Biblioteka

Wykonawcy:

Marcin Bogdan

Przemysław Zawadzki

Semestr V

Grupa IV

Spis treści

[1. Wstęp 3](#_Toc468966501)

[2. Analiza wymagań systemu 3](#_Toc468966502)

[2.1 Wymagania funkcjonalne 3](#_Toc468966505)

[2.2 Wymagania niefunkcjonalne 3](#_Toc468966506)

[2.3 Diagram przypadków użycia 5](#_Toc468966507)

[3. Wykorzystane technologie 6](#_Toc468966509)

[4. Projekt i implementacja aplikacji 6](#_Toc468966510)

[4.1 Architektura aplikacji 6](#_Toc468966513)

[4.2 Projekt koncepcyjny bazy danych 7](#_Toc468966514)

[4.3 Projekt schematu relacyjnego 8](#_Toc468966515)

[4.4 Modelowanie relacji w encjach 8](#_Toc468966516)

[5. Funkcjonalność aplikacji 8](#_Toc468966521)

[6. Interfejs użytkownika 9](#_Toc468966522)

[7. Podsumowanie 11](#_Toc468966529)

[Dodatek A: Skrypty tworzące obiekty baz danych 12](#_Toc468966530)

# Wstęp

Tematem projektu jest serwis internetowy wypożyczalni książek. Wypożyczalnia umożliwia przeglądanie listy książek, dodawanie książek do koszyka i w efekcie ich wypożyczenie. Aplikacja przeznaczona jest dla trzech typów użytkowników:

* Administrator – może zarządzać zawartością wypożyczalni i użytkownikami korzystającymi z serwisu;
* Pracownik – te uprawnienia pozwalają na edycje zawartości biblioteki;
* Zwykły użytkownik – osoba która założyła konto w serwisie aby móc korzystać z zawartości biblioteki, rezerwować i wypożyczać książki.

Serwis został zrealizowany w języku C# z wykorzystaniem technologii .NET MVC.

# Analiza wymagań systemu



## Wymagania funkcjonalne

Serwis powinien pozwalać na wyświetlanie listy książek. Katalog książek powinien być zarządzany z poziomu panelu administracyjnego. Serwis powinien pozwalać na rejestrację użytkowników za pomocą odpowiedniego formularza dostępnego dla niezalogowanych osób wchodzących do serwisu. Tylko zalogowani użytkownicy powinni mieć dostęp do możliwości wypożyczenia książek.

Pracownicy wypożyczalni powinni mieć odpowiednie konta dzięki którym mogą edytować zbiór książek dostępnych w wypożyczalni, czytelnikami biblioteki, a także ich wypożyczeniami. Pracownikami powinien zarządzać administrator.

Książki powinny być kategoryzowane za pomocą predefiniowanej listy kategorii możliwej do edycji przez administratora jak i przez pracowników. Do każdej książki powinna istnieć możliwość dodania okładki, opisu i informacji potrzebnych do zidentyfikowania egzemplarza. W aplikacji powinien istnieć mechanizm wyszukiwania na podstawie tytułu, nazwy autora, numeru ISBN.

Użytkownik powinien posiadać koszyk w którym może składować książki które chce wypożyczyć. Opcja wypożyczenia powinna być dostępna z poziomu widoku koszyka. Po wypożyczeniu jak i zwrocie stan książek w bibliotece powinien być aktualizowany.

Jednorazowe wypożyczenie powinno być ustawione na jeden miesiąc od kliknięcia wypożycz.

## Wymagania niefunkcjonalne

Wymagania dotyczące sprzętu użytkownika korzystającego z serwisu są niewielkie, do prawidłowej pracy potrzebne jest urządzenie z dostępem do Internetu i zainstalowaną przeglądarką internetową. Aplikacja dostosowuje się do wielkości ekranu urządzenia, więc korzystanie z niej może odbywać się również poprzez telefony komórkowe.

Wymagania dotyczące technologii: program został napisany w C# korzystając z szablonów Bootstrap, technologii .NET MVC i bazy danych Microsoft SQL Server. Do komunikacji z bazą danych wykorzystywany jest Entity Framework.

Baza danych powinna zostać zainicjalizowana na serwerze bazodanowym poprzez skrypt tworzący obiekty bazy danych dołączony do dokumentacji (Dodatek A).

Schemat bazy danych powinien zawierać tabele odpowiedzialne za przechowywanie danych o użytkownikach i ich rolach w systemie. Hasła powinny być szyfrowane. Wszystkie informacje o książkach znajdujących się w bibliotece powinny znajdować się w odpowiednich tabelach bazy danych. Zdjęcia poglądowe książek powinny być zapisywane w odpowiednich folderach na serwerze aplikacji.

Bezpieczeństwo aplikacji na serwerze nie było celem pracy, zastosowano tylko weryfikację pól formularzy, uniemożliwienie wprowadzania specjalnych znaków do Microsoft SQL Server.

## Diagram przypadków użycia

# C:\Users\przemyslawz\Downloads\15401287_10207833527304503_1527117293_n.png

# Wykorzystane technologie

* ASP.NET MVC
* Entity Framework
* Bootstrap
* JavaScript
* jQuery
* LightBox
* Database Migration

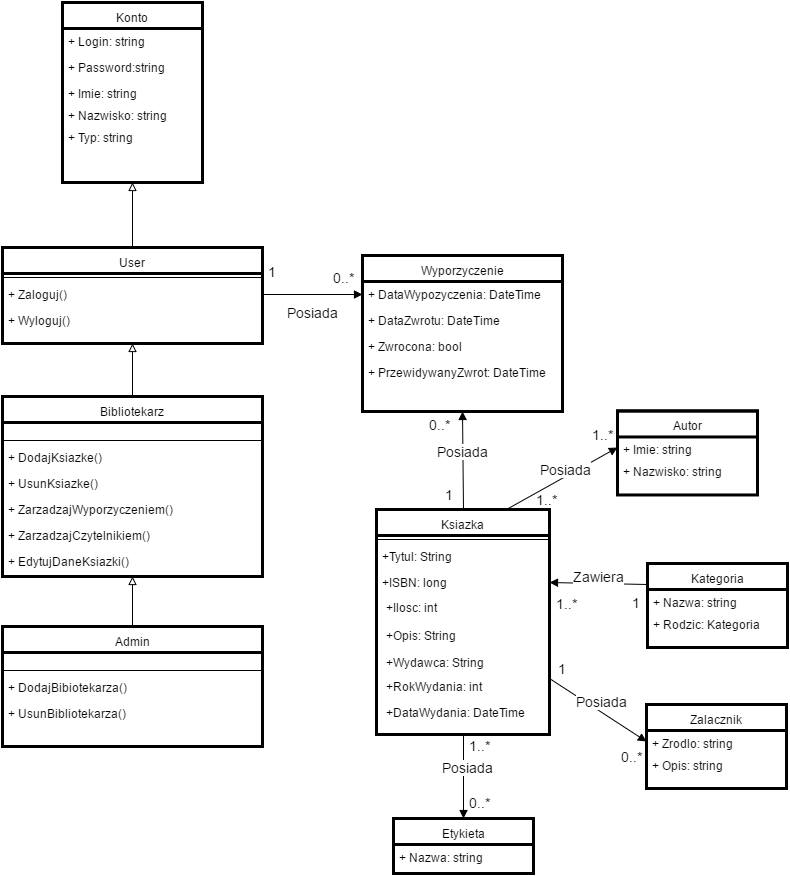
# Projekt i implementacja aplikacji



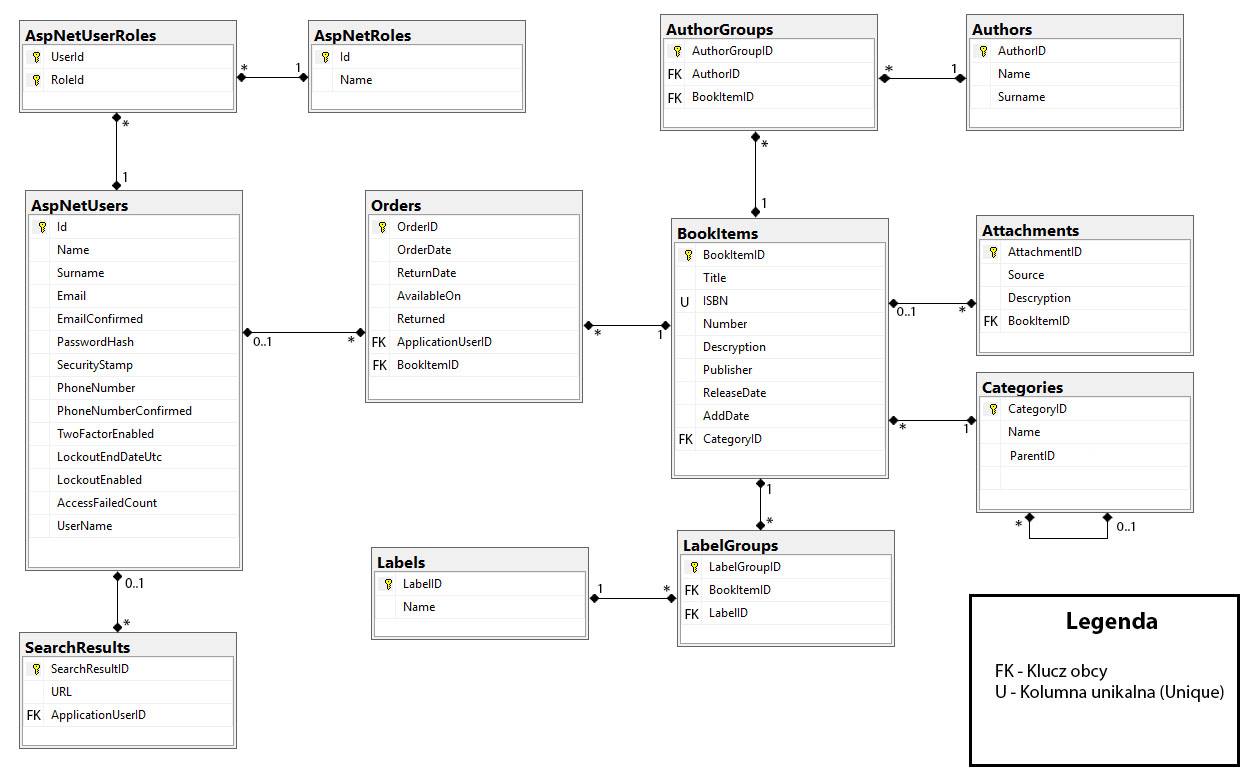
## Architektura aplikacji

Aplikacja została stworzona zgodnie z zasadami .NET MVC. Klasy modelowe odpowiedzialne za obiektową reprezentacje bazy danych są oddzielone od reszty aplikacji. Za przetwarzanie i odbieranie danych odpowiedzialne są kontrolery w których znajduje się logika aplikacji. Kontrolery te wysyłają dane do widoków które nimi są wypełniane.

## Projekt koncepcyjny bazy danych



## Projekt schematu relacyjnego



## Modelowanie relacji w encjach

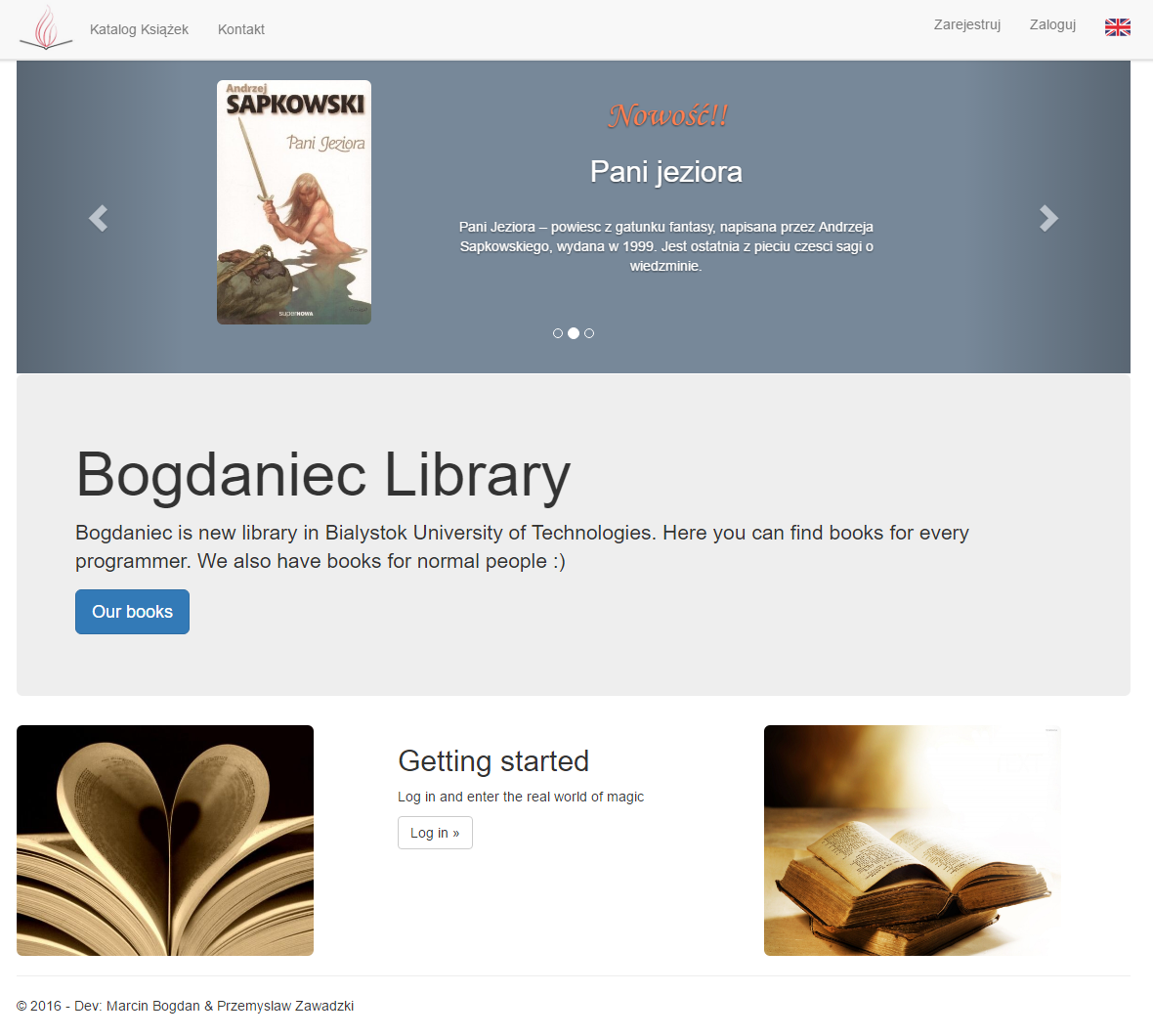
Wykorzystano podejście Code-First gdzie na podstawie klas zostały stworzone tabele bazy danych. Połączenie pomiędzy tabelami odbywa się poprzez obiekty oraz kolekcje wirtualne. W połączeniu jeden do wielu obiekt bazodanowy posiadający wiele powiązanych rekordów w innych obiektach posiada odwołanie do nich poprzez kolekcje. Obiekt który ma powiązanie z jednym rekordem posiada referencję na ten obiekt. Metody wirtualne wykorzystują leniwe dociąganie danych, tzn. dane pobierane są dopiero w momencie odwołania do nich w kodzie aplikacji.



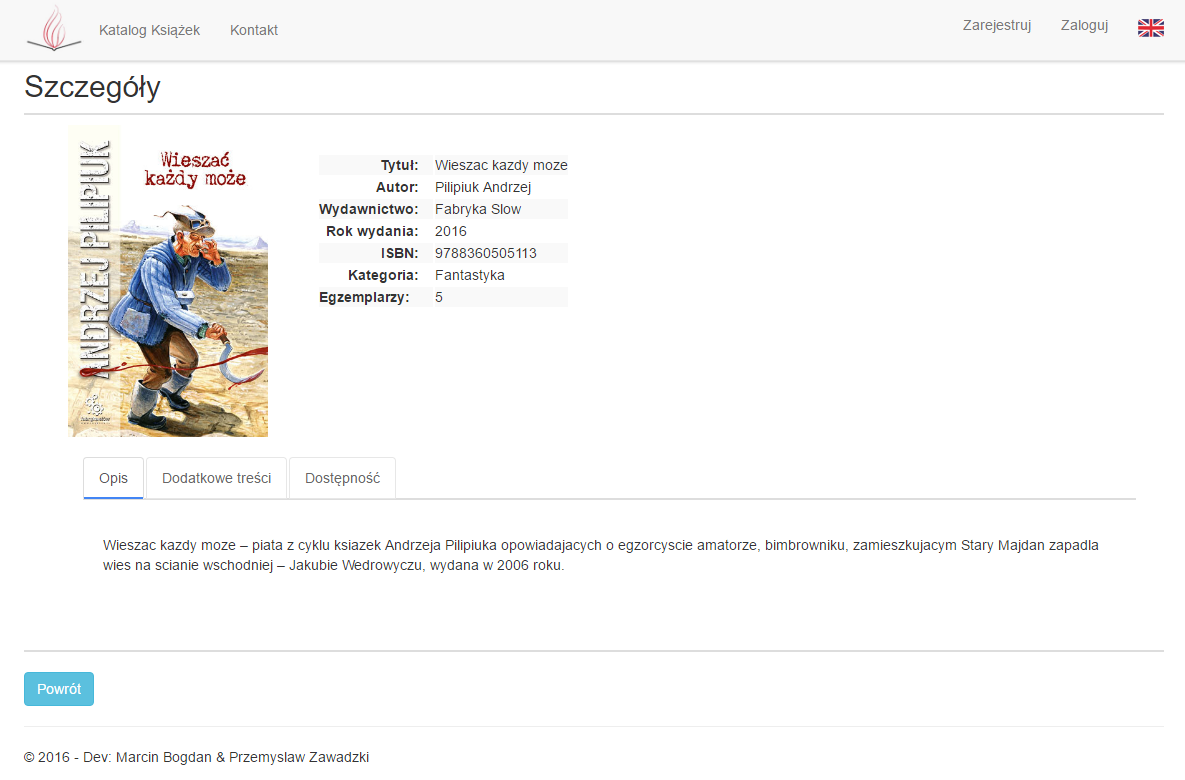
# Funkcjonalność aplikacji

Serwis pozwala na przeglądanie i wypożyczanie książek dostępnych w bazie biblioteki. Użytkownicy po uprzednim zalogowaniu mają dostęp do informacji o książkach, a także o ilość egzemplarzy dostępnych do wypożyczenia. Administrator ma możliwość edycji uprawnień użytkowników, a także zasobów biblioteki. W serwisie istnieją specjalne typy konta dla pracowników dzięki czemu mogą oni przeglądać i zarządzać stanami rezerwacji użytkowników, oraz inwentarzem biblioteki. Na stronie głównej wyświetlane są trzy ostatnio dodane książki w formie karuzeli.

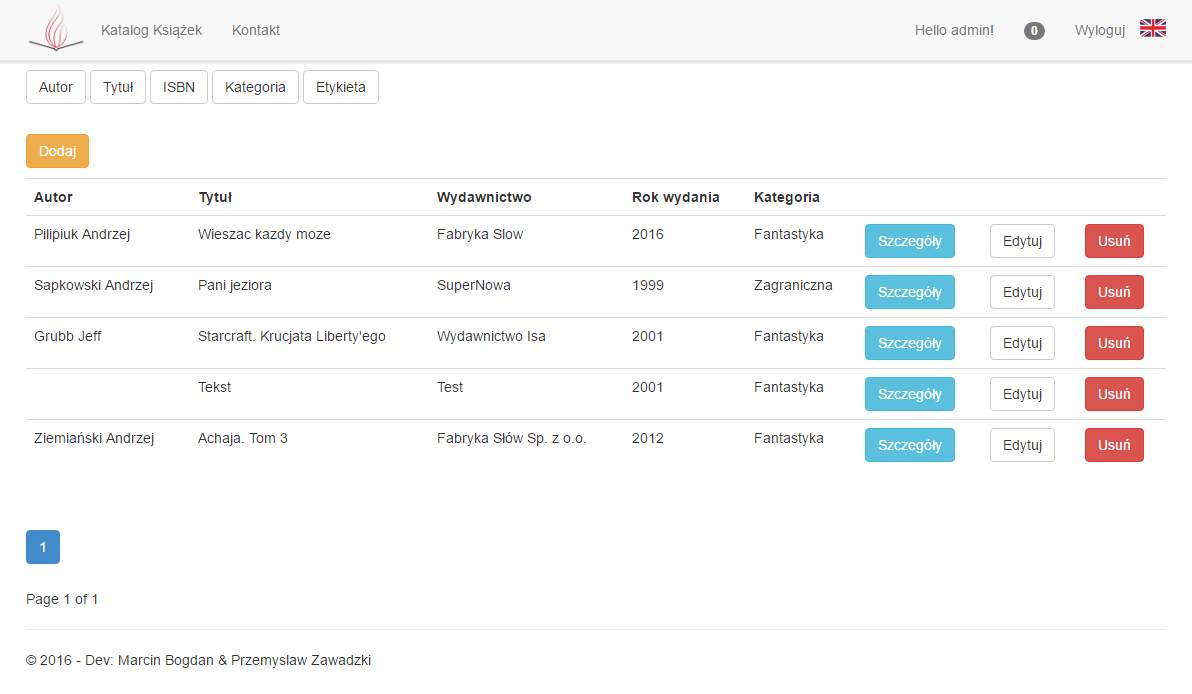
# Interfejs użytkownika



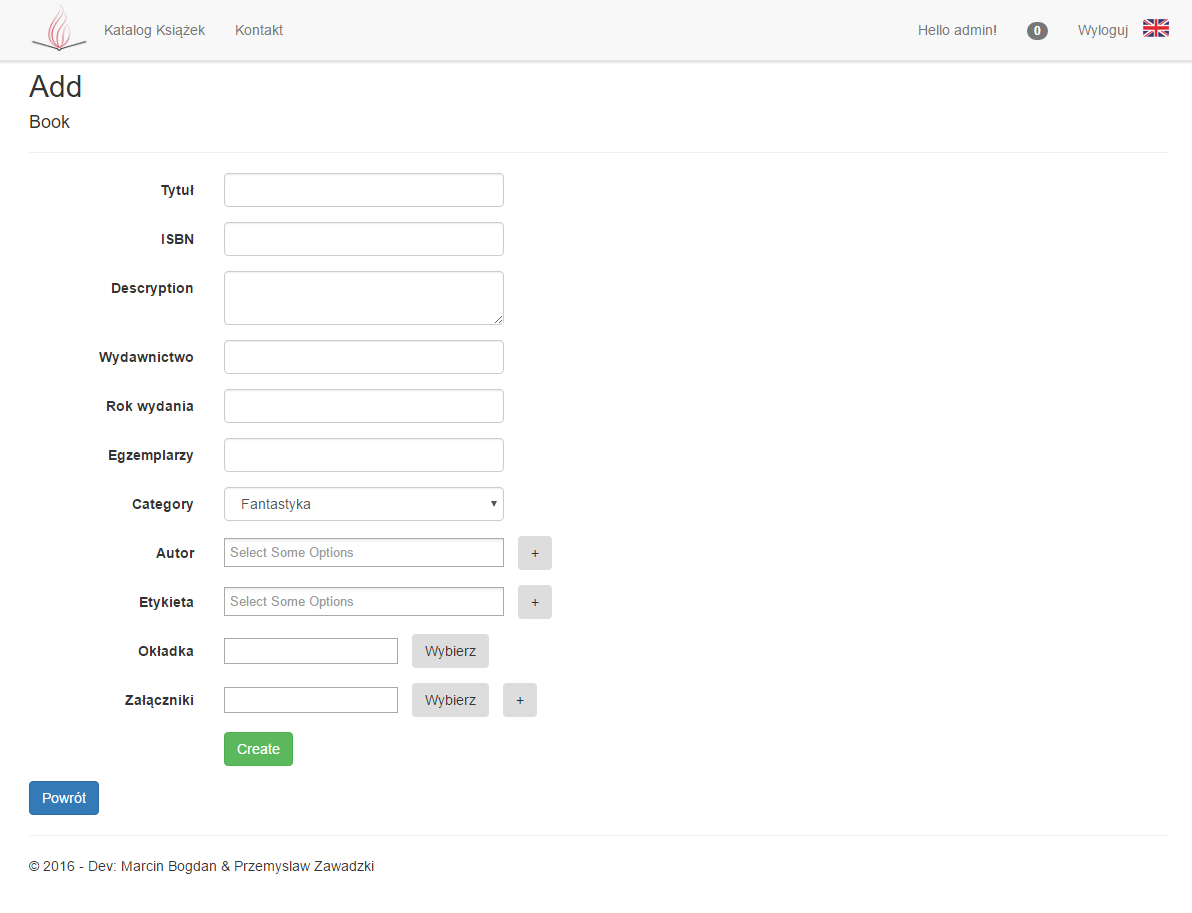
Strona główna



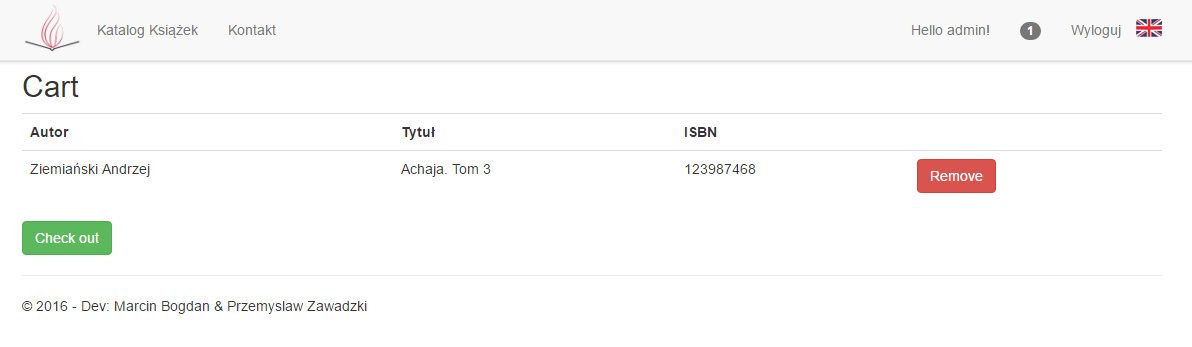
Szczegóły książki



Lista dostępnych książek z pozimu Administratora



Dodawanie ksiązki



Koszyk zamówienia



# Podsumowanie

Zaczynając pracę nad projektem posiadaliśmy podstawową wiedzę oraz umiejętności związane z tworzeniem serwisów internetowych oraz językiem C#. Dobra znajomość oprogramowania Visual Studio była atutem. Jednak w trakcie realizacji okazało się, że to nie wystarczy. Podczas całego procesu tworzenia poznaliśmy wiele ciekawych i przyspieszających pracę rozwiązań. Wykorzystanie takich technologii jak JQuery, Bootstrap, czy też JavaScript pozwoliło osiągnąć estetycznie wyglądające, jak również bardziej funkcjonalne widoki stron.

Największymi problemami w tworzeniu aplikacji były problemy natury koncepcyjnej oraz związane z implementacją niektórych funkcjonalności. Niekiedy konieczna była zmiana w budowie tabel, jednakże dzięki wykorzystaniu systemu migracji wprowadzenie takich zmian było ułatwione, gdyż w każdym momencie istnieje możliwość powrotu do poszczególnych stanów bazy danych dzięki kontroli wersji.

Aplikacja jest w pełni funkcjonalna i może zostać zaimplementowana do użytku prawdziwej biblioteki. Ułatwia ona procesem zarządzania stanem biblioteki. Dzięki niej klienci mają szybki dostęp do sprawdzenia dostępności wybranych pozycji a także możliwości ich wypożyczenia.

# Dodatek A: Skrypty tworzące obiekty baz danych

/\*\*\*\*\*\* Table AspNetRoles \*\*\*\*\*\*/

CREATE TABLE AspNetRoles(

Id nvarchar(128) NOT NULL,

Name nvarchar(256) NOT NULL,

CONSTRAINT [PK\_AspNetRoles] PRIMARY KEY (Id))

GO

/\*\*\*\*\*\* Table AspNetUserRoles \*\*\*\*\*\*/

CREATE TABLE AspNetUserRoles(

UserId nvarchar(128) NOT NULL,

RoleId nvarchar(128) NOT NULL,

CONSTRAINT [PK\_AspNetUserRoles] PRIMARY KEY (UserId, RoleId))

GO

/\*\*\*\*\*\* Table AspNetUsers \*\*\*\*\*\*/

CREATE TABLE AspNetUsers(

Id nvarchar(128) NOT NULL,

Name nvarchar(max) NOT NULL,

Surname nvarchar(max) NOT NULL,

Email nvarchar(256) NULL,

EmailConfirmed bit NOT NULL,

PasswordHash nvarchar(max) NULL,

SecurityStamp nvarchar(max) NULL,

PhoneNumber nvarchar(max) NULL,

PhoneNumberConfirmed bit NOT NULL,

TwoFactorEnabled bit NOT NULL,

LockoutEndDateUtc datetime NULL,

LockoutEnabled bit NOT NULL,

AccessFailedCount int NOT NULL,

UserName nvarchar(256) NOT NULL,

CONSTRAINT [PK\_AspNetUsers] PRIMARY KEY (Id))

GO

/\*\*\*\*\*\* Table AuthorGroups \*\*\*\*\*\*/

CREATE TABLE AuthorGroups(

AuthorGroupID int IDENTITY(1,1) NOT NULL,

AuthorID int NOT NULL,

BookItemID int NOT NULL,

CONSTRAINT [PK\_AuthorGroups] PRIMARY KEY (AuthorGroupID))

GO

/\*\*\*\*\*\* Table Authors \*\*\*\*\*\*/

CREATE TABLE Authors(

AuthorID int IDENTITY(1,1) NOT NULL,

Name nvarchar(max) NULL,

Surname nvarchar(max) NULL,

CONSTRAINT [PK\_Authors] PRIMARY KEY (AuthorID))

GO

/\*\*\*\*\*\* Table BookItems \*\*\*\*\*\*/

CREATE TABLE BookItems(

BookItemID int IDENTITY(1,1) NOT NULL,

Title nvarchar(max) NULL,

ISBN bigint NOT NULL,

Number int NOT NULL,

Descryption nvarchar(max) NULL,

Publisher nvarchar(max) NULL,

ReleaseDate int NOT NULL,

AddDate datetime NOT NULL,

CategoryID int NOT NULL,

CONSTRAINT [PK\_BookItems] PRIMARY KEY (BookItemID))

GO

/\*\*\*\*\*\* Table Categories \*\*\*\*\*\*/

CREATE TABLE Categories(

CategoryID int IDENTITY(1,1) NOT NULL,

Name nvarchar(max) NULL,

Parent\_CategoryID int NULL,

ParentID int NOT NULL,

CONSTRAINT [PK\_Categories] PRIMARY KEY (CategoryID))

GO

/\*\*\*\*\*\* Table LabelGroups \*\*\*\*\*\*/

CREATE TABLE LabelGroups(

LabelGroupID int IDENTITY(1,1) NOT NULL,

BookItemID int NOT NULL,

LabelID int NOT NULL,

CONSTRAINT [PK\_LabelGroups] PRIMARY KEY (LabelGroupID))

GO

/\*\*\*\*\*\* Table Labels \*\*\*\*\*\*/

CREATE TABLE Labels(

LabelID int IDENTITY(1,1) NOT NULL,

Name nvarchar(max) NULL,

CONSTRAINT [PK\_Labels] PRIMARY KEY (LabelID))

GO

/\*\*\*\*\*\* Table Orders \*\*\*\*\*\*/

CREATE TABLE Orders(

OrderID int IDENTITY(1,1) NOT NULL,

OrderDate datetime NOT NULL,

ReturnDate datetime NOT NULL,

AvailableOn datetime NOT NULL,

Returned bit NOT NULL,

ApplicationUserID nvarchar(128) NULL,

BookItemID int NOT NULL,

CONSTRAINT [PK\_Orders] PRIMARY KEY (OrderID))

GO

/\*\*\*\*\*\* Table SearchResults \*\*\*\*\*\*/

CREATE TABLE SearchResults(

SearchResultID int IDENTITY(1,1) NOT NULL,

URL nvarchar(max) NULL,

ApplicationUserID nvarchar(128) NULL,

CONSTRAINT [PK\_SearchResults] PRIMARY KEY (SearchResultID))

GO

/\*\*\*\*\*\* Table Attachments \*\*\*\*\*\*/

CREATE TABLE Attachments(

AttachmentID int IDENTITY(1,1) NOT NULL,

Source nvarchar(max) NULL,

Descryption nvarchar(max) NULL,

BookItemID int NOT NULL,

CONSTRAINT [PK\_dbo.Attachments] PRIMARY KEY (AttachmentID))

GO

ALTER TABLE Categories ADD DEFAULT ((0)) FOR ParentID

GO

ALTER TABLE AspNetUserRoles WITH CHECK ADD CONSTRAINT FK\_AspNetUserRoles\_AspNetRoles\_RoleId FOREIGN KEY(RoleId)

REFERENCES AspNetRoles (Id)

ON DELETE CASCADE

GO

ALTER TABLE AspNetUserRoles CHECK CONSTRAINT FK\_AspNetUserRoles\_AspNetRoles\_RoleId

GO

ALTER TABLE AspNetUserRoles WITH CHECK ADD CONSTRAINT FK\_AspNetUserRoles\_AspNetUsers\_UserId FOREIGN KEY(UserId)

REFERENCES AspNetUsers (Id)

ON DELETE CASCADE

GO

ALTER TABLE AspNetUserRoles CHECK CONSTRAINT FK\_AspNetUserRoles\_AspNetUsers\_UserId

GO

ALTER TABLE AuthorGroups WITH CHECK ADD CONSTRAINT FK\_AuthorGroups\_Authors\_AuthorID FOREIGN KEY(AuthorID)

REFERENCES Authors (AuthorID)

ON DELETE CASCADE

GO

ALTER TABLE AuthorGroups CHECK CONSTRAINT FK\_AuthorGroups\_Authors\_AuthorID

GO

ALTER TABLE AuthorGroups WITH CHECK ADD CONSTRAINT FK\_AuthorGroups\_BookItems\_BookItemID FOREIGN KEY(BookItemID)

REFERENCES BookItems (BookItemID)

ON DELETE CASCADE

GO

ALTER TABLE AuthorGroups CHECK CONSTRAINT FK\_AuthorGroups\_BookItems\_BookItemID

GO

ALTER TABLE BookItems WITH CHECK ADD CONSTRAINT FK\_BookItems\_Categories\_CategoryID FOREIGN KEY(CategoryID)

REFERENCES Categories (CategoryID)

ON DELETE CASCADE

GO

ALTER TABLE BookItems CHECK CONSTRAINT FK\_BookItems\_Categories\_CategoryID

GO

ALTER TABLE Categories WITH CHECK ADD CONSTRAINT FK\_Categories\_Categories\_Parent\_CategoryID FOREIGN KEY(Parent\_CategoryID)

REFERENCES Categories (CategoryID)

GO

ALTER TABLE Categories CHECK CONSTRAINT FK\_Categories\_Categories\_Parent\_CategoryID

GO

ALTER TABLE LabelGroups WITH CHECK ADD CONSTRAINT FK\_LabelGroups\_BookItems\_BookItemID FOREIGN KEY(BookItemID)

REFERENCES BookItems (BookItemID)

ON DELETE CASCADE

GO

ALTER TABLE LabelGroups CHECK CONSTRAINT FK\_LabelGroups\_BookItems\_BookItemID

GO

ALTER TABLE LabelGroups WITH CHECK ADD CONSTRAINT FK\_LabelGroups\_Labels\_LabelID FOREIGN KEY(LabelID)

REFERENCES Labels (LabelID)

ON DELETE CASCADE

GO

ALTER TABLE LabelGroups CHECK CONSTRAINT FK\_LabelGroups\_Labels\_LabelID

GO

ALTER TABLE Orders WITH CHECK ADD CONSTRAINT FK\_Orders\_AspNetUsers\_ApplicationUserID FOREIGN KEY(ApplicationUserID)

REFERENCES AspNetUsers (Id)

GO

ALTER TABLE Orders CHECK CONSTRAINT FK\_Orders\_AspNetUsers\_ApplicationUserID

GO

ALTER TABLE Orders WITH CHECK ADD CONSTRAINT FK\_Orders\_BookItems\_BookItemID FOREIGN KEY(BookItemID)

REFERENCES BookItems (BookItemID)

ON DELETE CASCADE

GO

ALTER TABLE Orders CHECK CONSTRAINT FK\_Orders\_BookItems\_BookItemID

GO

ALTER TABLE SearchResults WITH CHECK ADD CONSTRAINT FK\_SearchResults\_AspNetUsers\_ApplicationUserID FOREIGN KEY(ApplicationUserID)

REFERENCES AspNetUsers (Id)

GO

ALTER TABLE SearchResults CHECK CONSTRAINT FK\_SearchResults\_AspNetUsers\_ApplicationUserID

GO

ALTER TABLE dbo.Attachments WITH CHECK ADD CONSTRAINT FK\_Attachments\_BookItems\_BookItemID FOREIGN KEY(BookItemID)

REFERENCES dbo.BookItems (BookItemID)

ON DELETE CASCADE

GO

ALTER TABLE dbo.Attachments CHECK CONSTRAINT FK\_Attachments\_BookItems\_BookItemID

GO