

Programowanie wizualne

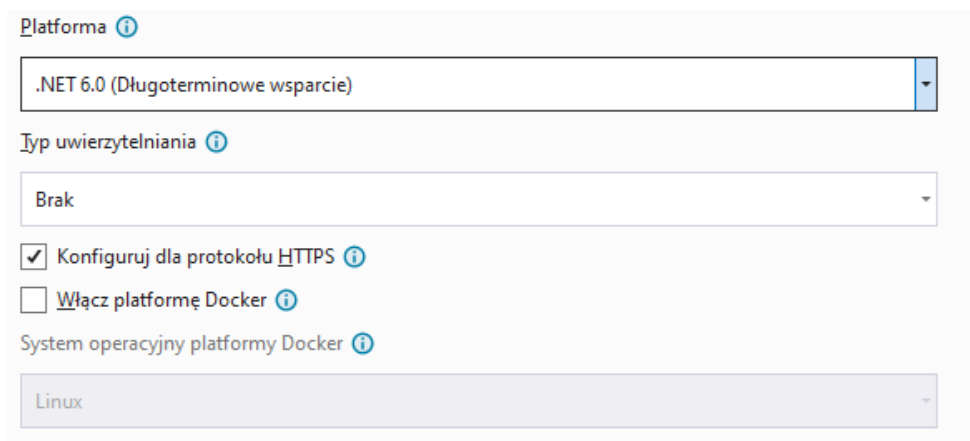
opracował: Wojciech Frohmberg

Lab 5

Zagadnienia do opanowania:

- Projekt MVC
- Entity Framework
- Razor

1. Utwórz projekt typu Aplikacja internetowa ASP.NET Core (Model-View-Controller), nadaj mu nazwę TaskShare i w opcjach wybierz następujące ustawienia:



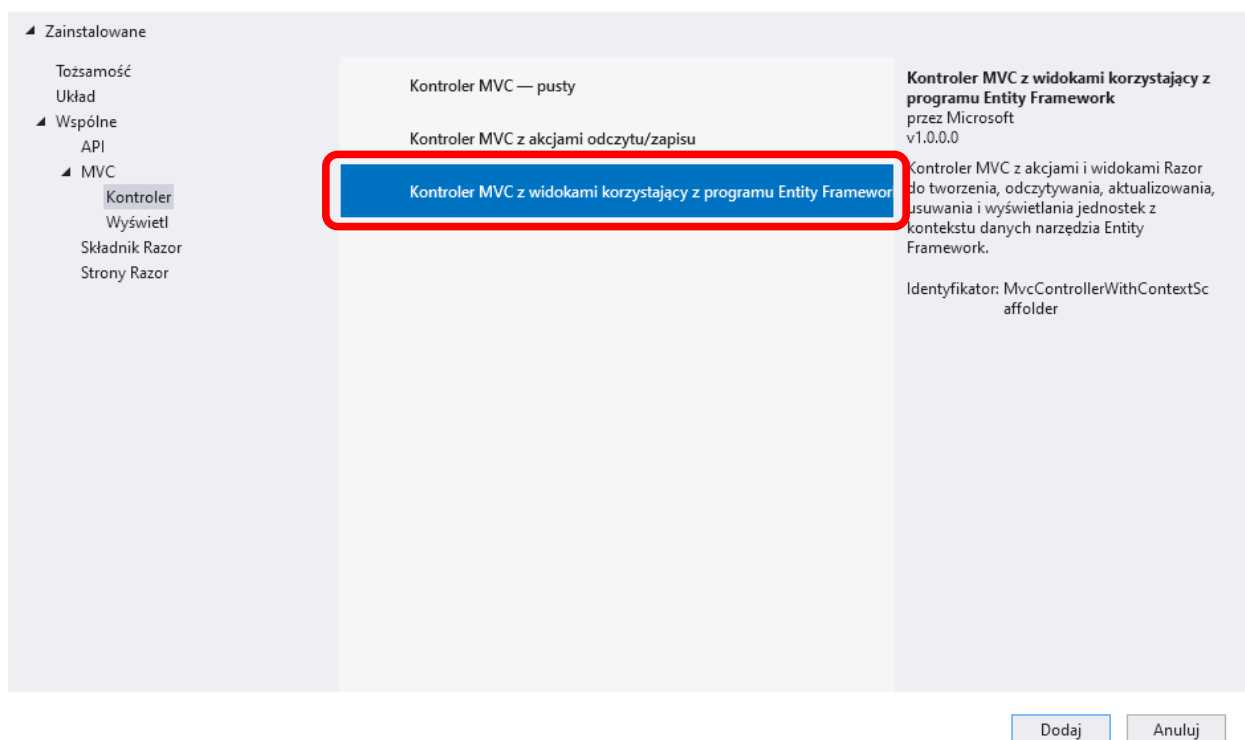
The screenshot shows the 'Platforma' (Platform) section of the ASP.NET Core project creation wizard. It includes a dropdown menu for '.NET 6.0 (Długoterminowe wsparcie)', a dropdown for 'Typ uwierzytelniania' (Authentication type) set to 'Brak' (None), a checked checkbox for 'Konfiguruj dla protokołu HTTPS' (Configure for HTTPS), an unchecked checkbox for 'Włącz platformę Docker' (Enable Docker), and a dropdown for 'System operacyjny platformy Docker' (Docker platform OS) set to 'Linux'.

2. Do folderu Models dodaj klasę Task charakteryzującą się właściwościami: Label, Description, TimeCost (liczone w szacowanych godzinach pracy nad zadaniem) oraz Priority (typu int).
3. Dodaj kolejną klasę modelu – Status charakteryzującą się właściwościami: StatusType (typ wyliczeniowy z dostępnymi rodzajami statusu) oraz Occurred.
4. Połącz relacją jeden-do-wiele modele Task oraz Status, tak by każdy status mógł referować do zmieniających się jego statusów.
5. Dodaj klasę modelu – User cechującą się właściwościami FirstName, LastName oraz Pseudonym.
6. Zadbaj by do każdego zadania mógł być podpięty jeden użytkownik wykonujący zadanie. Zastosuj relację jeden-do-wiele.
7. Dodaj klasę modelu Issue. Klasa ta będzie sprzęgała ze sobą zadania z użytkownikami. Dzięki niej określimy dla grupy użytkowników pulę zadań, którymi powinni się podzielić. Przy użyciu tej klasy łączymy zatem klasy User z klasą Task przy czym powinno to być połączenie jeden do wiele pomiędzy Issue a Task’iem oraz wiele do wiele pomiędzy Issue i użytkownikiem. Klasa Issue powinna posiadać właściwości pozwalające na określenie etykiety oraz opisu.
8. Zadbaj by wszystkie klasy modelu miały pola Id jednoznacznie identyfikujące obiekty tych klasy.
9. Przy użyciu konsolowego menedżera pakietów doinstaluj niezbędne pakiety wynikające z użycia Entity Frameworka.
10. Przy użyciu atrybutu Index (na klasach modelu) dodaj ograniczenie na unikalność elementów typu Label oraz Pseudonym poszczególnych klas. Pamiętaj, że w celu użycia atrybutu Index (a właściwie klasy atrybutu IndexAttribute w ramach kodu pliku źródłowego) należy użyć przestrzeni nazw Microsoft.EntityFrameworkCore. Można tego

dokonać poprzez automatyczne rozwiązanie zależności po wpisaniu nazwy atrybutu i wciśnięcie skrótu ctrl+space

11. Dodaj klasę kontekstu danych, zadбай o przekazanie odpowiedniej konfiguracji przy użyciu mechanizmu dependency injection. Connection string dla bazy danych umieść w pliku appsettings.json w ramach grupy wpisów „ConnectionStrings” (grupa może nie być tam zainicjalizowana stąd może być konieczność jej dodania). W klasie kontekstu pobierz z konfiguracji connection string przy użyciu metody GetConnectionString wywołanej na obiekcie konfiguracji. Na potrzeby zajęć skorzystaj z plikowej bazy danych typu Sqlite.
12. Do klasy kontekstu nie zapomnij dodać właściwości odpowiadających za gromadzenie obiektów typu Task, Status, User oraz Issue.
13. W głównym pliku aplikacji (Program.cs) zarejestruj usługę tworzącą obiekty kontekstu. Skorzystaj do tego celu z metody generycznej AddDbContext przekazując typu kontekstu w generycznym parametrze.
14. Dodaj migrację inicjalizacyjną i zainicjuj nią bazę danych. Zweryfikuj czy powstał plik odpowiadający bazie danych wynikający z ConnectionString, który umieściłaś/eś w pliku konfiguracyjnym appsettings.json.
15. Doinstaluj NuGet Microsoft.VisualStudio.Web.CodeGeneration.Design (jeśli przy instalacji będzie problem z wersją .neta skorzystaj z wersji 6.0.10).
16. Wygeneruj kod kontrolera ze standardowymi akcjami oraz widokami dla akcji – dla klasy Task, nazwij go TasksController.

Dodaj nowy element szkieletowy



17. W wygenerowanym kodzie zamień wystąpienia modelu Task na TaskShare.Models.Task.
18. Przeanalizuj wygenerowany kod i postaraj się zrozumieć implementację poszczególnych akcji, a następnie uruchom program i przejdź do akcji Index kontrolera Tasks (np. <https://localhost:<portusługi>/Tasks/Index>).
19. Przy użyciu wygenerowanych widoków spróbuj dodać obiekt typu Task do bazy. Czy akcja dodawania powiodła się? W akcji Create usuń warunek ModelState.IsValid spróbuj dodać obiekt tak jeszcze raz i zweryfikuj wyjątek wyrzucony w ramach transakcji i określ jego przyczynę. Spróbuj naprawić problem przez użycie w klasie Task właściwości z nieobligatoryjną wartością (dla użytkownika typ User?, dla właściwości Issue typ Issue?).

Pamiętaj żeby po zmianie utworzyć migrację, z odpowiednią nazwą, i zaaplikować ją do bazy. Po zaaplikowaniu zmian niezbędne może być usunięcie klasy kontrolera i widoków w folderze Views\Tasks i wygenerowanie kontrolera TasksController повторно.