1. Zmienić **ConnectionString** w appsettings.json - bazę danych proszę nazwać pierwszą literą imienia oraz nazwiskiem - (w moim przypadku – **MPajak**). Proszę dodać metodę seedującą bazę (po 2-3 rekordy do tabeli).
2. Dodać walidację danych do **BookForm:**
    a. Wszystkie pola formularza mają być wymagane oraz posiadać czytelne komunikaty w przypadku niepoprawnej walidacji
    b. Pole **numberInStock** powinno zostać wypełnione liczbą całkowitą od 1 do 20, w przypadku nie podania wartości powinna zostać wyświetlona informacja o konieczności wypełnienia pola. Jeśli podana zostanie wartość spoza przedziału, pod polem powinna zostać wyświetlona informacja o ograniczeniu wartości od 1 do 20.
    c. Implementacja walidacji po stronie klienta
    d. Przypisać do formularza anti-forgery token
3. Zaimplementować interfejs do komunikacji z bazą dannych, zamiast bezpośrednio pracować na niej w kontrolerze, zgodnie z podejściem zaprezentowanym w czasie konwersatorium.
4. Zmodyfikować kontrolery MVC tak aby komunikowały się z serwerem za pomocą API oraz typy zwracane akcji były zgodne z konwencją RestAPI.
5. Odtworzenie widoku Index.cshtml, tak aby naciśnięcie linku w nazwie obiektu **Customer** przekierowywało do widoku **Details,** za pośrednictwem **API**. Wewnątrz widoku **Details** powinien znajdować się przycisk, przekierowujący do widoku **Edit.** Obecnie klik w nazwę **Customer** skutkuje przeniesieniem do formularza **Edit.** Aby zmienić na wyświetlanie widoku **Details** konieczne jest dodanie stosownej akcji w kontrolerze API oraz odpowiednia modyfikacja widoku **Details.** (przycissk Edit powininen być widoczny jedynie dla użytkowników posiadających odpowiednie uprawnienia)
6. Odtworzenie podejścia w widoku Index, wykorzystującego **DataTables** dla modelu Book.
7. Dodanie Autoryzacji oraz autentykacji do aplikacji - podejście prezentowane w czasie konwersatorium, bądź wykorzystanie gotowych rozwiązań dostarczonych przez framework Identity. Jako login należy zastosować adres e-mail, a hasło w aplikacji powinno być przechowywane w formie hasha.

    a. Utworzyć role: **User**, **StoreManager, Owner**
    b. Do przechowywania danych osób ze wszystkimi rolami należy skorzystać z istniejącej już tabeli **Customers**.
    c. Dostęp dla roli **User:**
        i. Widok z listą obiektów typu **Book** (Index) - Użytkownik powinien móc zobaczyć jedynie listę książek, gatunek oraz mieć możliwość wyświetlenia szczegółów.
    d. Dostęp dla roli **StoreManager**:
        i. Widok z listą obiektów typu **Book** (Index) **-** pracownik powinien móc dodawać nowe książki oraz edytować i usuwać już istniejące
        ii. Widok z listą obiektów typu **Customer** (index) - pracownik może wyświetlać jedynie użytkowników oraz ich adresy e-mail, datę urodzenia oraz informacje o subskrybcji.
    e. Dostęp dla roli **Owner**:
        i. Widok z listą obiektów typu **Book** (Index) **-** właściciel powinien móc dodawać nowe książki oraz edytować i usuwać już istniejące
        ii. Widok z listą obiektów typu **Customer** (index) - pracownik może wyświetlać wszystkich użytkowników aplikacji oraz ich adresy e-mail, datę urodzenia oraz informacje o subskrybcji.

iii. Dostęp do wszystkich funkcjonalności aplikacji. W tym usuwania i edytowania użytkowników.