1. Cel ćwiczenia: Rozwinąć projekt
2. Przebieg ćwiczenia:
3. Stwórz Klasy stanowiące model danych dla potrzebnego systemu
4. Zdefiniuj typy określające model danych a docelowo tabele w bazie danych

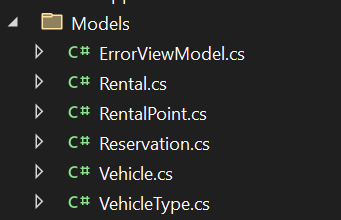
**Typ Pojazdu**

**Pojazd,**

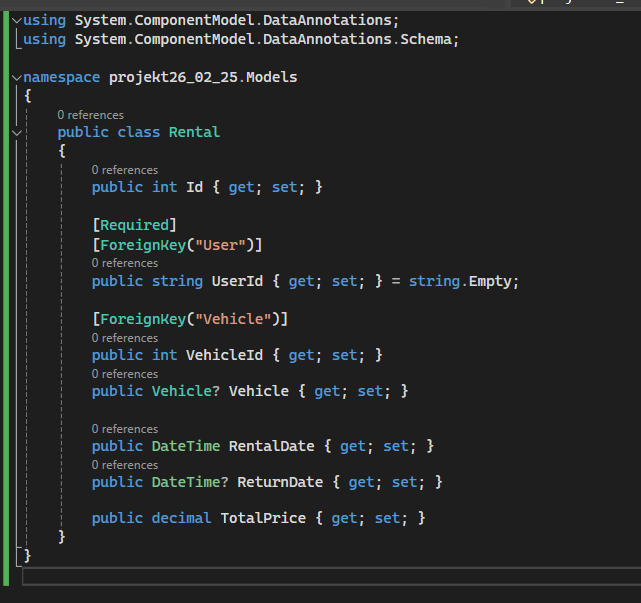
**PunktyWypozyczen**

**Rezerwacje**(pojazdów)

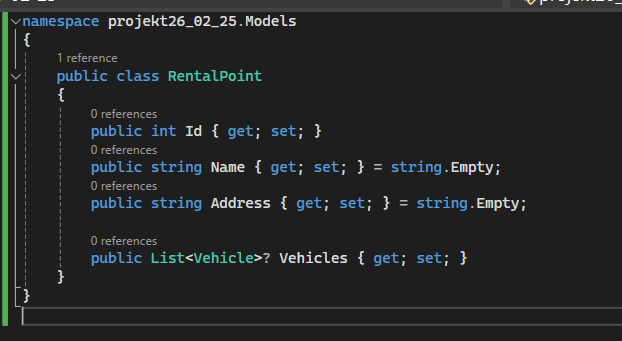
**Wypozyczenia**

7

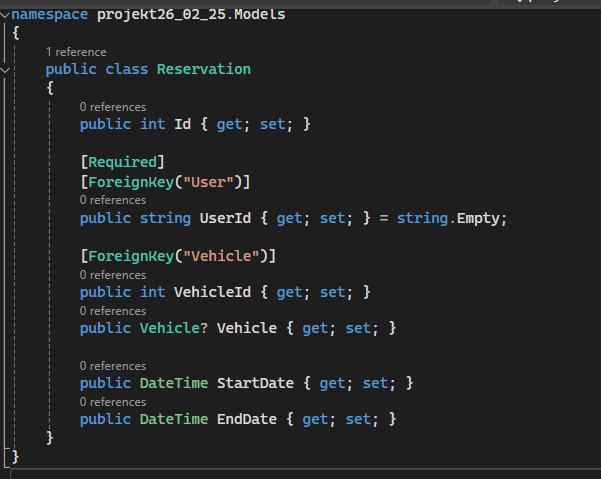
Rental.cs



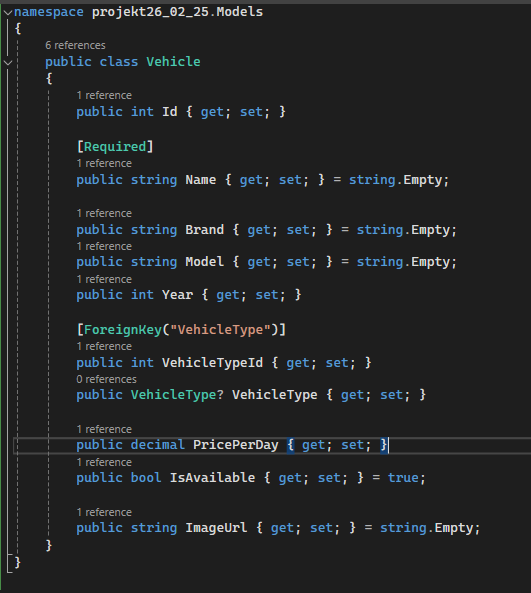
RentalPoint.cs



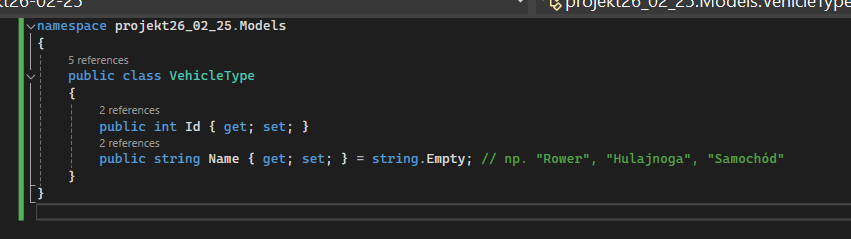
Reservation.cs



Vehicle.cs

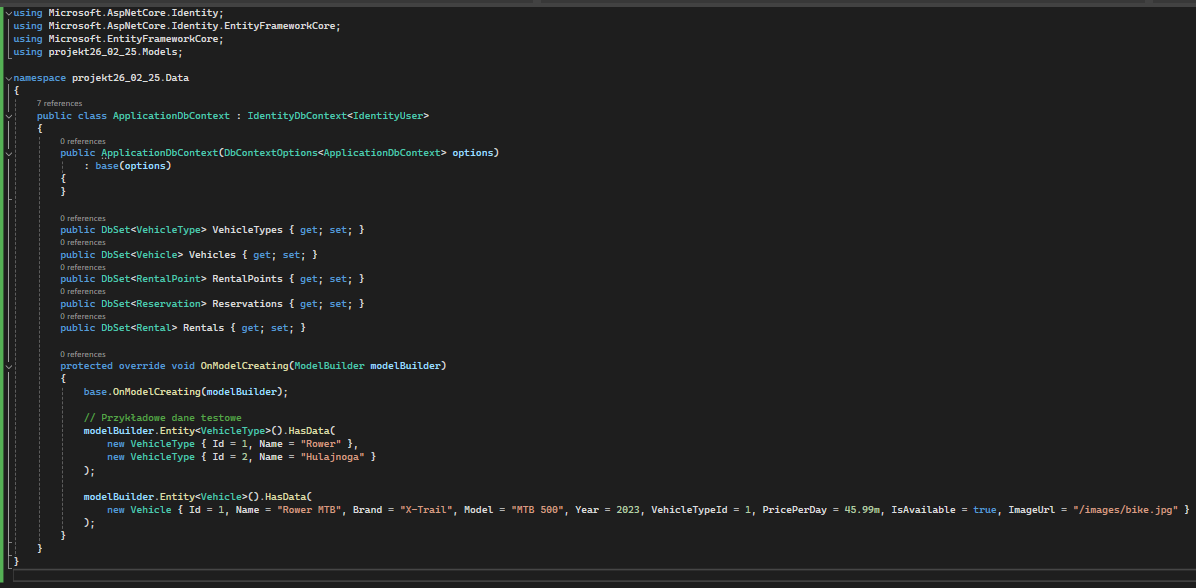


VehicleType.cs

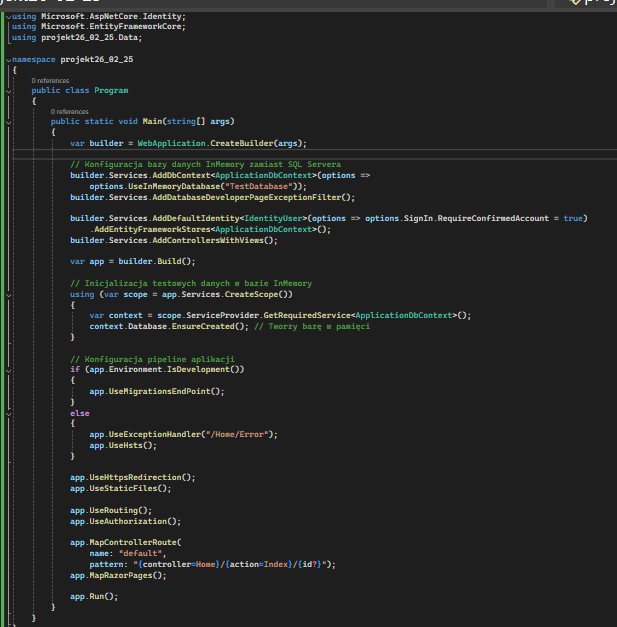


1. Jako dane użytkowników będą wykorzystywane tabele uzyskane przez odpowiednie stworzenie DBContextu
2. Zdefiniuj DbContext (można wykorzystać wygenerowaną klasę ApliactionDbContext) zawierający odpowiednie tabele. (Wywodząc DbContext od IdentityDbContex można uzyskać obsługę użytkowników).
3. Skonfiguruj aplikację by Kontekst Bazodanowy był połączony z testową baza w Pamięci. (InMemory, options.UseInMemory ....)

AppliationDbContext.cs

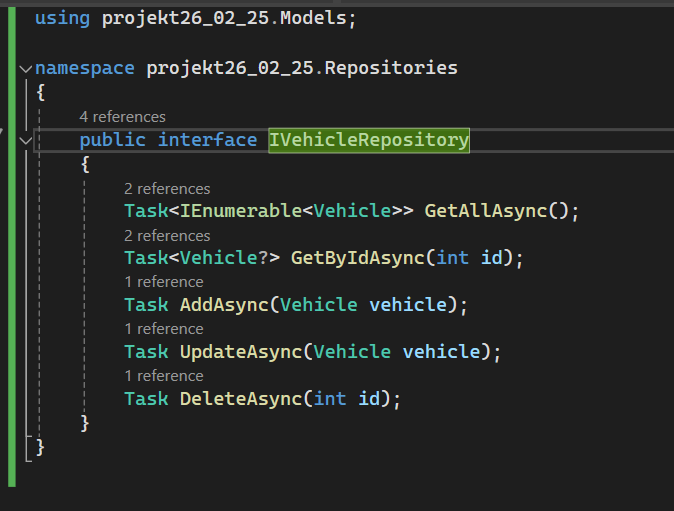


Zmodyfikowany Program.cs:

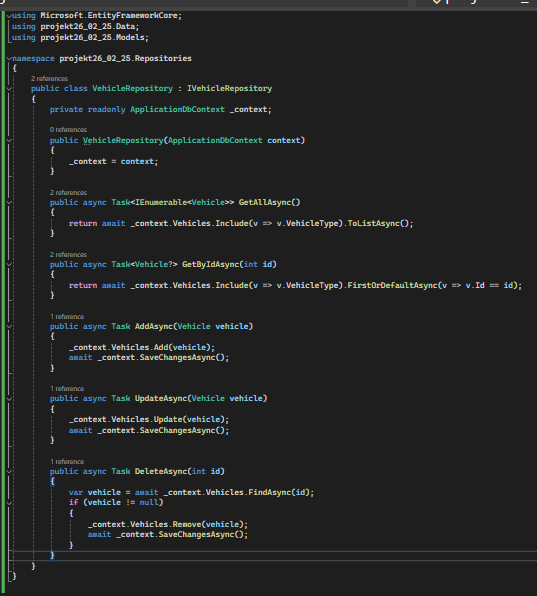


1. Zdefiniuj Serwis Implementujący wzorzec Repository służący do wykonania podstawowych operacji CRUD na wybranej tabeli

IVehicleController.cs



VehicleRepository.cs

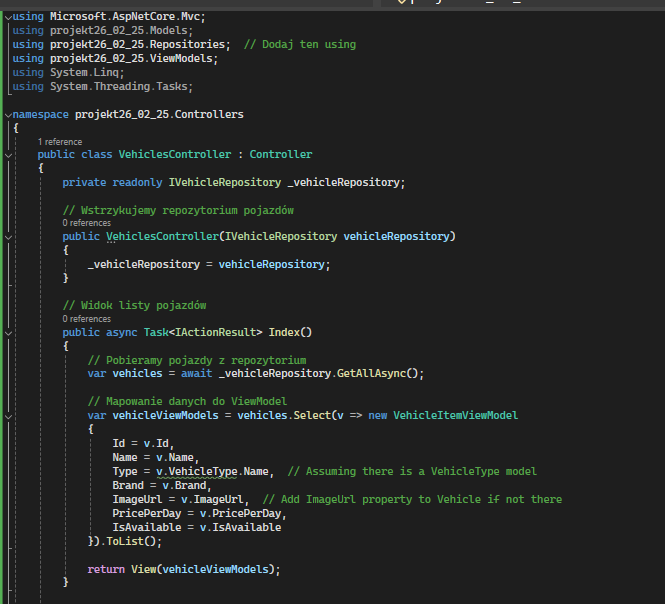


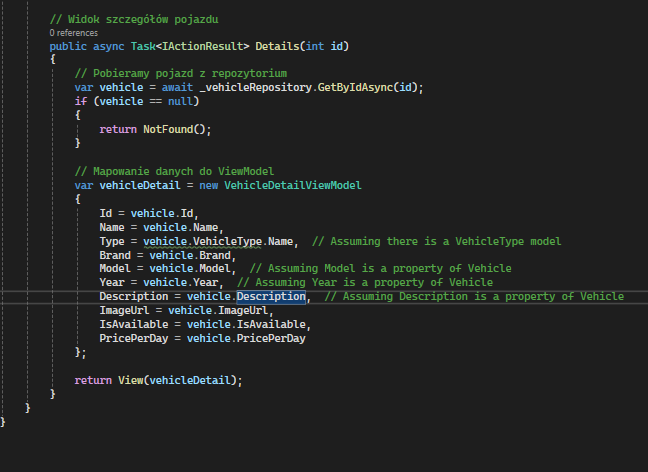
1. Skonfiguruj mechanizm wstrzykiwania dla stworzonego wzorca Repository

Program.cs



VehiclesController.cs  
(Zmiany pozwolają na wstrzykiwanie IVehicleRepository)





Vehicle.cs (dodano możliwość null dla Description)



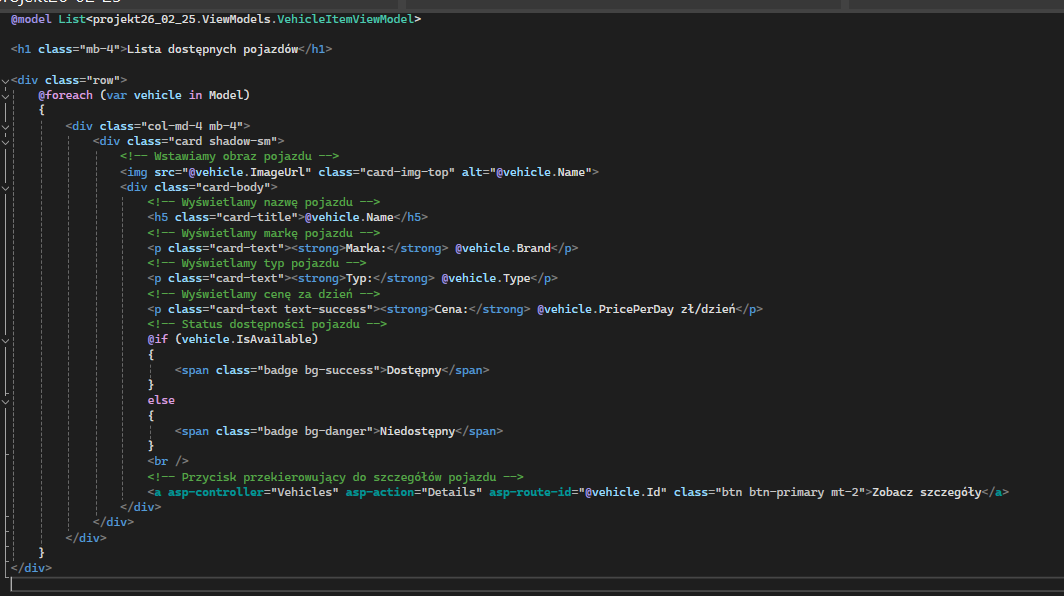
1. Stwórz  Kontroler służący do Wprowadzania Danych Pojazdów (implementujący CRUD). (Można wykorzystać scaffolding.).  
   *Kontroler Docelowo powinien wykorzystywać wstrzykiwane repozytorium.*

*Wygeneruj widoki dla akcji w kontrolerze*

VehiclesController.cs

1. using Microsoft.AspNetCore.Mvc;
2. using projekt26\_02\_25.Models;
3. using projekt26\_02\_25.Repositories;
4. using projekt26\_02\_25.ViewModels;
5. using System.Threading.Tasks;
6. using System.Linq;
7. using Microsoft.EntityFrameworkCore;
8. namespace projekt26\_02\_25.Controllers
9. {
10. public class VehiclesController : Controller
11. {
12. private readonly IVehicleRepository \_vehicleRepository;
13. public VehiclesController(IVehicleRepository vehicleRepository)
14. {
15. \_vehicleRepository = vehicleRepository;
16. }
17. // GET: Vehicles
18. public async Task<IActionResult> Index()
19. {
20. var vehicles = await \_vehicleRepository.GetAllAsync();
21. var vehicleViewModels = vehicles.Select(v => new VehicleItemViewModel
22. {
23. Id = v.Id,
24. Name = v.Name,
25. Type = v.VehicleType.Name,
26. Brand = v.Brand,
27. ImageUrl = v.ImageUrl,
28. PricePerDay = v.PricePerDay,
29. IsAvailable = v.IsAvailable
30. }).ToList();
31. return View(vehicleViewModels);
32. }
33. // GET: Vehicles/Details/5
34. public async Task<IActionResult> Details(int id)
35. {
36. var vehicle = await \_vehicleRepository.GetByIdAsync(id);
37. if (vehicle == null)
38. {
39. return NotFound();
40. }
41. var vehicleDetail = new VehicleDetailViewModel
42. {
43. Id = vehicle.Id,
44. Name = vehicle.Name,
45. Type = vehicle.VehicleType.Name,
46. Brand = vehicle.Brand,
47. Model = vehicle.Model,
48. Year = vehicle.Year,
49. Description = vehicle.Description,
50. ImageUrl = vehicle.ImageUrl,
51. IsAvailable = vehicle.IsAvailable,
52. PricePerDay = vehicle.PricePerDay
53. };
54. return View(vehicleDetail);
55. }
56. // GET: Vehicles/Create
57. public IActionResult Create()
58. {
59. return View();
60. }
61. // POST: Vehicles/Create
62. [HttpPost]
63. [ValidateAntiForgeryToken]
64. public async Task<IActionResult> Create([Bind("Id,Name,VehicleType,PricePerDay,Description,ImageUrl,IsAvailable")] Vehicle vehicle)
65. {
66. if (ModelState.IsValid)
67. {
68. await \_vehicleRepository.AddAsync(vehicle);
69. return RedirectToAction(nameof(Index));
70. }
71. return View(vehicle);
72. }
73. // GET: Vehicles/Edit/5
74. public async Task<IActionResult> Edit(int id)
75. {
76. var vehicle = await \_vehicleRepository.GetByIdAsync(id);
77. if (vehicle == null)
78. {
79. return NotFound();
80. }
81. return View(vehicle);
82. }
83. // POST: Vehicles/Edit/5
84. [HttpPost]
85. [ValidateAntiForgeryToken]
86. public async Task<IActionResult> Edit(int id, [Bind("Id,Name,VehicleType,PricePerDay,Description,ImageUrl,IsAvailable")] Vehicle vehicle)
87. {
88. if (id != vehicle.Id)
89. {
90. return NotFound();
91. }
92. if (ModelState.IsValid)
93. {
94. try
95. {
96. await \_vehicleRepository.UpdateAsync(vehicle);
97. }
98. catch (DbUpdateConcurrencyException)
99. {
100. if (!VehicleExists(vehicle.Id))
101. {
102. return NotFound();
103. }
104. else
105. {
106. throw;
107. }
108. }
109. return RedirectToAction(nameof(Index));
110. }
111. return View(vehicle);
112. }
113. // GET: Vehicles/Delete/5
114. public async Task<IActionResult> Delete(int id)
115. {
116. var vehicle = await \_vehicleRepository.GetByIdAsync(id);
117. if (vehicle == null)
118. {
119. return NotFound();
120. }
121. return View(vehicle);
122. }
123. // POST: Vehicles/Delete/5
124. [HttpPost, ActionName("Delete")]
125. [ValidateAntiForgeryToken]
126. public async Task<IActionResult> DeleteConfirmed(int id)
127. {
128. await \_vehicleRepository.DeleteAsync(id);
129. return RedirectToAction(nameof(Index));
130. }
131. private bool VehicleExists(int id)
132. {
133. return \_vehicleRepository.GetByIdAsync(id) != null;
134. }
135. }
136. }

Views/Vehicles/Index.cshtml



Views/Vehicles/Create.cshtml

