o książkach i autorach | Osoba mająca dostęp do bazy danych może dodawać rekordy |
| Modyfikacja danych | Osoba z odpowiednimi uprawnieniami do bazy danych może edytować wprowadzone dane. |

Źródło: opracowanie własne

Wymagania niefunkcjonalne określają jakość utworzonego systemu. Nie musi on spełniać tych wymagań, lecz muszą się one znajdować, aby spełnić wymagania interesariuszy. Poniższa tabela przedstawia wymaganie niefunkcjonalne, które zostały spełnione w aplikacji.

Tabela 2 Wymagania niefunkcjonalne

|  |  |
| --- | --- |
| **Wymaganie** | **Opis** |
| Responsywność aplikacji | Oznacza, że wyświetlana strona, za pomocą siatki, dopasowuje się do rozdzielczości ekranu. |
| Intuicyjny interfejs | Aplikacja powinna być prosta w obsłudze, a wyświetlane informacje czytelne oraz zrozumiałe dla użytkownika. |
| Bezpieczeństwo | Wprowadzane dane do systemu powinny być możliwe tylko dla uprawnionych osób. Dane wyświetlane powinny być dostępne tylko po zalogowaniu użytkownika. |

Źródło: opracowanie własne

* 1. **Opis poszczególnych paneli systemu**

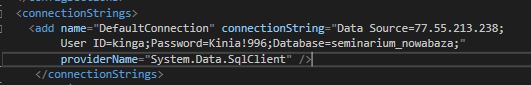
System został podzielony na część, którą zarządza administrator oraz na część, którą obsługuje użytkownik. W części administracyjnej zarządzanie odbywa się w ***SQL Server Management Studio***, na serwerze zdalnym. W dalszym planie rozwoju systemu przewidziane jest utworzenie panelu administracyjnego bezpośrednio w aplikacji Internetowego zbioru książek. Po zalogowaniu się użytkownik ma możliwość przeglądania danych o autorach, książkach oraz pobierać wybrane książki. Aby wyszukać odpowiednie dane, użytkownik ma możliwość filtrowania oraz sortowania danych. Opisane panele zostały szczegółowo przedstawione w rozdziale 6.

# **Wybrane funkcjonalności Internetowego zbioru książek**

W niniejszym rozdziale zostały opisane poszczególne funkcjonalności systemu z podziałem na wybrane w kontrolerze, widoku i modelu.

* 1. **Model**

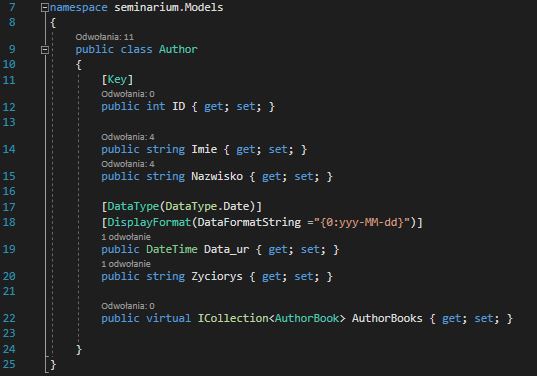
Na poniższym zdjęciu (Rys. 21) przedstawione zostało połączenie do bazy danych, która jest na zdalnym serwerze. Parametrami, które trzeba uzupełnić, są: nazwa połączenia, adres serwera bazy danych, nazwa użytkownika, hasło oraz nazwa bazy, w której dane mają się znajdować.



Rys. 21 Fragment kodu odpowiadający za połączenie z bazą danych

Źródło: opracowanie własne

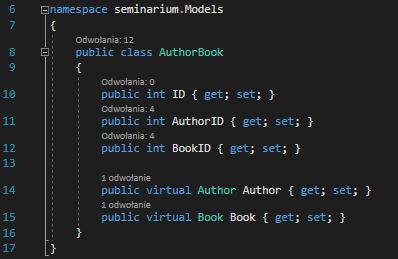
Fragment kodu na rysunku (Rys. 22) poniżej przedstawia model bazy tabeli Author. W tabeli znajdują się takie pola jak ID, czyli klucz główny, Imię, Nazwisko, Data urodzenia oraz Życiorys. Na końcu modelu ustawiona została zależność do tabeli AuthorBook. Parametr public oznacza, że dane są widoczne publicznie dla wszystkich klas. Parametry takie jak: int, string, DateTime oznaczają typy zmiennych. Zmienna ID przyjmuje typ int, czyli jest zmienna liczbową. Zmienne Imię, Nazwisko, Życiorys przyjmują typ string, czyli są zmiennymi tekstowymi. Typ zmiennej DateTime, przypisany do Daty\_urodzenia oznacza datę oraz czas. Na potrzeby aplikacji, zmienna została zmodyfikowana do typu DateTime.Date. Do zmiennej ID został przypisany parametr [Key], oznacza on że zmienna jest kluczem głównym tabeli. Zmienna DateTime została oznaczona parametrem [DisplayFormat], w nawiasach okrągłych jest wskazane jaki format ma przyjmować domyślny format daty.



Rys. 22 Model bazy tabeli Author

Źródło: opracowanie własne

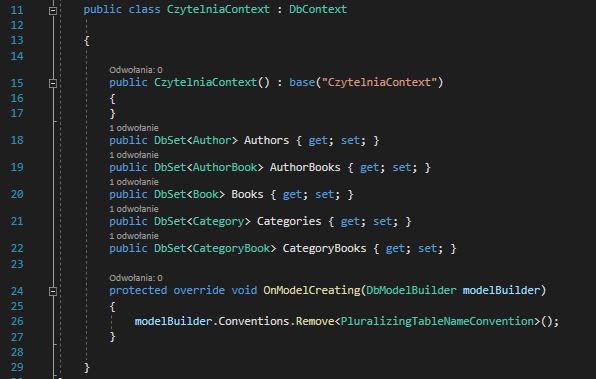
Rys. 23 przedstawia model tabeli AuthorBook. Ta tabela odpowiada za połączenie dwóch tabel Author oraz Book. Zawiera ona tylko pola ID, AuthorID, BookID oraz powiązania do odpowiednich tabel. Linijki kodu 14 oraz 15 tworzą powiązanie z tabelami Author oraz Book. Utworzone zmienne przyjmują typ int, czyli są zmiennymi numerowanymi. Zmienna ID jest numerem unikalnym każdego rekordu w tej tabeli. Zmienna AuthorID zawiera numer ID autora, do którego ma zostać przypisana książka. Zmienna BookID zawiera numer ID książki, która ma być przypisana do autora.



Rys. 23 Model tabeli AuthorBook

Źródło: opracowanie własne

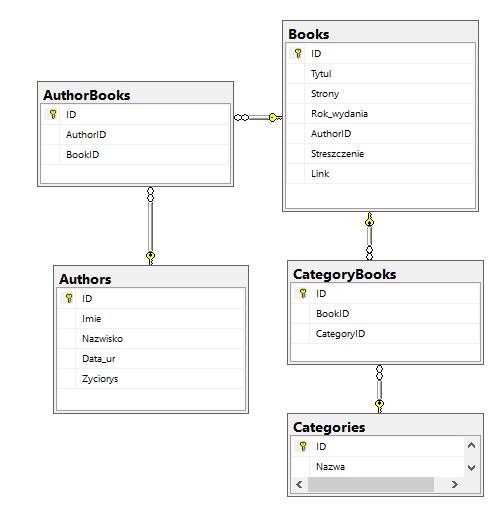
Po utworzeniu wszystkich modeli utworzono kontekst bazy. Fragment kodu został pokazany na Rys. 24. Kontekst zawiera odwołanie do wszystkich tabel utworzonych w modelu bazy.



Rys. 24 Fragment kodu CzytelniaContext

Źródło: opracowanie własne

Na Rys. 25 przedstawiony został diagram zależności między tabelami. Aby utworzyć zależność wiele do wielu, potrzebna jest tabela pośrednia. W przypadku poniższego modelu bazy, tabelami pośrednimi są AuthorBook oraz CategoryBook.

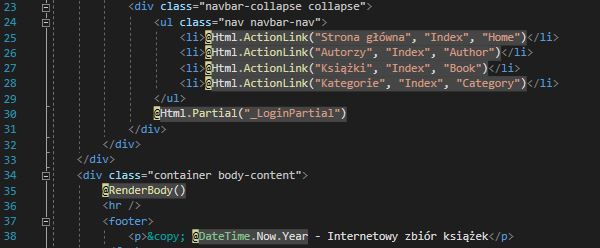


Rys. 25 Diagram zależności między tabelami

Źródło: opracowanie własne

* 1. **Widok**

Na poniższym rysunku (Rys. 26) znajduje się fragment kodu, który został utworzony domyślnie przez szablon tworzonej aplikacji ***ASP .NET***. Na potrzeby tworzonej aplikacji został on zmodyfikowany. Do automatycznie utworzonego panelu menu zostały dodane odwołania do widoków listy autorów, książek oraz kategorii. W nagłówku oraz stopce strony została dodana nazwa aplikacji. Funkcja ActionLink, działa jako odnośnik do wskazanej w parametrach funkcji strony. Na ekranie użytkownik zobaczy tylko tekst, wskazany w funkcji jako nazwa akcji. Funkcja Action Link zawiera takie parametry jak: nazwa akcji, strona która ma zostać otworzona, katalog widoku danej strony.



Rys. 26 Fragment kodu odpowiedzialnego za pasek menu

Źródło: opracowanie własne

Wraz z utworzeniem kontrolerów, utworzone zostały widoki dla każdego z nich. Utworzone katalogi zawierają standardowo pliki:

* Index.cshtml, który zawiera listę rekordów wprowadzonych do bazy,
* Create.cshtml, plik ten umożliwia utworzenie nowego rekordu,
* Details.cshtml, zawiera on szczegóły wybranego rekordu,
* Edit.cshtml, ten widok umożliwia edytowanie wybranego elementu,
* Delete.cshtml, ten plik pozwoli na usunięcie rekordu.

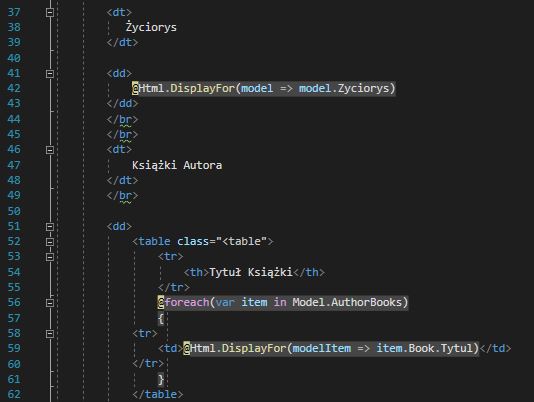
Na potrzeby utworzonej aplikacji, użytkownik widzi tylko dwa z tych widoków, Index oraz Details. Fragmenty kodu pliku Index.cshtml zostały przedstawione na Rys. 27. W przedstawionym fragmencie kodu znajdują się również pola umożliwiające filtrowanie danych (linijki od 10 do 13) oraz sortowanie danych (linijki 18 i 21). Parametr TextBox oznacza wprowadzenie do widoku pola tekstowego. Wewnątrz tego parametru znajduje się odwołanie do funkcji searchString, która jest zdefiniowana w kontrolerze. Znacznik input z parametrem type=”submit”, oznacza możliwość wysłania wprowadzonych danych do funkcji searchString. Funkcja sortOrder określona w linijkach 18 i 21 odwołuje się do definicji funkcji w kontrolerze. Funkcja ta pozwala na sortowanie rekordów w tabeli po kolumnach Imię i Nazwisko.



Rys. 27 Fragment kodu pliku Index.cshtml

Źródło: opracowanie własne

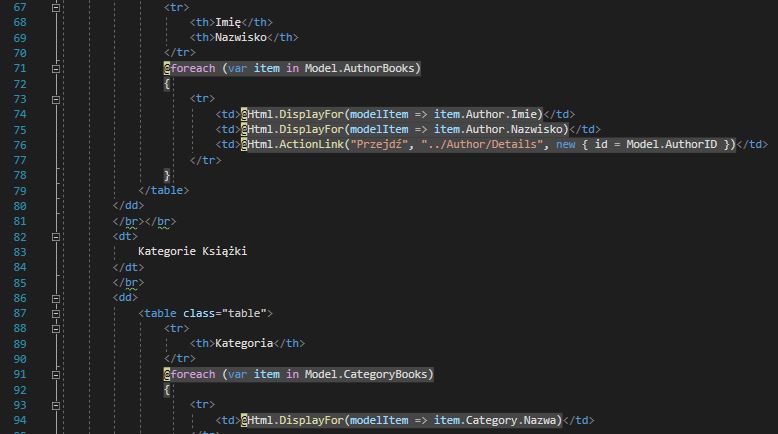
Na Rys. 28 znajduje się fragment kodu pliku Details.cshtml, przedstawia on odwołanie do tabeli AuthorBook. Za pomocą tego odwołania, widok Details tabeli Author, ma możliwość pokazać Tytuły książek, które są powiązane z danym autorem.



Rys. 28 Fragment kodu pliku Details.cshtml

Źródło: opracowanie własne

Poniższy rysunek (Rys. 29) przedstawia fragment kodu odpowiedzialny za wyświetlanie szczegółów wybranej książki. Przedstawiona funkcja pozwala również, aby widok wyświetlił dane o autorze oraz kategorii książki. Jest to możliwe, dzięki zależnościom pomiędzy tabelami. W linijce 71 wywoływane są elementy z tabeli AuthorBooks, natomiast linijki 74 oraz 75 wywołują poszczególne elementy z tabel. Funkcja z linijki 76 pozwala na przejście do widoku szczegółów autora, który jest wskazany w wybranej książce.

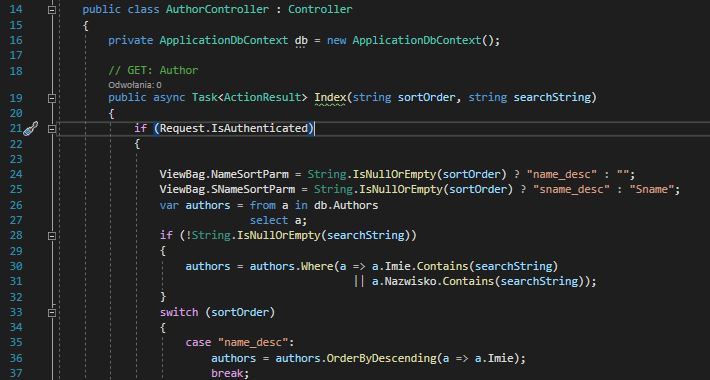


Rys. 29 Fragment kodu pliku Details.cshtml widoku Book

Źródło: opracowanie własne

* 1. **Kontroler**

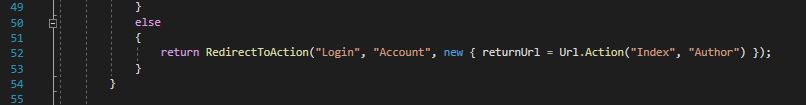
Aby kontekst utworzonej bazy danych był spójny z kontekstem bazy utworzonym według schematu ***ASP .NET***, kontrolery zostały stworzone na podstawie kontekstu szablonu projektu. Celem takiego zabiegu jest, aby dane wprowadzane do tabel Author, Book, Category, były w tej samej bazie, co użytkownicy rejestrujący się do serwisu. Fragment kodu z kontrolera AuthorController znajduje się na Rys. 30. Poniższy kod zawiera również fragment funkcji, która sprawdza czy użytkownik, który chce przeglądać listę autorów jest zalogowany, czy też nie. Odpowiada za to linijka nr 21. W utworzonej aplikacji użytkownik po zalogowaniu ma możliwość sortować dane, odpowiada za to funkcja sortOrder. Sortowanie danych jest możliwe według kolumny Imię oraz Nazwisko. Funkcja searchString, która również jest pokazana na poniższym rysunku, umożliwia użytkownikowi filtrowanie danych. Przedstawione na rysunku funkcje warunkowe if określają, jakie polecenia zostaną wykonane. Zależy to, od tego czy warunek zostanie spełniony czy też nie.



Rys. 30 Fragment kodu AuthorController

Źródło: opracowanie własne

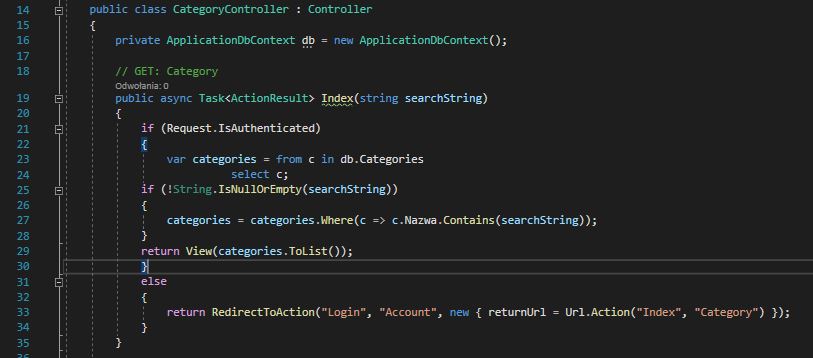
Na poniższym rysunku (Rys. 31) pokazany został fragment kodu, który zamyka funkcję sprawdzania, czy użytkownik jest zalogowany. Jeśli użytkownik nie zalogował się do systemu, zostanie automatycznie przekierowany do ekranu logowania. Funkcja RedirectToAction przekieruje niezalogowanego użytkownika do ekranu logowania. Wskazują na to parametry wewnątrz funkcji: nazwa strony, która ma się uruchomić, nazwa katalogu w sekcji Views, ostatnim parametrem jest informacja, do jakiej strony na powrócić kontroler po udanym zalogowaniu użytkownika.



Rys. 31 Zakończenie funkcji weryfikacji logowania użytkownika

Źródło: opracowanie własne

Poniższy rysunek (Rys. 32) przedstawia zmodyfikowany kod pliku CategoryController.cs. Również znajduje się tutaj weryfikacja, czy użytkownik się zalogował. Występuje w tym kontrolerze tylko możliwość filtrowania kategorii po nazwie. Funkcja znajdująca się w linijkach 25 do 29 sprawdza, czy pole tekstowe jest różne od pustego. Następnie porównuje wprowadzony ciąg znaków, czy pasuje do rekordów znajdujących się w bazie danych.



Rys. 32 Fragment kodu pliku CategoryController.cs

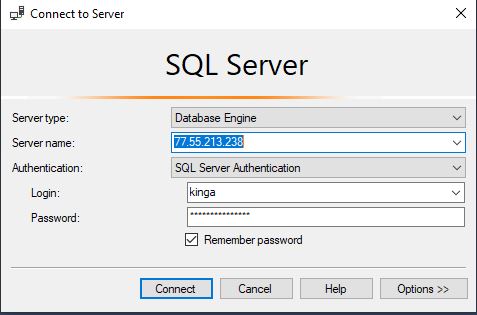
Źródło: opracowanie własne

# **System Internetowego zbioru książek**

W tym rozdziale przedstawiony zostanie sposób działania utworzonej aplikacji. Prezentacja systemu zostanie podzielona na dwa panele: panel administracyjny, w której administrator ma możliwość dodawać dane, oraz panel przeznaczony dla użytkownika, w której zalogowany czytelnik ma możliwość wyszukać wybraną książkę i ją pobrać.

* 1. **Panel administracyjny**

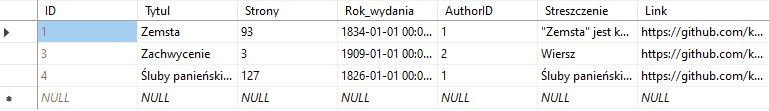
Administrator systemu nie posiada swojego panelu w widoku aplikacji, lecz ma do tego osobne narzędzie. Poniższy rysunek (Rys. 33) przedstawia okno logowania do bazy danych w programie ***SQL Server Management Studio 2017***. Administrator po zalogowaniu się do bazy danych ma możliwość edytowania wprowadzanych rekordów.



Rys. 33 Ekran logowania do zdalnego serwera z bazą danych

Źródło: opracowanie własne

Poniższe rysunki (Rys 34 – Rys. 38) przedstawiają widoki tabel w narzędziu ***SQL Serwer Management Studio 2017***. Pierwszy rysunek tabeli (Rys. 34) zawiera informacje o wprowadzonych książkach do bazy. Następny rysunek (Rys. 35) przedstawia widok tabeli z danymi o autorach książek. Na Rys. 36 przedstawiony został widok tabeli Category. Następne dwa rysunki (Rys. 37 oraz Rys. 38) przedstawiają widoki tabel pośrednich, AuthorBooks oraz CategoryBooks.



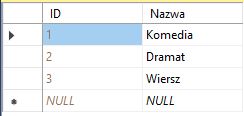
Rys. 34 Widok edycji rekordów w tabeli Books

Źródło: opracowanie własne



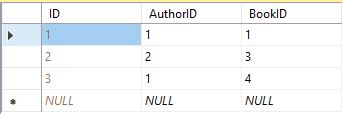
Rys. 35 Widok tabeli Authors

Źródło: opracowanie własne



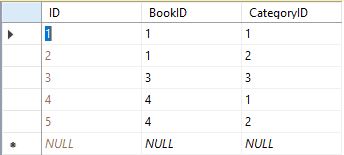
Rys. 36 Widok tabeli Categories

Źródło: opracowanie własne



Rys. 37 Widok tabeli pośredniej AuthorBooks

Źródło: opracowanie własne

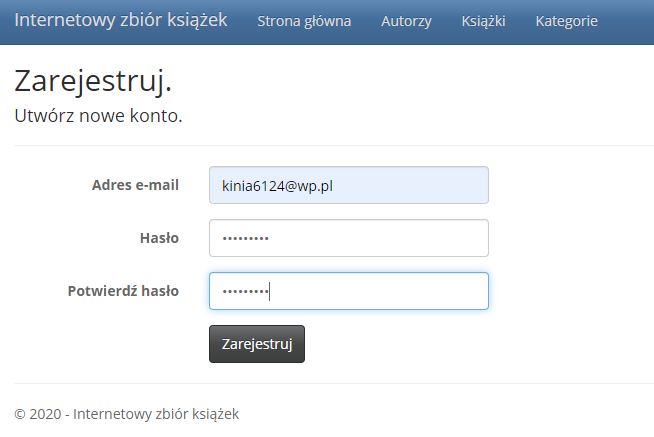


Rys. 38 Widok tabeli pośredniej CategoryBooks

Źródło: opracowanie własne

* 1. **Panel użytkownika**

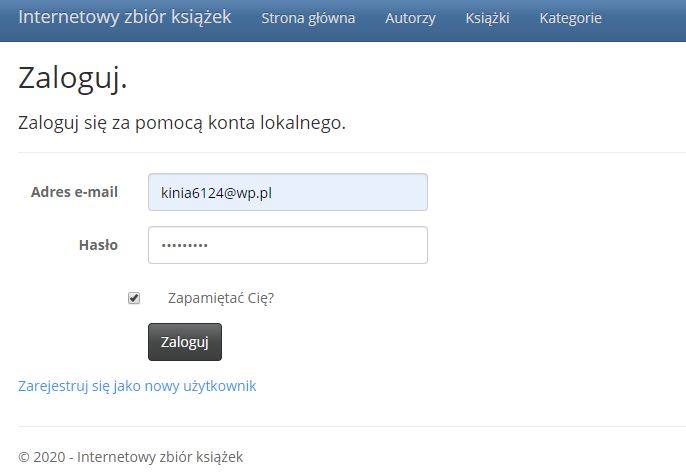
Aby czytelnik mógł korzystać z aplikacji internetowej musi się najpierw zarejestrować do systemu. Ekran rejestracji przedstawia Rys. 39. Do rejestracji wymagany jest adres e-mail, hasło oraz dla celów walidacyjnych powtórzenie hasła.



Rys. 39 Ekran rejestracji do systemu

Źródło: opracowanie własne

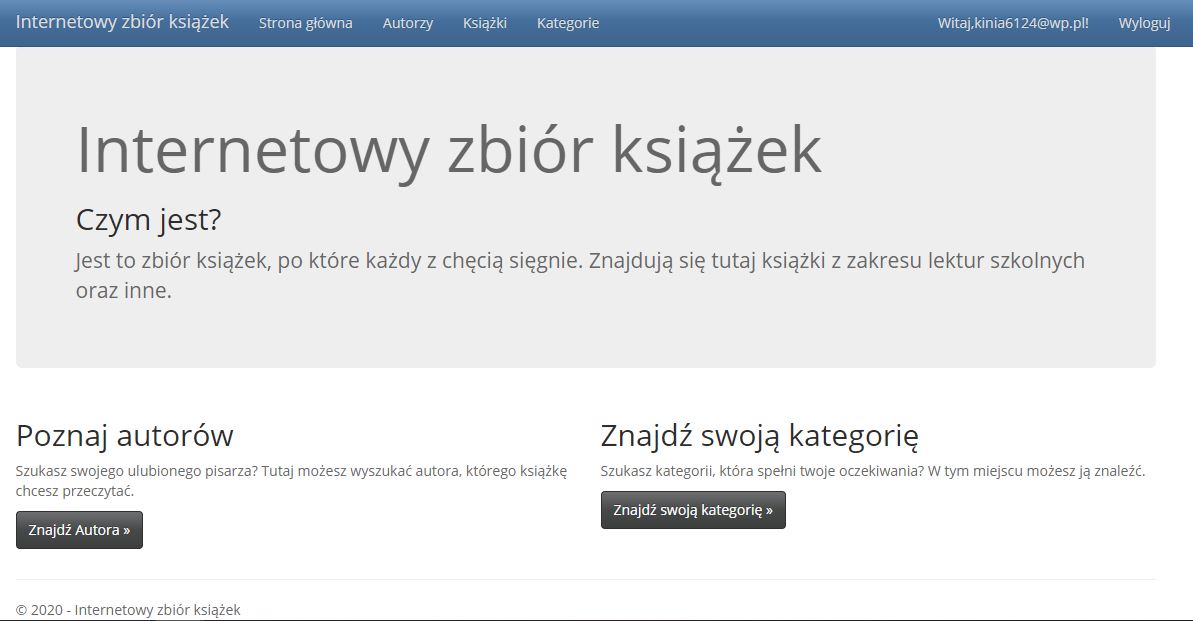
Po zarejestrowaniu użytkownik zostaje automatycznie przekserowany do strony głównej aplikacji (Rys.41). Jeśli użytkownik chce się zalogować to musi podać dane takie jak: adres e-mail oraz hasło podane podczas rejestracji. Ekran logowania został przedstawiony na Rys. 30. Użytkownik ma możliwość zaznaczenia opcji zapamiętania. Oznacza to, że przy kolejnym logowaniu, jego dane logowania zostaną automatycznie uzupełnione. Aplikacja została utworzona za pomocą ***Bootstrapa***, dzięki czemu użytkownik może ją uruchomić na przeglądarce w smartfonie. Widoki aplikacji w przeglądarce komputerowej zostały przedstawione na Rys. 40 – Rys. 47. Natomiast Rys. 48 – Rys. 53 zawierają widoki aplikacji uruchomionej w przeglądarce na smartfonie.



Rys. 40 Ekran logowania

Źródło: opracowanie własne

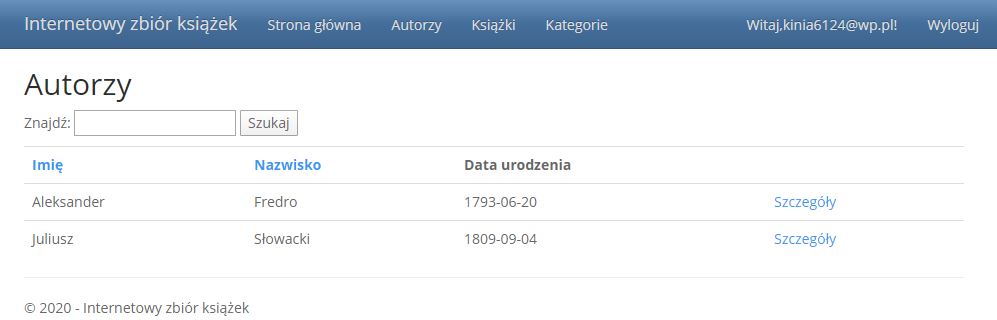
Strona główna jest kolejnym widokiem aplikacji jaki widzi użytkownik (Rys. 41). Na ekranie widoczne są ogólne informacje, w tym informacja o stronie oraz możliwość wyświetlanie listy autorów, książek, kategorii za pomocą paska menu, widocznego u góry strony.



Rys. 41 Strona główna aplikacji

Źródło: opracowanie własne

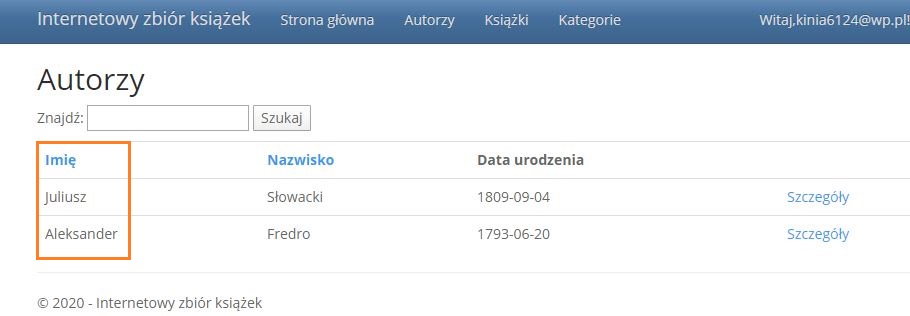
Na poniższym rysunku (Rys. 42) umieszczony został widok listy autorów. Podświetlone kolumny pozwalają na sortowanie danych. Natomiast panel wyszukiwarki pozwoli na filtrowanie danych.



Rys. 42 Lista Autorów

Źródło: opracowanie własne

Poniższe rysunki (Rys. 43) przedstawia listę autorów posortowaną malejąco według imion (Z-A).



Rys. 43 Posortowana lista autorów

Źródło: opracowanie własne

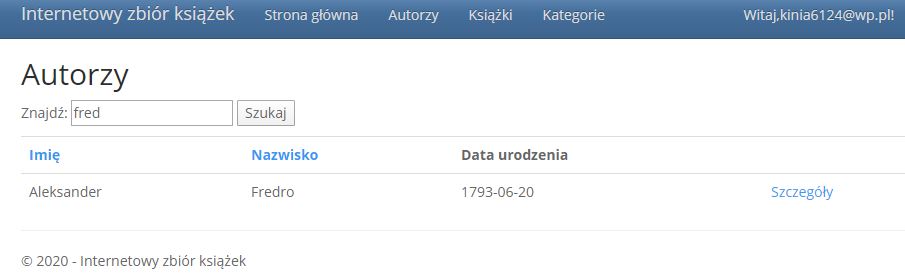
Potwierdzeniem pomyślnego posortowania listy, poniżej znajduje się rysunek zawierający pasek adresu (Rys. 44). Wskazany w nim jest typ sortowania listy.



Rys. 44 Pasek adresu posortowanego widoku

Źródło: opracowanie własne

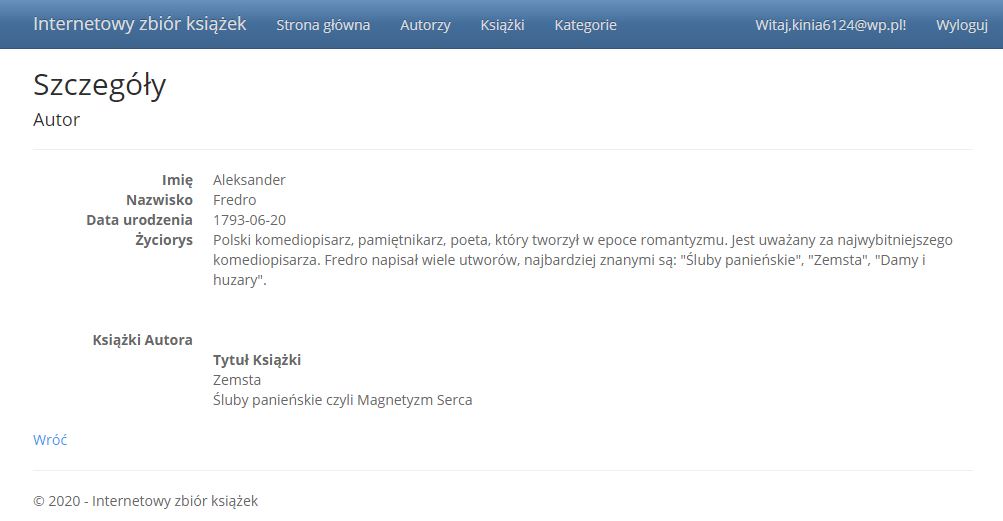
Poniższy rysunek (Rys. 45) przedstawia funkcjonalność filtrowania danych. Jest to możliwe po wpisaniu w polu tekstowym fragmentu Imienia bądź Nazwiska autora, którego chcemy odszukać.



Rys. 45 Filtrowanie danych

Źródło: opracowanie własne

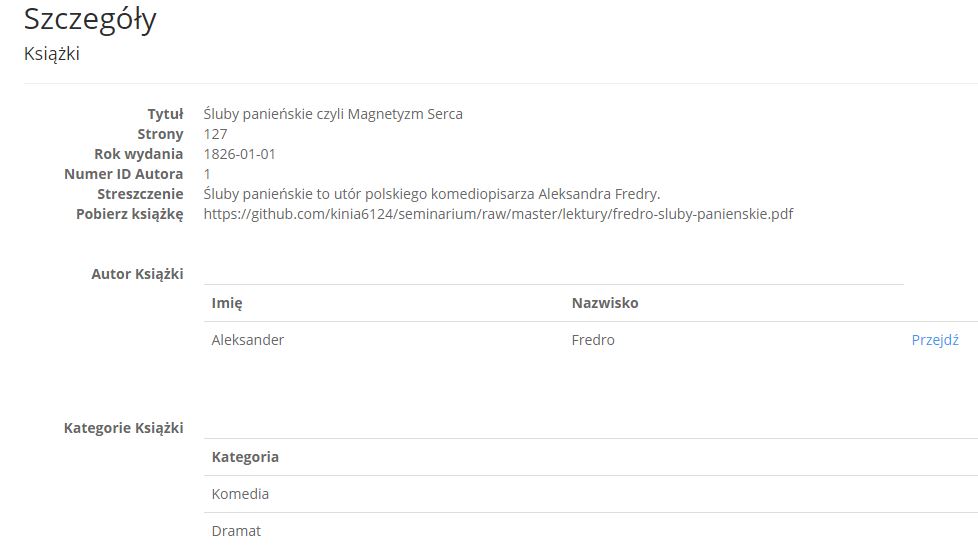
Poniższy rysunek (Rys. 35) przedstawia widok szczegółów wybranego autora. Widać również listę jakie książki napisał autor.



Rys. 46 Widok szczegółów wybranego autora

Źródło: opracowanie własne

Rys. 47 przedstawia szczegóły wybranej książki. Na tym ekranie widać również informacje o autorze książki oraz o kategorii do jakiej została przypisana wybrana książka.



Rys. 47 Szczegóły wybranej książki

Źródło: opracowanie własne

Na poniższych rysunkach (Rys. 48 – Rys. 53) zostały przedstawione widoki aplikacji, uruchomione w przeglądarce na smartfonie. Należy zauważyć, iż wykorzystany podczas tworzenia systemu ***Bootstrap***, dopasowuje widok uruchomionej aplikacji do wielkości ekranu, na którym ma zostać wyświetlona.

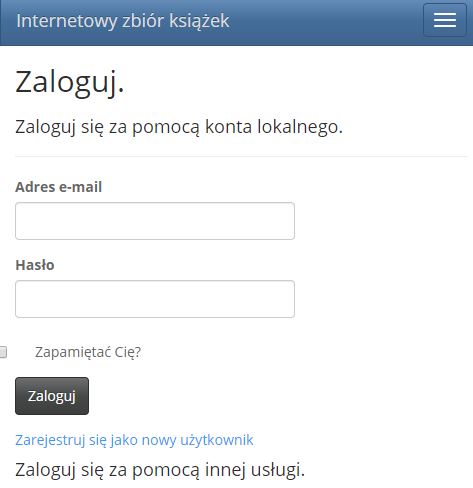
Poniższy rysunek (Rys. 48) przedstawia widok menu aplikacji.



Rys. 48 Widok menu aplikacji

Źródło: opracowanie własne

Na poniższym rysunku (Rys. 49) przedstawiony został ekran logowania do systemu.



Rys. 49 Logowanie do systemu

Źródło: opracowanie własne

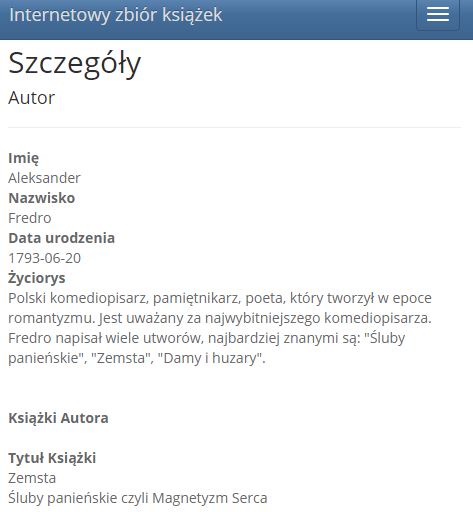
Rys. 50 przedstawia listę autorów dodanych do bazy systemu.



Rys. 50 Lista autorów

Źródło: opracowanie własne

Na poniższym rysunku (Rys. 51) widać szczegóły wybranego autora.



Rys. 51 Widok szczegółów autora

Źródło: opracowanie własne

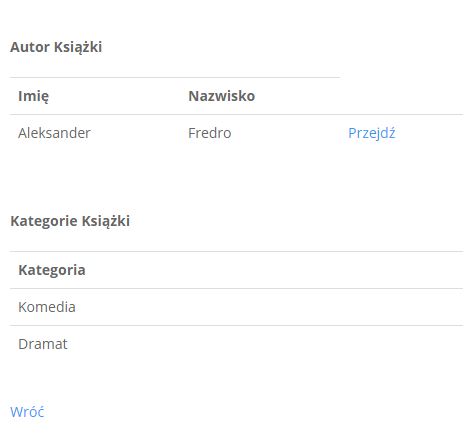
Rys. 52 oraz Rys. 53 przedstawiają ekran szczegółów wybranej książki w wersji mobilnej. Można zauważyć jak ***Bootstrap*** dopasowuje wygląd strony do rozdzielczości ekranu.



Rys. 52 Szczegóły książek

Źródło: opracowanie własne

Rysunek (Rys. 53) poniżej przedstawia również autora oraz kategorie książki.



Rys. 53 Szczegóły książki

Źródło: opracowanie własne

# **Streszczenie**

**Wyższa Szkoła Informatyki i Zarządzania z siedzibą w Rzeszowie**

**Kolegium Informatyki Stosowanej**

**Streszczenie pracy dyplomowej**

Internetowy zbiór książek

**Autor:** Kinga Pszeniczna

**Promotor:** dr inż. Łukasz Piątek

**Słowa kluczowe:** aplikacje internetowe, ASP .NET, C#. Bootstrap

W ramach pracy utworzono aplikację internetową pozwalającą na przeglądanie informacji o autorach oraz książkach, z możliwością pobrania wybranej książki. Strona została napisana w technologii ASP .NET MVC 5 z wykorzystaniem Bootstrapa, dzięki któremu strona jest responsywna. Do przechowywania danych wykorzystano narzędzie bazodanowe SQL Server 2017 Express. Zarządzanie danymi odbywa się dzięki narzędziu SQL Server Management Studio. Aplikacja została napisana w środowisku Visual Studio Community 2019. Książki wykorzystane do projektu są wolne od praw autorskich.

Poza częścią implementacyjną jest część teoretyczna, w której opisane zostało wprowadzenie do aplikacji internetowych. W części teoretycznej przedstawione zostały podstawowe technologie pozwalające na tworzenie aplikacji internetowych. Opisana została aktualna wiedza na temat aplikacji internetowych oraz krótka specyfikacja utworzonego systemu.

System został zaimplementowany zgodnie z założeniami postawionymi przed jego realizacją.

Zaimplementowany system posiada praktyczne zastosowanie. Po wprowadzeniu ewentualnych zmian aplikacja może zostać wdrożona na rynek komercyjny.

# **Literatura**

Dmitri Nesteruk, *Wzorce projektowe w .NET. Projektowanie zorientowane obiektowo z wykorzystaniem C# i F#,* Helion, 2019

Frahaan Hussain, *Responsive Web Design. Nowoczesne strony WWW na przykładach*, Helion, 2019

Jamie Munro, *ASP.NET MVC 5, Bootstrap i Knockout.js. Tworzenie dynamicznych i elastycznych aplikacji internetowych,* Helion, 2015

Marcin Lis, *C#. Praktyczny kurs. Wydanie III,* Helion, 2016

**Spis stron WWW**

http://lukaszkosiorowski.pl/programowanie/net/asp-mvc/kurs-asp-mvc-5/(16.01.2020)

http://www.php.pl/Wortal/Artykuly/Framework/Frameworki-dla-PHP-czyli-wydajne-tworzenie-aplikacji/MVC(03.02.2020)

https://devszczepaniak.pl/wstep-do-rest-api/(28.01.2020)

https://docs.microsoft.com/pl-pl/visualstudio/ide/whats-new-visual-studio-2019?view=vs-2019(17.02.2020)

https://javastart.pl/baza-wiedzy/wprowadzenie/eclipse-ide-wstep-i-instalacja(17.02.2020)

https://psd2css.pl/baza-wiedzy/co-to-jest-bootstrap(16.01.2020)

https://wiedzainformatyczna.pl/asp-net-entity-framework/#.Xk7zayhKjIW(16.01.2020)

https://www.java.com/pl/download/faq/whatis\_java.xml(17.02.2020)

https://www.spidersweb.pl/rozrywka/2019/03/25/czytelnictwo-w-polsce-2018-raport-opinia/(13.01.2020)

<https://www.devmobile.pl/rest-api-od-podstaw/(28.02.2020)>

<http://www.galaxy.agh.edu.pl/~pamalino/programowanie/mysql/img/phpMyAdmin3.png(28.02.2020)>

<https://www.instalki.pl/programy/download/Windows/bazy_danych/Navicat_for_MongoDB.html>(28.02.2020)

# **Spis rysunków**

Rys. 1 Katalog online Biblioteki Miejskiej w Mielcu 5

Rys. 2 Widok wybranej książki w serwisie lubimyczytać.pl 5

Rys. 3 Schemat działania protokołu HTTP 6

Rys. 4 Podstawowa struktura kodu języka HTML 7

Rys. 5 Znacznik <p> 7

Rys. 6 Rezultat uruchomienia strony z tekstem w znaczniku <p> 7

Rys. 7 Odnośnik do pliku style.css 8

Rys. 8 Treść pliku style.css 8

Rys. 9 Rezultat modyfikacji kodu 8

Rys. 10 Schemat działania Entity Framework 9

Rys. 11 Środowisko programistyczne Visual Studio 2019 12

Rys. 12 Środowisko programistyczne Visual Studio Code 12

Rys. 13 Środowisko programistyczne Eclipse 13

Rys. 14 Przykładowa tabela w bazie MySQL 14

Rys. 15 Widok kolekcji w bazie MongoDB 15

Rys. 16 Wykres ilości czytanych książek w latach 2000 do 2018 16

Rys. 17 Źródła pozyskiwania książek w latach 2014 do 2018 17

Rys. 18 Schemat wzorca MVC 18

Rys. 19 Przykład wyszukanej książki w katalogu Biblioteki Miejskiej w Mielcu 19

Rys. 20 Wyszukana książka w serwisie lubimyczytac.pl 19

Rys. 21 Fragment kodu odpowiadający za połączenie z bazą danych 22

Rys. 22 Model bazy tabeli Author 22

Rys. 23 Model tabeli AuthorBook 23

Rys. 24 Fragment kodu CzytelniaContext 23

Rys. 25 Diagram zależności między tabelami 24

Rys. 26 Fragment kodu odpowiedzialnego za pasek menu 24

Rys. 27 Fragment kodu pliku Index.cshtml 25

Rys. 28 Fragment kodu pliku Details.cshtml 26

Rys. 29 Fragment kodu pliku Details.cshtml widoku Book 26

Rys. 30 Fragment kodu AuthorController 27

Rys. 31 Zakończenie funkcji weryfikacji logowania użytkownika 27

Rys. 32 Fragment kodu pliku CategoryController.cs 28

Rys. 33 Ekran logowania do zdalnego serwera z bazą danych 29

Rys. 34 Widok edycji rekordów w tabeli Books 29

Rys. 35 Widok tabeli Authors 29

Rys. 36 Widok tabeli Categories 30

Rys. 37 Widok tabeli pośredniej AuthorBooks 30

Rys. 38 Widok tabeli pośredniej CategoryBooks 30

Rys. 39 Ekran rejestracji do systemu 30

Rys. 40 Ekran logowania 31

Rys. 41 Strona główna aplikacji 31

Rys. 42 Lista Autorów 32

Rys. 43 Posortowana lista autorów 32

Rys. 44 Pasek adresu posortowanego widoku 32

Rys. 45 Filtrowanie danych 32

Rys. 46 Widok szczegółów wybranego autora 33

Rys. 47 Szczegóły wybranej książki 33

Rys. 48 Widok menu aplikacji 34

Rys. 49 Logowanie do systemu 34

Rys. 50 Lista autorów 35

Rys. 51 Widok szczegółów autora 35

Rys. 52 Szczegóły książek 36

Rys. 53 Szczegóły książki 36

# **Załączniki**

Załącznikiem do pracy jest płyta CD z kodem źródłowym.

1. https://www.devmobile.pl/rest-api-od-podstaw/[28.02.2020] [↑](#footnote-ref-1)
2. https://www.java.com/pl/download/faq/whatis\_java.xml[17.02.2020] [↑](#footnote-ref-2)