**ПРОЕКТ**

*Тема:* ***„Автокъщи“***

**Изготвили:**

- Катерина Александрова Димитрова

- Петър Павлов Бакларов

- Асен Иванов Царев

**Съдържание**

[**1.** **Тема** 3](#_Toc190601517)

[**2.** **Цели на проекта** 3](#_Toc190601518)

[**3.** **Описание на проекта** 4](#_Toc190601519)

[**4.** **Ръководство на потребителя за работа със системата** 32](#_Toc190601520)

[**5.** **Принос на участниците за изработката на проекта** 44](#_Toc190601521)

[**6.** **Линк към проекта в GitHub** 45](#_Toc190601522)

[**7.** **Заключение** 45](#_Toc190601523)

[**8.** **Използвана литература** 45](#_Toc190601524)

# **Тема**

Темата на проектната ни работа е: Интернет Програмиране: „Автокъщи“. Той трябва да предоставя удобен и ефективен начин за управление на автокъщи. Основната цел на проекта е да се създаде уеб приложение, което да улеснява потребителите в намирането на перфектната кола, а управителят на всички автокъщи (реализиран под името администратор) да може да „зарежда“ нови коли в съответната автокъща.

Нашият софтуер предоставя функционалности от тип CRUD (Create, Read, Update, Delete), описани по-долу в документацията. Приложението се базира на MVC модела и съдържа:

* Слой за данни, отговарящ за работата с базата данни.
* Слой за услуги, където се обработва логиката на приложението.
* Презентационен слой (уеб интерфейс), чрез който потребителите взаимодействат с платформата.

Приложението използва база данни за съхранение на информацията и представя връзка от тип N:M (много към много) между колите и автокъщите, като всяка кола може да бъде налична в повече от една автокъща.

# **Цели на проекта**

Целта на този проект е да покаже организиране на код в MVC (Model–View–Controller) модел, разработка на трислойно приложение и работа в GitHub. Проектът използва връзки с бази данни чрез Entity Framework, като за презентационен слой избрахме уеб базиран интерфейс (Web-based App).

Освен това, проектът демонстрира добри практики в софтуерната архитектура. Чрез разделяне на логиката в отделни слоеве (данни, услуги и уеб базиран интерфейс (web app)), се постига по-добра поддръжка и в приложението лесно и бързо може да се вградят допълнителни функционалности. Работата в GitHub позволява ефективно управление на версиите, сътрудничество между разработчиците и проследяване на направените промени. Системата осигурява и сигурност на данните, като използва механизми за автентикация и авторизация, базирани на ASP.NET Identity. Това гарантира, че различните типове потребители имат достъп само до разрешените за тях ресурси.

# **Описание на проекта**

* Създаване на среда за комуникация в Discord
* Създаване на хранилище в GitHub
* Създаване на бази от данни в MYSQL Workbench
* Реализация на MVC модел
* Създаване на връзка с базите от данни
* Писане на CRUD операции
* Изграждане на уеб базиран интерфейс
* Визуализация на основните операции
* Изготвяне на презентация и документация

**Използвахме технологии (приложения), като:**

- Microsoft Visual Studio (ASP.NET Core Web App (Model-View-Controller))

- GitHub

- Diffchecker

- Postman

- Discord

- EntityFramework

- MySQL Workbench

Нека проследим проекта от самото начало:

* Създаване на среда за комуникация в Discord
  + Първо създадохме група в приложението Discord, където поддържахме бурна комуникация относно проекта. Там се известяваше за различни грешки и там събирахме екипа за писане било то на код, на документацията или за обсъждане на дефекти в приложението.

Картина, която съдържа текст, Мултимедиен софтуер, Графичен софтуер, софтуер

Генерираното от ИИ съдържание може да е неправилно.

Картина, която съдържа текст, екранна снимка, Шрифт

Генерираното от ИИ съдържание може да е неправилно.

* + Създаване на хранилище в GitHub
  + Създадохме хранилище в GitHub за нашия проект, което позволява ефективно управление на версиите на кода и сътрудничество в екип. Чрез използването на GitHub проследяваме всички промени, осигуряваме сигурно съхранение на проекта и улесняваме бъдещото му развитие.

Картина, която съдържа текст, софтуер, Мултимедиен софтуер, екранна снимка

Генерираното от ИИ съдържание може да е неправилно.

Картина, която съдържа текст, екранна снимка, софтуер

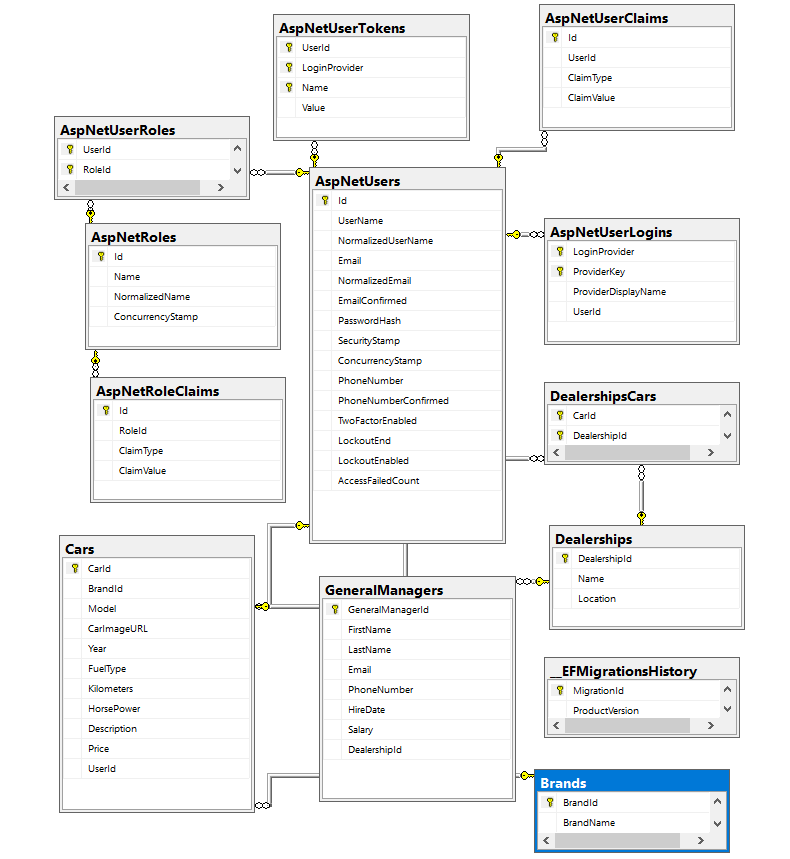
Генерираното от ИИ съдържание може да е неправилно.

* Създаване на бази от данни в MYSQL Workbench
  + Създадохме база от данни за нашия проект в **MySQL Workbench**, която съхранява информация за автокъщите, автомобилите и потребителите.

Картина, която съдържа текст, екранна снимка, Шрифт, номер

Генерираното от ИИ съдържание може да е неправилно.

* Описание на базата
  + Brands **(Марки автомобили)**
    - Съхранява информация за марките автомобили.
    - **BrandId** (уникален идентификатор) и **BrandName** (име на марката).
  + Dealerships (Дилърства)
    - Съдържа информация за различните автокъщи.
    - DealershipId (уникален идентификатор), Name (име на дилърството) и Location (местоположение).
  + Cars (Автомобили)
    - Представя списък с наличните автомобили в системата.
    - Свързана с Brands чрез BrandId.
    - Съдържа подробности като модел, година, вид гориво, километраж, мощност, цена и описание.
    - UserId е свързан с потребителя, който е от тип IdentityUser.
  + **DealershipsCars (Автомобили в дилърства)**
    - Свързва автомобили с автокъщи, като осигурява много към много релация между Cars и Dealerships.
  + **GeneralManagers (Главни мениджъри на дилърствата)**
    - Съхранява информация за управителите на автокъщите.
    - Включва лични данни като име, имейл, телефон, дата на наемане и заплата.
    - Свързана с Dealerships чрез DealershipId, като всеки мениджър ръководи конкретна автокъща.
* Основни характеристики на базата:
  + Връзка **едно към много** между **Brands** и **Cars** (една марка може да има много автомобили).
  + Връзка **много към много** между **Cars** и **Dealerships** (един автомобил може да се предлага в няколко автокъщи).
  + Връзка **едно към много** между **Dealerships** и **GeneralManagers** (всяко дилърство има един мениджър).
* БД след минаване на миграцията:



* ASP.NET Identity Таблици
  + AspNetUsers (Потребители)
    - Основната таблица за съхранение на потребителите.
    - Включва полета като UserName, Email, PasswordHash, PhoneNumber и флагове за статус (например EmailConfirmed, LockoutEnabled).
    - Свързана с останалите ASP.NET Identity таблици чрез Id (уникален идентификатор).
  + AspNetRoles (Роли)
    - Определя различни роли (напр. Администратор, Клиент, Мениджър).
    - Включва Name (име на ролята) и NormalizedName (версия на името, подходяща за търсене).
  + AspNetUserRoles (Свързване на потребители с роли)
    - Свързваща таблица между AspNetUsers и AspNetRoles (много към много релация).
    - Позволява един потребител да има множество роли.
  + AspNetUserClaims
    - Позволява добавяне на допълнителна информация за потребителите, като персонализирани права или атрибути.
  + AspNetUserLogins (Логини)
    - Поддържа вход чрез външни доставчици като Google, Facebook, Microsoft.
    - Свързва потребители с LoginProvider и ProviderKey (уникален идентификатор от външния доставчик).
  + AspNetUserTokens (Токени)
    - Съхранява допълнителни токени за удостоверяване, като двуфакторна автентикация (2FA) или запазване на сесии.
  + AspNetRoleClaims
    - Позволява асоцииране на специфични „претенции“ към роли.
* Архитектура на проекта

**Картина, която съдържа текст, екранна снимка, софтуер, Мултимедиен софтуер

Генерираното от ИИ съдържание може да е неправилно.**

**Картина, която съдържа текст, екранна снимка, софтуер, дизайн

Генерираното от ИИ съдържание може да е неправилно.Картина, която съдържа текст, екранна снимка

Генерираното от ИИ съдържание може да е неправилно.**

* Започваме описанието на проекта
  + Папка Controllers (ще опишем какво прави само CarController, тъй като другите са със сходна логика)

|  |
| --- |
| using CarDealership.Data;  using CarDealership.Data.Models;  using CarDealership.Models.Cars;  using CarDealership.Models.Dealerships;  using Humanizer.Localisation;  using Microsoft.AspNetCore.Authorization;  using Microsoft.AspNetCore.Mvc;  using Microsoft.EntityFrameworkCore;  using System.IO;  using static System.Net.WebRequestMethods;  using System.Security.Claims;  using static CarDealership.Common.AdminRole;  using static CarDealership.Data.Models.Brand;  namespace CarDealership.Controllers  {  public class CarController : Controller  {  private readonly CarDealershipDbContext \_context;  public CarController(CarDealershipDbContext carDealershipDbContext)  {  \_context = carDealershipDbContext;  }  public IActionResult Index()  {  List<Car> cars = FillBrand().OrderByDescending(x => x.CarId).ToList();  return View(cars);  }  public List<Car> FillBrand()  {  List<Car> cars = \_context.Cars.ToList();  foreach (var car in cars)  {  Brand brand = BrandById(car.BrandId);  car.Brand = brand;  }  return cars;  }  public Brand BrandById(int id)  {  List<Brand> brands = \_context.Brands.ToList();  foreach (var brand in brands)  {  if (brand.BrandId == id)  {  return brand;  }  }  return null!;  }  [HttpGet]  [Authorize(Roles = AdminRoleName)]  public async Task<IActionResult> Create()  {  List<Brand> brands = await \_context.Brands.ToListAsync();  CarCreateViewModel carCreateViewModel = new CarCreateViewModel();  carCreateViewModel.Brands = brands;  return View(carCreateViewModel);  }  [HttpPost]  public async Task<IActionResult> Create(CarCreateViewModel carCreateViewModel)  {  if (!ModelState.IsValid)  {  return View(carCreateViewModel);  }  Car car = new Car(carCreateViewModel.BrandId,  carCreateViewModel.Model,  carCreateViewModel.CarImageURL,  carCreateViewModel.Year,  carCreateViewModel.FuelType,  carCreateViewModel.Kilometers,  carCreateViewModel.HorsePower,  carCreateViewModel.Description,  carCreateViewModel.Price);  var userId = User.FindFirstValue(ClaimTypes.NameIdentifier);  car.UserId = userId;  await \_context.Cars.AddAsync(car);  await \_context.SaveChangesAsync();  return RedirectToAction("Index", "Car");  }  [HttpGet]  [Authorize(Roles = AdminRoleName)]  public async Task<IActionResult> Edit(int id)  {  Car cars = \_context.Cars.Find(id);  if (cars == null)  {  return RedirectToAction("Index", "Car");  }  CarCreateViewModel carCreateViewModel = new CarCreateViewModel()  {  Model = cars.Model,  CarImageURL = cars.CarImageURL,  Year = cars.Year,  FuelType = cars.FuelType,  Kilometers = cars.Kilometers,  HorsePower = cars.HorsePower,  Description = cars.Description,  Price = cars.Price,  BrandId = cars.BrandId,  };  ViewData["CarId"] = cars.CarId;  List<Brand> brands = await \_context.Brands.ToListAsync();  carCreateViewModel.Brands = brands;  return View(carCreateViewModel);  }  [HttpPost]  public async Task<IActionResult> Edit(int id, CarCreateViewModel carCreateViewModel)  {  Car cars = \_context.Cars.Find(id);  if (cars == null)  {  return RedirectToAction("Index", "Car");  }  if (!ModelState.IsValid)  {  ViewData["CarId"] = cars.CarId;  return View(cars);  }  cars.Model = carCreateViewModel.Model;  cars.CarImageURL = carCreateViewModel.CarImageURL;  cars.Year = carCreateViewModel.Year;  cars.FuelType = carCreateViewModel.FuelType;  cars.Kilometers = carCreateViewModel.Kilometers;  cars.HorsePower = carCreateViewModel.HorsePower;  cars.Description = carCreateViewModel.Description;  cars.Price = carCreateViewModel.Price;  cars.BrandId = carCreateViewModel.BrandId;  await \_context.SaveChangesAsync();  return RedirectToAction("Index", "Car");  }  [HttpGet]  [Authorize(Roles = AdminRoleName)]  public IActionResult Delete(int id)  {  Car car = \_context.Cars.Find(id);  if (car == null)  {  return RedirectToAction("Index", "Car");  }  \_context.Cars.Remove(car);  \_context.SaveChanges(true);  return RedirectToAction("Index", "Car");  }  }  } |

* + - Контролер, който предоставя функционалности за работа с автомобили – създаване, редактиране, изтриване и показване на списък с коли.
      * Index(): Връща списък с коли от базата данни, сортиран по ID на колата в низходящ ред. Използва метода FillBrand, за да добави марката на всяка кола към данните.
      * FillBrand(): Зарежда всички коли и за всяка кола намира съответната марка чрез BrandById(). Връща списък от коли с техните марки.
      * BrandById(): Търси и връща марката на колата по ID.
      * Create() (GET): Показва формуляр за създаване на нова кола. Изисква потребителят да е администратор (с ролята AdminRoleName). Зарежда всички марки коли и ги изпраща към изгледа.
      * Create (POST): Обработва изпратеният формуляр за създаване на нова кола. Ако формулярът е валиден, създава нова кола в базата данни и я добавя, след което пренасочва към Index за показване на списъка с коли.
      * Edit (GET): Показва формуляр за редактиране на съществуваща кола по ID. Изисква потребителят да е администратор. Зарежда съществуващата информация за колата и марки и ги изпраща към изгледа.
      * Edit (POST): Обработва изпратеният формуляр за редактиране на кола. Ако колата съществува и формулярът е валиден, актуализира данните в базата данни и пренасочва към Index.
      * Delete: Изтрива кола по ID, ако тя съществува в базата данни. След успешното изтриване пренасочва към Index.
    - Контролерът използва CarDealershipDbContext, за да осъществява връзката с базата данни.
  + От папка Data ще опишем само CarDealershipDbContext.cs и DataConstants.cs, тъй като в папка Models се съдържат данните за БД. Папка Models използва ORM Framework за работа с БД, като предоставя необходимите данни за последваща манипулация на базата.
    - CarDealershipDbContext.cs

|  |
| --- |
| using Microsoft.AspNetCore.Identity.EntityFrameworkCore;  using Microsoft.EntityFrameworkCore;  using CarDealership.Data.Models;  using Microsoft.AspNetCore.Identity;  using Humanizer.Localisation;  using System.IO;  using static CarDealership.Data.DataConstants;  using static CarDealership.Common.AdminRole;  namespace CarDealership.Data  {  public class CarDealershipDbContext : IdentityDbContext  {  public CarDealershipDbContext(DbContextOptions<CarDealershipDbContext> options)  : base(options)  {  }  public DbSet<Car> Cars { get; set; }  public DbSet<Dealership> Dealerships { get; set; }  public DbSet<DealershipsCars> DealershipsCars { get; set; }  public DbSet<GeneralManager> GeneralManagers { get; set; }  public DbSet<Brand> Brands { get; set; }  public IdentityUser AdminUser { get; set; }  private IdentityUser GuestUser { get; set; }  private List<Car> CarsList { get; set; }  private List<Dealership> DealershipsList { get; set; }  private List<GeneralManager> GeneralManagersList { get; set; }  private List<DealershipsCars> DealershipsCarsList { get; set; }  private List<Brand> BrandsList { get; set; }  // Remove seeding from here  protected override void OnModelCreating(ModelBuilder modelBuilder)  {  modelBuilder.Entity<DealershipsCars>().HasKey(dc => new  { dc.CarId, dc.DealershipId });  SeedUsers(modelBuilder);  BrandsList = SeedBrands();  modelBuilder.Entity<Brand>()  .HasData(BrandsList);  DealershipsList = SeedDealerships();  modelBuilder.Entity<Dealership>()  .HasData(DealershipsList);  GeneralManagersList = SeedGeneralManagers();  modelBuilder.Entity<GeneralManager>()  .HasData(GeneralManagersList);  CarsList = SeedCars();  modelBuilder.Entity<Car>()  .HasData(CarsList);  DealershipsCarsList = SeedDealershipsCars();  modelBuilder.Entity<DealershipsCars>()  .HasData(DealershipsCarsList);  base.OnModelCreating(modelBuilder);  }  private List<Brand> SeedBrands()  {  BrandsList = new List<Brand>  {  new Brand()  {  BrandId = 1,  BrandName = "Skoda"  },  new Brand()  {  BrandId = 2,  BrandName = "Alfa Romeo"  },  new Brand()  {  BrandId = 3,  BrandName = "BMW"  },  new Brand()  {  BrandId = 4,  BrandName = "Fiat"  },  new Brand()  {  BrandId = 5,  BrandName = "Ford"  },  new Brand()  {  BrandId = 6,  BrandName = "Honda"  },  new Brand()  {  BrandId = 7,  BrandName = "Opel"  },  new Brand()  {  BrandId = 8,  BrandName = "Audi"  },  new Brand()  {  BrandId = 9,  BrandName = "Mercedes"  },  new Brand()  {  BrandId = 10,  BrandName = "Nissan"  },  new Brand()  {  BrandId = 11,  BrandName = "Volkswagen"  },  new Brand()  {  BrandId = 12,  BrandName = "Toyota"  }  };  return BrandsList;  }  private List<GeneralManager> SeedGeneralManagers()  {  GeneralManagersList = new List<GeneralManager>  {  new GeneralManager()  {  GeneralManagerId = 1,  FirstName = "James",  LastName = "Carter",  Email="james.carter@email.com",  PhoneNumber = "+44 7512 345678",  HireDate = "08/04/2020",  Salary = 5000.00m,  DealershipId = DealershipsList[0].DealershipId  },  new GeneralManager()  {  GeneralManagerId = 2,  FirstName = "Emily",  LastName = "Johnson",  Email="emily.johnson@email.com",  PhoneNumber = "+1 415-555-1234",  HireDate = "11/11/2020",  Salary = 3000.00m,  DealershipId = DealershipsList[1].DealershipId  },  new GeneralManager()  {  GeneralManagerId = 3,  FirstName = "Lucas",  LastName = "Bennett",  Email="jrlucas.bennett@email.com",  HireDate = "28/02/2020",  PhoneNumber = "+1 202-555-4321",  Salary = 10000.00m,  DealershipId = DealershipsList[2].DealershipId  },  new GeneralManager()  {  GeneralManagerId = 4,  FirstName = "Charlotte",  LastName = "Robinson",  Email="charlotte.robinson@email.com",  HireDate = "21/06/2020",  PhoneNumber = "+44 7700 987654",  Salary = 900000m,  DealershipId = DealershipsList[3].DealershipId  },  };  return GeneralManagersList;  }  private List<Dealership> SeedDealerships()  {  DealershipsList = new List<Dealership>  {  new Dealership()  {  DealershipId = 1,  Name = "Elite Motors",  Location = "London, United Kingdom"  },  new Dealership()  {  DealershipId = 2,  Name = "Silverline Dealership",  Location = "Sydney, Australia"  },  new Dealership()  {  DealershipId = 3,  Name = "Victory Wheels",  Location = "Madrid, Spain" },  new Dealership()  {  DealershipId = 4,  Name = "Trusty Rides",  Location = "Dublin, Ireland"  },  };  return DealershipsList;  }  private List<Car> SeedCars()  {  CarsList = new List<Car>  {  new Car()  {  CarId = 1,  BrandId = BrandsList[0].BrandId,  Model = "Octavia 2",  CarImageURL="https://upload.wikimedia.org/wikipedia/commons/thumb/0/0b/Skoda\_Octavia\_II\_silver\_vr.jpg/1024px-Skoda\_Octavia\_II\_silver\_vr.jpg",  Year = 2008,  FuelType = "diesel",  Kilometers = 399138,  HorsePower = 105,  Description = "Production years: from 2004 to 2013\r\n Engine displacement: from 1197 cm3 / 73 cu-in to 1984 cm3 / 121.1 cu-in\r\n Horsepower: from 75 PS / 74 HP / 55 kW to 200 PS / 197 HP / 147 kW\r\n Length: 457.2 cm / 180 inches Width: 176.9 cm / 69.65 inches Height: 146.2 cm / 57.56 inches Wheelbase: 257.8 cm / 101.5 inches\r\n CO2 emissions: from 119 to 242 g/Km",  Price = 6500.00m,  UserId = GuestUser.Id  },  new Car()  {  CarId = 2,  BrandId = BrandsList[2].BrandId,  Model = "X5",  CarImageURL="https://stimg.cardekho.com/images/carexteriorimages/930x620/BMW/X5-2023/10452/1688992642182/front-left-side-47.jpg",  Year = 2011,  FuelType = "gasoline",  Kilometers = 206000,  HorsePower = 306,  Description = "The BMW X5 has 1 Diesel Engine and 1 Petrol Engine on offer.\r\n The Diesel engine is 2993 cc while the Petrol engine is 2998 cc.\r\n It is available with Automatic transmission.\r\nDepending upon the variant and fuel type the X5 has a mileage of 12 kmpl. The X5 is a 5 seater 6 cylinder car and has length of 4922 mm,\r\n width of 2004 mm and a wheelbase of 2975 mm.",  Price = 22000.00m,  UserId = GuestUser.Id  },  new Car()  {  CarId = 3,  BrandId = BrandsList[0].BrandId,  Model = "Octavia 3",  CarImageURL="https://www.tuning.bg/public/upload/gallery/extended/normal/sfa1.jpg",  Year = 2016,  FuelType = "diesel",  Kilometers = 174000,  HorsePower = 110,  Description = " Production years: from 2013 to 2016\r\n Engine displacement: from 1197 cm3 / 73 cu-in to 1984 cm3 / 121.1 cu-in\r\n Horsepower: from 86 PS / 85 HP / 63 kW to 220 PS / 217 HP / 162 kW\r\n Length: 465.9 cm / 183.43 inches Width: 181.4 cm / 71.42 inches Height: 146.1 cm / 57.52 inches Wheelbase: 268.6 cm / 105.75 inches\r\n Curb Weight: from 1215 kg / 2679 lbs to 1470 kg / 3241 lbs\r\n CO2 emissions: from 97 to 142 g/Km",  Price = 35000.00m,  UserId = GuestUser.Id  },  new Car()  {  CarId = 4,  BrandId = BrandsList[7].BrandId,  Model = "A6",  CarImageURL="https://stimg.cardekho.com/images/carexteriorimages/930x620/Audi/A6/10552/1708935437947/front-left-side-47.jpg",  Year = 2015,  FuelType = "gasoline",  Kilometers = 232000,  HorsePower = 272,  Description = "The Audi A6 has 1 Petrol Engine on offer. \r\nThe Petrol engine is 1984 cc . It is available with Automatic transmission.\r\nDepending upon the variant and fuel type the A6 has a mileage of 14.11 kmpl & Ground clearance of A6 is 165 mm. \r\nThe A6 is a 5 seater 4 cylinder car and has length of 4939 mm,\r\n width of 2110 mm and a wheelbase of 2500 mm.",  Price = 34500.00m,  UserId = GuestUser.Id  },  };  return CarsList;  }  private List<DealershipsCars> SeedDealershipsCars()  {  DealershipsCarsList = new List<DealershipsCars>  {  new DealershipsCars()  {  CarId = 1,  DealershipId = 1  },  new DealershipsCars()  {  CarId = 1,  DealershipId = 2  },  new DealershipsCars()  {  CarId = 2,  DealershipId = 3  },  new DealershipsCars()  {  CarId = 2,  DealershipId = 4  },  new DealershipsCars()  {  CarId = 3,  DealershipId = 1  },  new DealershipsCars()  {  CarId = 3,  DealershipId = 4  },  new DealershipsCars()  {  CarId = 4,  DealershipId = 3  },  new DealershipsCars()  {  CarId = 4,  DealershipId = 2  },  };  return DealershipsCarsList;  }  private void SeedUsers(ModelBuilder modelBuilder)  {  var hasher = new PasswordHasher<IdentityUser>();  GuestUser = new IdentityUser()  {  UserName = "GuestUser@gmail.com",  NormalizedUserName = "GuestUser@gmail.com",  };  GuestUser.PasswordHash = hasher.HashPassword(GuestUser, "123");  AdminUser = new IdentityUser()  {  Id = Guid.NewGuid().ToString(),  Email = AdminEmail,  NormalizedEmail = AdminEmail,  UserName = AdminEmail,  NormalizedUserName = AdminEmail,  };  AdminUser.PasswordHash = hasher.HashPassword(AdminUser, "admin");  modelBuilder.Entity<IdentityUser>()  .HasData(GuestUser, AdminUser);  }  }  } |

* + - * Класът CarDealershipDbContext е наследник на IdentityDbContext и предоставя контекст за взаимодействие с базата данни, която съдържа данни за автомобили, автокъщи, служители и други свързани таблици.
      * Конструкторът инициализира контекста на базата данни с предоставените опции, използвайки базовия конструктор на IdentityDbContext.
      * **DbSet свойствата** представляват различни таблици в базата данни и са основни за работа с данни. Всеки DbSet съответства на една таблица, като например Cars, Dealerships, GeneralManagers и тн.
      * Класът съдържа списъци, които съдържат предварително написани данни, използвани за наливане на информация в таблиците по време на първоначалната инициализация на базата данни.
      * **Метод OnModelCreating** основен за конфигуриране на базата данни. В него се задават ключовете за свързване на таблиците, както и се извършва инициализация на данните с помощта на методи като SeedBrands(), SeedDealerships() и други с подобна логика, които добавят предварително записи в таблиците при първоначално създаване на базата данни.
        + Seed методите генерират тестови данни. Те се използват за попълване на таблиците със зададени начални стойности при създаването на базата данни.
        + В метода SeedUsers се създават начални потребители за системата, включително администраторски и гостови профили с техните пароли и други данни.
  + DataConstants.cs

|  |
| --- |
| namespace CarDealership.Data  {  public class DataConstants  {  public class Cars  {  public const int ModelMaxLength = 20;  public const int FuelTypeMaxLength = 10;  public const int KilometersMaxLength = 10;  public const int DescriptionMaxLength = 500;  public const decimal PriceMin = 0.01m;  public const decimal PriceMax = 1000000m;  }  public class Dealerships  {  public const int NameMaxLength = 100;  public const int LocationMaxLenght = 200;  }  public class GeneralManagers  {  public const int FirstNameMaxLength = 50;  public const int LastNameMaxLength = 50;  public const int EmailMaxLength = 100;  public const int PhoneNumberMaxLength = 20;  public const decimal SalaryMin = 0.01m;  public const decimal SalaryMax = 1000000m;  }  public class Brands  {  public const int BrandMaxLength = 20;  }  }  } |

* + - Класът DataConstants съдържа различни константи, които се използват в приложението, за да се дефинират ограничения за дължината на текстови полета и стойности за минимални и максимални граници на различни атрибути в базата данни.
  + Папка Models (ще опишем какво прави само CarCreateViewModel, тъй като другите са със сходна логика)

|  |
| --- |
| using CarDealership.Data.Models;  namespace CarDealership.Models.Cars  {  public class CarCreateViewModel  {  public string Model { get; set; }  public string CarImageURL { get; set; }  public int Year { get; set; }  public string FuelType { get; set; }  public int Kilometers { get; set; }  public int HorsePower { get; set; }  public string Description { get; set; }  public decimal Price { get; set; }  public int BrandId { get; set; }  public List<Brand> Brands { get; set; } = new List<Brand>();  }  } |

* + - Класът CarCreateViewModel е модел, използван за създаване на нов автомобил в приложението. Той представлява формуляр, която потребителят попълва, за да добави нов автомобил в базата данни. Създава се като модел, защото по този начин се предотвратява директно “бъркане“ в базата данни.
  + Папка Views (ще опишем какво прави само папка Car, тъй като другите са със сходна логика)

|  |
| --- |
| //CREATE  @using CarDealership.Models.Cars;  @model CarCreateViewModel;  <div class="row">  <div class="col-md-8 mx-auto rounded border p-3">  <h2 class="text-center mb-5">New Advertisment</h2>  <**form** method="post" enctype="multipart/form-data">  <div class="row mb-3">  <label class="col-sm-4 col-form-label">Brand</label>  <div class="col-sm-8">  <**select** class="form-control" **asp-for**="@Model.BrandId">  <**option** **value**="">Choose Brand</**option**>  @foreach (var brand in Model.Brands)  {  <**option** **value**="@brand.BrandId">@brand.BrandName</**option**>  }  </**select**>  </div>  </div>  <div class="row mb-3>">  <label class="col-sm-4 col-form-label">Model</label>  <div class="col-sm-8">  <**input** class="form-control" **asp-for**="Model" />  <**span** **asp-validation-for**="Model" class="text-danger"></**span**>  </div>  </div>  <div class="row mb-3>">  <label class="col-sm-4 col-form-label">Car Image</label>  <div class="col-sm-8">  <**input** class="form-control" **asp-for**="CarImageURL" />  <**span** **asp-validation-for**="CarImageURL" class="text-danger"></**span**>  </div>  </div>  <div class="row mb-3>">  <label class="col-sm-4 col-form-label">Year</label>  <div class="col-sm-8">  <**input** class="form-control" **asp-for**="Year" />  <**span** **asp-validation-for**="Year" class="text-danger"></**span**>  </div>  </div>  <div class="row mb-3>">  <label class="col-sm-4 col-form-label">Fuel Type</label>  <div class="col-sm-8">  <**input** class="form-control" **asp-for**="FuelType" />  <**span** **asp-validation-for**="FuelType" class="text-danger"></**span**>  </div>  </div>  <div class="row mb-3>">  <label class="col-sm-4 col-form-label">Kilometers</label>  <div class="col-sm-8">  <**input** class="form-control" **asp-for**="Kilometers" />  <**span** **asp-validation-for**="Kilometers" class="text-danger"></**span**>  </div>  </div>  <div class="row mb-3>">  <label class="col-sm-4 col-form-label">Horse Power</label>  <div class="col-sm-8">  <**input** class="form-control" **asp-for**="HorsePower" />  <**span** **asp-validation-for**="HorsePower" class="text-danger"></**span**>  </div>  </div>  <div class="row mb-3>">  <label class="col-sm-4 col-form-label">Description</label>  <div class="col-sm-8">  <**input** class="form-control" **asp-for**="Description" />  <**span** **asp-validation-for**="Description" class="text-danger"></**span**>  </div>  </div>  <div class="row mb-3>">  <label class="col-sm-4 col-form-label">Price</label>  <div class="col-sm-8">  <**input** class="form-control" **asp-for**="Price" />  <**span** **asp-validation-for**="Price" class="text-danger"></**span**>  </div>  </div>    <div class="row d-flex justify-content-center gap-3">  <button type="submit" class="btn btn-primary w-50">Publish</button>  <**a** class="btn btn-outline-primary w-50" **asp-controller**="Car" **asp-action**="Index" role="button">  Cancel  </**a**>  </div>  </**form**>  </div>  </div> |
|  |
| //EDIT  @using CarDealership.Models.Cars;  @model CarCreateViewModel;  <div class="row">  <div class="col-md-8 mx-auto rounded border p-3">  <h2 class="text-center mb-5">Edit Car Advertisment</h2>  <form method="post" enctype="multipart/form-data">  <div class="row mb-3>">  <label class="col-sm-4 col-form-label">Id</label>  <div class="col-sm-8">  <input class="form-control-plaintext" readonly value="@ViewData["CarId"]" />  </div>  </div>  <div class="row mb-3">  <label class="col-sm-4 col-form-label">Brand</label>  <div class="col-sm-8">  <select class="form-control" asp-for="@Model.BrandId">  <option value="">Choose Brand</option>  @foreach (var brand in Model.Brands)  {  <option value="@brand.BrandId">@brand.BrandName</option>  }  </select>  </div>  </div>  <div class="row mb-3>">  <label class="col-sm-4 col-form-label">Model</label>  <div class="col-sm-8">  <**input** class="form-control" **asp-for**="Model" />  <**span** **asp-validation-for**="Model" class="text-danger"></**span**>  </div>  </div>  <div class="row mb-3>">  <label class="col-sm-4 col-form-label">Car Image</label>  <div class="col-sm-8">  <**input** class="form-control" **asp-for**="CarImageURL" />  <**span** **asp-validation-for**="CarImageURL" class="text-danger"></**span**>  </div>  </div>  <div class="row mb-3>">  <label class="col-sm-4 col-form-label">Year</label>  <div class="col-sm-8">  <**input** class="form-control" **asp-for**="Year" />  <**span** **asp-validation-for**="Year" class="text-danger"></**span**>  </div>  </div>  <div class="row mb-3>">  <label class="col-sm-4 col-form-label">Fuel Type</label>  <div class="col-sm-8">  <**input** class="form-control" **asp-for**="FuelType" />  <**span** **asp-validation-for**="FuelType" class="text-danger"></**span**>  </div>  </div>  <div class="row mb-3>">  <label class="col-sm-4 col-form-label">Kilometers</label>  <div class="col-sm-8">  <**input** class="form-control" **asp-for**="Kilometers" />  <**span** **asp-validation-for**="Kilometers" class="text-danger"></**span**>  </div>  </div>  <div class="row mb-3>">  <label class="col-sm-4 col-form-label">Horse Power</label>  <div class="col-sm-8">  <**input** class="form-control" **asp-for**="HorsePower" />  <**span** **asp-validation-for**="HorsePower" class="text-danger"></**span**>  </div>  </div>  <div class="row mb-3>">  <label class="col-sm-4 col-form-label">Description</label>  <div class="col-sm-8">  <**input** class="form-control" **asp-for**="Description" />  <**span** **asp-validation-for**="Description" class="text-danger"></**span**>  </div>  </div>  <div class="row mb-3>">  <label class="col-sm-4 col-form-label">Price</label>  <div class="col-sm-8">  <**input** class="form-control" **asp-for**="Price" />  <**span** **asp-validation-for**="Price" class="text-danger"></**span**>  </div>  </div>    <div class="row d-flex justify-content-center gap-3">  <button type="submit" class="btn btn-primary w-50">Publish Updated</button>  <**a** class="btn btn-outline-primary w-50" **asp-controller**="Car" **asp-action**="Index" role="button">  Cancel  </**a**>  </div>  </**form**>  </div>  </div> |
|  |
| //INDEX  @using CarDealership.Data.Models;  @using static CarDealership.Common.AdminRole;  @model List<Car>  <h2 class="text-center mb-5" style="text-align: center">List of @Model.Count Cars</h2>  <div class="row mb-5">  <div class="col">  @if (User.IsInRole(AdminRoleName))  {  <center><**a** class="btn btn-primary" **asp-area**="" **asp-controller**="Car" **asp-action**="Create" style="padding: 6px 12px; border-radius: 5px; cursor: pointer;">New Advertisment</**a**></center>  }  </div>  </div>  <!-- Контейнерът, който ще държи картите на един ред -->  <div class="car-container">  @foreach (var car in Model)  {  <div class="car-card card">  <img src="@car.CarImageURL" class="card-img-top" alt="Image of @car.Brand.BrandName @car.Model">  <div class="card-body">  <h5 class="card-title">@car.Brand.BrandName @car.Model (@car.Year)</h5>  <p class="card-text">  <strong>Fuel Type:</strong> @car.FuelType<br>  <strong>Kilometres:</strong> @car.Kilometers km<br>  <strong>Horse Power:</strong> @car.HorsePower HP<br>  <strong>Price:</strong> $@car.Price  </p>  <p class="card-text">@car.Description</p>  </div>  <div class="card-footer">  @if (User.IsInRole(AdminRoleName))  {  <**a** class="btn btn-warning btn-sm" **asp-controller**="Car" **asp-action**="Edit" **asp-route-id**="@car.CarId">Edit</**a**>  <**a** class="btn btn-danger btn-sm" **asp-controller**="Car" **asp-action**="Delete" **asp-route-id**="@car.CarId" onclick="return confirm('Are you sure?')">Delete</**a**>  }  </div>  </div>  }  </div> |

* + - //CREATE - Razor страница, използвана за създаване на нов автомобил в CarDealership проекта. Формулярът включва различни полета, които съответстват на свойствата на модела CarCreateViewModel. **Brand** е падащо меню за избор на марка на автомобила. Потребителят избира марка от списък от марки (Model.Brands). Бутон **Publish е** за публикуване на нова кола. Бутон **Cancel** е за връщане към основната страница с автомобилите.
    - //EDIT - Razor страницата е предназначена за редактиране на обяви за автомобили в CarDealership. Формулярът има същата структура като предишната за създаване на нова кола, с някои разлики. Id е поле за показване на идентификатора на автомобила, което е само за четене и не може да бъде редактирано. Стойността му се подава от ViewData["CarId"]. Бутонът Publish Updated е за обновяване на колата. Бутонът Cancel е за връщане към основната страница с автомобилите.
    - //INDEX - Razor страницата показва списък с автомобили, предоставен от модела, и предоставя функционалности за добавяне, редактиране и изтриване на коли, ако потребителят има съответната роля. Заглавието на страницата показва броя на автомобилите в списъка (@Model.Count). Ако потребителят има роля AdminRoleName, ще се появи бутон за създаване на нова обява за автомобил с надпис „New Advertisment“. Бутонът води до страницата за създаване на нова обява. Всяка обява за автомобил се показва в условно наречената “карта”. Ако потребителят има роля AdminRoleName, под всяка “карта” с информация за автомобилите има два бутона. Edit бутона служи за за редактиране на обявата, който води до страницата за редактиране на съответния автомобил. Delete бутана служи за изтриване на обявата, който показва съобщение за потвърждение преди да изтрие автомобила.

# **Ръководство на потребителя за работа със системата**

* Преди да обясним как потребителя да работи с нашето приложение, трябва да ни бъде ясно какви права на достъп притежава всеки тип потребител:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Таблица за нива на достъп | | | | |
| Вид потребителя: | CREATE | EDIT | DELETE | READ |
| Administrator |  |  |  |  |
| GuestUser |  |  |  |  |
| User |  |  |  |  |

Картина, която съдържа текст, екранна снимка, превозно средство, колело

Генерираното от ИИ съдържание може да е неправилно.

- Home е общ изглед за всеки тип потребител. Бутонът "Learn More” препраща към документацията на ASP.NET.

Картина, която съдържа текст, екранна снимка, софтуер, Компютърна икона

Генерираното от ИИ съдържание може да е неправилно.

- Register е общ изглед за всеки тип потребител.

Картина, която съдържа текст, екранна снимка, софтуер, Компютърна икона

Генерираното от ИИ съдържание може да е неправилно.

- Log In е общ изглед за всеки тип потребител.

* Изглед за потребители:

Картина, която съдържа текст, екранна снимка, Уебсайт, Уеб страница

Генерираното от ИИ съдържание може да е неправилно.

Картина, която съдържа текст, екранна снимка, софтуер, Компютърна икона

Генерираното от ИИ съдържание може да е неправилно.

Картина, която съдържа текст, екранна снимка, софтуер, Компютърна икона

Генерираното от ИИ съдържание може да е неправилно.

Картина, която съдържа текст, екранна снимка, софтуер, Компютърна икона

Генерираното от ИИ съдържание може да е неправилно.

Картина, която съдържа текст, екранна снимка, софтуер, дисплей

Генерираното от ИИ съдържание може да е неправилно.

* Изглед за администратори:

Картина, която съдържа текст, кола, екранна снимка, Сухопътно превозно средство

Генерираното от ИИ съдържание може да е неправилно.

Картина, която съдържа текст, екранна снимка, софтуер, Компютърна икона

Генерираното от ИИ съдържание може да е неправилно.

Картина, която съдържа текст, екранна снимка, софтуер, Компютърна икона

Генерираното от ИИ съдържание може да е неправилно.

Картина, която съдържа текст, екранна снимка, софтуер, Компютърна икона

Генерираното от ИИ съдържание може да е неправилно.

Картина, която съдържа текст, софтуер, Компютърна икона, Мултимедиен софтуер

Генерираното от ИИ съдържание може да е неправилно.

Картина, която съдържа текст, софтуер, Компютърна икона, Мултимедиен софтуер

Генерираното от ИИ съдържание може да е неправилно.

Картина, която съдържа текст, екранна снимка, софтуер, Компютърна икона

Генерираното от ИИ съдържание може да е неправилно.

Картина, която съдържа екранна снимка, текст, софтуер, дисплей

Генерираното от ИИ съдържание може да е неправилно.

Картина, която съдържа текст, екранна снимка, софтуер, Компютърна икона

Генерираното от ИИ съдържание може да е неправилно.

Картина, която съдържа текст, софтуер, Компютърна икона, Уеб страница

Генерираното от ИИ съдържание може да е неправилно.

Картина, която съдържа текст, софтуер, Компютърна икона, Мултимедиен софтуер

Генерираното от ИИ съдържание може да е неправилно.

Картина, която съдържа текст, софтуер, Компютърна икона, Мултимедиен софтуер

Генерираното от ИИ съдържание може да е неправилно.

Картина, която съдържа текст, екранна снимка, софтуер, Компютърна икона

Генерираното от ИИ съдържание може да е неправилно.

Картина, която съдържа текст, екранна снимка, софтуер, Компютърна икона

Генерираното от ИИ съдържание може да е неправилно.

# **Принос на участниците за изработката на проекта**

|  |  |  |  |
| --- | --- | --- | --- |
| Принос на участниците за изработката на проекта | | | |
| Участници | Катерина Димитрова | Петър Бакларов | Асен Царев |
| site.cs |  |  |  |
| Areas Folder |  |  |  |
| Common Folder |  |  |  |
| Controllers Folder |  |  |  |
| Data Folder |  |  |  |
| Data/Models Folder |  |  |  |
| Models Folder |  |  |  |
| Views Folder |  |  |  |
| appsettings.json |  |  |  |
| презентация |  |  |  |
| документацията |  |  |  |

\* По презентацията и документацията сме работили заедно. Въпреки, че Git системата отчита повече или по-малко commits на някой от членовете на екипа, ние сме работили заедно през цялото време върху почти всички компоненти. Комуникацията се водеше главно в Discord и ако някой от нас е работил самостоятелно върху някой компонент при евентуални проблеми винаги се е свързвал с другите.

# **Линк към проекта в GitHub**

* [**https://github.com/cathy-09/CarDealership**](https://github.com/cathy-09/CarDealership)

# **Заключение**

Резултатът от написаната програма е видим. С множество подобрения тя може да се развие до голямо уеб приложение с лесен и удобен интерфейс. Създаването на този софтуер обогати много знанията ни, научи ни как да работим по-добре в екип.

# **Използвана литература**

- Авторска

- Microsoft Learn