Instituto Superior de Engenharia de

Coimbra



Departamento de Engenharia Informática e de Sistemas

Engenharia Informática

Programação WEB 2019/2020

e-CarSharing

ASP.NET MVC 5 C#

Trabalho Prático – Tema B

Diogo Marques- 21240125

Introdução

No âmbito da unidade curricular de programação WEB, foi proposto a realização de um trabalho prático de forma a aplicar os conhecimentos adquiridos ao longo do primeiro semestre.

Este trabalho consiste na criação de uma aplicação dinâmica em web em ASP.NET MVC 5, recorrendo à linguagem de programação C#. Este projeto teve como principal prioridade a verificação de dados, deixando para segundo plano a aparência, não recorrendo a templates.

O tema atribuído ao aluno, tema b, consiste na existência de uma entidade e-CarSharing, cujo principal objetivo é criar a possibilidade da partilha de veículos elétricos como trotinetes, bicicletas e até mesmo veículos de quatro rodas. Estes veículos podem ser alugados numa das localizações predefinidas da e-CarSharing disponíveis no website.

1. **As regras de negócio, no contexto proposto, com base nas quais a aplicação web foi desenvolvida.**

Não é necessária uma conta para visualizar os veículos disponibilizados na plataforma, pelos restantes utilizadores.

Apenas as contas do tipo profissional e particular, podem disponibilizar os seus veículos na e-CarSharing. No entanto, é necessária a aprovação do veículo por um administrador da aplicação web.

É necessária uma conta mobilidade para alugar qualquer tipo de veículo. Ao criar um aluguer, o utilizador que deseja usufruir do veículo apenas está a fazer uma requisição, sendo necessário a sua aceitação pelo dono do veículo.

Um cliente pode verificar as datas em que um veículo está alugado, verificando assim a disponibilidade desse veículo.

Na e-CarSharing, é possível alugar um veículo numa localização pré-definida e devolver o mesmo noutra localização. Por exemplo, um cliente poderia alugar um veículo em Coimbra e devolvê-lo em Lisboa, desde que o dono permitisse o aluguer.

1. **Os tipos de utilizadores e funcionalidades implementadas.**

Administrador:

* Eliminar todas as contas existentes na plataforma
* Criar contas do tipo administrador
* Eliminar/Editar todos os veículos da plataforma
* Criar/Editar/Eliminar as localizações existentes da plataforma
* Criar/Editar/Eliminar tipos de veículos existentes na plataforma
* Criar/Editar/Eliminar marcas de veículos existentes na plataforma
* Criar/Editar/Eliminar modelos de marcas de veículos existentes na plataforma
* Eliminar todos os alugueres na plataforma

Anónimo:

* Pesquisar veículos

Particular/Profissional:

* Criar/Eliminar veículos
* Aceitar/Rejeitar alugueres dos seus veículos

Mobilidade:

* Criar Aluguer
* Utilizar veículo

1. **O modelo da Entity Framework 6.4 e os “contextos” utilizados**

Neste projeto foi utilizada a versão “Code First” na criação da base de dados, usando “IdentityDbContext<ApplicationUser>”.

* public DbSet<Brand> Brands { get; set; }
* public DbSet<Vehicle> Vehicles { get; set; }
* public DbSet<Type> Types { get; set; }
* public DbSet<Colour> Colours { get; set; }
* public DbSet<Location> Locations { get; set; }
* public DbSet<Model> Models { get; set; }
* public DbSet<VehicleState> VehicleStates { get; set; }
* public DbSet<Rent> Rents { get; set; }
* public DbSet<RentState> RentStates { get; set; }

1. **Lista de packages Nuget Utilizadas**

* Bootstrap 3.3.7
* EntityFrameWork 6.2.0
* Microsoft.AspNet.Indentity 2.2.0

1. **Lista de modelos de dados implementados**

* AccountStaticRoles
* AccountViewModel
* ApplicationDbContext
* Brand
* Colour
* IdentityModel
* Location
* ManageViewModel
* Model
* Rent
* RentState
* RentViewModel
* Type
* Vehicle
* VehicleState
* VehicleViewModel

1. **Lista dos modelos de dados pré-definidos que foram alterados.**

* IdentityModel
* AccountViewModel

1. **Lista de controladores implementados.**

Esta lista está dividida em duas áreas, área de administração e área geral.

* Administração:
  + AdminAccounts
  + Brands
  + Colours
  + Locations
  + Models
  + Types
* Geral
  + Account
  + Home
  + Rent
  + Vehicle

1. **Lista de controladores pré-definidos que foram alterados.**

* Account

1. **Lista das Views implementadas.**

* AdminAccounts
  + Create
  + Delete
  + Index
* Brands
  + Create
  + Delete
  + Index
* Colours
  + Create
  + Delete
  + Index
* Locations
  + Create
  + Delete
  + Edit
  + Index
* Models
  + Create
  + Delete
  + Index
* Types
  + Create
  + Delete
  + Index
* Account
  + Login
  + Register
  + UserProfile
* Home
  + About
  + Contact
  + Index
* Manage
  + ChangePassword
  + Index
* Rent
  + ChangeRentState
  + CheckAvailability
  + Create
  + Details
  + Index
  + MyRents
* Vehicle
  + Create
  + Delete
  + Details
  + Index
  + MyVehicles

1. **Lista das Views pré-definidas que foram alteradas.**

* Account
  + Login
  + Register
  + UserProfile
* Home
  + About
  + Contact
  + Index
* Manage
  + Index

1. **Lista dos ViewModels implementados.**

* AccountViewModel
* SearchAccountViewModel
* BrandViewModel
* BrandViewModelCreate
* ModelViewModel
* ModelViewModelCreate
* LoginViewModel
* RegisterViewModel
* ManageViewModel
* IndexViewModel
* ChangePasswordViewModel
* RentViewModel
* RentViewModelCreate
* RentViewModel ChangeState
* RentViewModelSeratch
* VehicleViewModel
* VehicleViewModelCreate
* VehicleViewModelDetails
* SearchVehicleViewModel

1. **Lista dos ViewModels pré-definidos que foram alterados.**

* AccountViewModel
* RegisterViewModel
* IndexViewModel
* ManageViewModel

1. **Lista das PartialViews implementadas.**

* \_Layout
* \_LoginPartial
* \_NavbarOptions

1. **Lista das PartialViews pré-definidas que foram alteradas.**

* Layout
* \_LoginPartial
* \_NavbarOptions

1. **Lista de templates (em cshtml) implementados**

Nenhum

1. **Lista de métodos de validação implementados**

Foram implementados dois métodos de validação neste projeto, validações de DataAnnotations e validações pelo servidor. Alguns exemplos de DataAnnotations são:

* Required
* DataType
* Range
* StringLength
* DisplayName
* MaxLength
* DisplayFormat

Quanto as verificações no lado do servidor, foram feitas precauções para, por exemplo, não existir valores inválidos como número de passageiros negativos.

1. **Lista de outras classes implementadas**

Nenhuma

1. **Lista das classes de metadados (data annotations) e das classes de configuração (FluentAPI) implementadas.**

Como mencionado no ponto 16, foram utilizados os vários tipos de DataAnnotations que seguem na lista.

* Required
* DataType
* Range
* StringLength
* DisplayName
* MaxLength
* DisplayFormat

1. **Lista dos “utilizadores-exemplo” (utilizador, senha, perfil).**

Admin, Admin123\_,"Administrador"

Diogo, Diogo123\_,"Profissional"

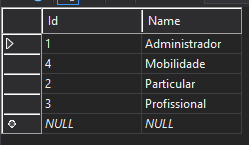
Daniela, Daniela123\_,"Particular"

Andre, Andre123\_,"Mobilidade"

1. **Lista dos dados de demonstração (existentes na base de dados).**

Neste ponto são declarados por tabela, os dados exemplo criados neste projeto.

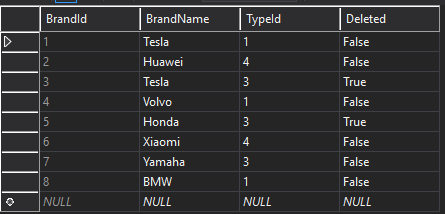
* AspNetRoles



* AspNetUsers

Utilizadores citados no ponto 19.

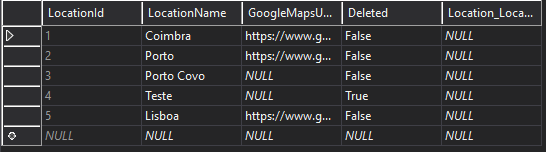
* Brands



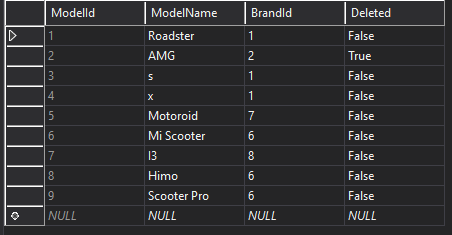
* Colours



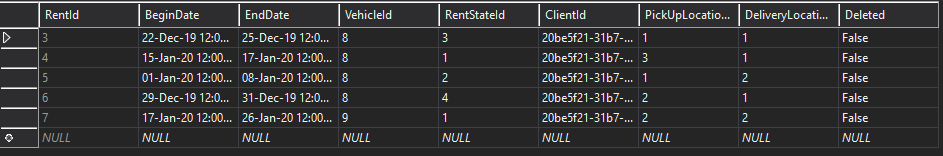
* Locations



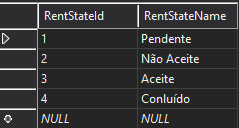
* Models



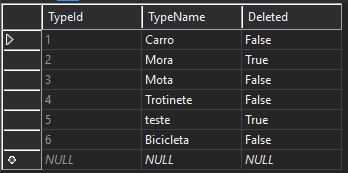
* Rents



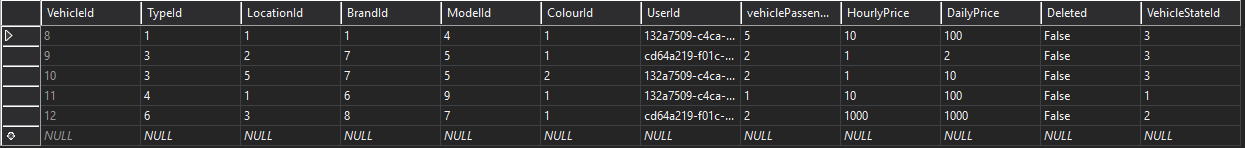
* RentStates



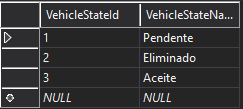
* Types



* Vehicles



* VehicleStates



Conclusão

Concluindo, neste trabalho foi desenvolvido um website dinâmico em ASP.NET MVC 5, recorrendo às suas frameworks e à linguagem de programação C#.

Este projeto foi extremamente importante para o conhecimento do aluno, como também para o desenvolver do mesmo na área da pesquisa. Permitiu melhor conhecer a linguagem de programação C#, sendo esta uma mais-valia.