Projeto de exemplo **LearningCloud**

Projeto => New Project

* Other Project Types
  + Visual Studio Solution
  + Blank Solution (Nome Projeto) **LearningCloud**
  + Seleciona a pasta de destino em **Location**:

Criar Pastas para Solution

Add - New Solution Folder

* 0 - Presentation
* 1 - Services (????)
* (2) 1 - Application
* (3) 2 - Domain
* (4) 3 - Infra
  + 3.1 - Data
  + 3.2 - CrossCutting

Adiciona o projeto MVC na camada de Apresentação (0 - Presentation)

* 0 - Presentation (Clica com Direito)
  + Add - New Project
  + Visual C# - Web - ASP.NET Web Application
  + LearningCloud.MVC - [OK] (Padrão de nomenclatura de Name space)
  + OU ====>> LearningCloud.UI.Site
  + Em *Select a template* seleciona MVC
  + Em *Add folders and core reference for:* [deixa marcado apenas] MVC
  + Change Authentication => No Authentication
  + [OK]
  + Deleta o arquivo Project\_Readme.html
  + Renomeia a pasta Models para ViewModels
* Package Manager Console
  + seleciona o *Default project* (LearningCloud.MVC)
  + **Update-Package**

Adiciona o projeto da Camada de Dominio (3 - Domain)

* 2 - Domain (Clica com Direito)
  + Add - New Project
  + Visual C# - Class Library
  + LearningCloud.Domain
  + Deleta todas as referencias em *References*
  + Deleta a Classe “Class1.cs”

Adiciona o projeto da camada de Dados em Infra - Data (3 - Infra - 3.1 - Data)

* 3 - Infra - 3.1 - Data
  + Add - New Project
  + Visual C# - Class Library
  + LearningCloud.Infra.Data
  + Deleta a Classe “Class1.cs”

Adiciona o projeto da camada de Aplicação (1 - Application)

* 1 - Application (Clica com Direito)
  + Add - New Project
  + Visual C# - Class Library
  + LearningCloud.Application
  + Deleta a Classe “Class1.cs”

Criar Pastas para organizar o Domain

* 2 - Domain
  + LearningCloud.Domain (Clica com Direito)
  + Add - New Folder
  + Entities
  + Add - New Folder
  + Interfaces
  + Add - New Folder
  + Services

Criar Pastas para organizar as Interfaces dentro de Domain / LearningCloud.Domain

* Interfaces (Clica com Direito)
  + Add - New Folder
  + Repositories
  + Add - New Folder
  + Services

Criar Pastas para organizar o Data

* 3 - Infra
* 3.1 - Data
  + LearningCloud.Infra.Data (Clica com Direito)
  + Add - New Folder
  + Context
  + Add - New Folder
  + Repositories
  + Add - New Folder
  + EntitiesConfig(EntityConfig )
  + Add - New Folder
  + EntityFramework

“Domain não conhece ninguém, mas a maioria das classes conhecem o Domain”

Adiciona a referência do Domain na camada de Dados em Infra - Data

* LearningCloud.Infra.Data
  + References (Clica com Direito)
  + Add References
  + Projects / Solution
  + Marca a opção “LearningCloud.Domain”

Agora podemos criar as classes de entidades em Domain (Video em 00:28:00)

“Classes com propriedades de estado e comportamentos. Regras de Negócio”

“Sempre public class”

Adiciona a classe na camada de Domain na pasta Entities

* 3 - Domain
* LearningCloud.Domain
* Pasta Entities (Clica com Direito)
  + Add - Class
  + NomeClasse.cs (Aula.cs)
  + definir como public

exemplo:

|  |
| --- |
| **using System;**  **//using System.Collections.Generic;**  **namespace LearningCloud.Domain.Entities**  **{**  **public class Aula**  **{**  **public int Aula\_Id { get; set; }**  **public string Aula\_Titulo { get; set; }**  **public string Aula\_TipoConteudo { get; set; }**  **public string Aula\_Descricao { get; set; }**  **public string Aula\_Prerequisitos { get; set; }**  **public string Aula\_Imagem { get; set; }**  **public string Aula\_TempoVideo { get; set; }**  **public string Aula\_Video { get; set; }**  **public string Aula\_ConteudoEscrito { get; set; }**  **public string Aula\_Status { get; set; }**  **//public int Aula\_CodigoAssinaturaNivel { get; set; }**  **//public virtual AssinaturaNivel Aula\_AssinaturaNivel { get; set; }**  **//public virtual int Aula\_CodigoInstrutor { get; set; }**  **//public virtual int Aula\_CodigoOperadorCadastro { get; set; }**  **public DateTime Aula\_DataCadastro { get; set; }**  **//public virtual int Aula\_CodigoOperadorAlteracao { get; set; }**  **public DateTime? Aula\_DataAlteracao { get; set; }**  **//public virtual ICollection<Curso> Aula\_Cursos { get; set; }**    **}**  **// podemos colocar comportamentos**  **}** |

O Campo “DateTime**?** aul\_dataalteracao” poderá aceitar valores nulos

Criar Classe de Context em Infra / Data

Adiciona a classe na camada de Dados em Infra - Data (4 - Infra - 4.1 - Data)

* 4 - Infra / 4.1 - Data
* Pasta Context (Clica com Direito)
  + Add - Class
  + LearningCloudContext.cs
  + definir como public
  + Deverá herdar de DbContext (Não está Instalado EntityFramework )

|  |
| --- |
| **public** class LearningCloudContext: **DbContext**  {  } |

Instalar EntityFramework (DbContext)

* Package Manager Console
  + seleciona o *Default project* (LearningCloud.Infra.Data)
  + Install-Package EntityFramework

adiciona o using do Entity

|  |
| --- |
| **using System.Data.Entity;** |

Criar construtor herdando de base para indicar a string de conexão com o banco

|  |
| --- |
| **public class LearningCloudContext: DbContext**  **{**  **public LearningCloudContext()**  **:base("LearningCloud") //Nome da Base de Dados**  **{**  **}**  **}** |

Criar um DbSet para criar a entidade no banco

**public DbSet<Aula> Aulas { get; set; }**

|  |
| --- |
| **public class LearningCloudContext: DbContext**  **{**  **public LearningCloudContext()**  **:base("LearningCloud")**  **{**  **}**  **public DbSet<Aula> Aulas { get; set; }**  **}** |

Adiciona o using using LearningCloud.Domain.Entities;

**using LearningCloud.Domain.Entities;**

e criar connection string no webConfig do projeto MVC (boot de entrada)

|  |
| --- |
| <configuration>  <appSettings>  <add key="webpages:Version" value="3.0.0.0"/>  <add key="webpages:Enabled" value="false"/>  <add key="ClientValidationEnabled" value="true"/>  <add key="UnobtrusiveJavaScriptEnabled" value="true"/>  </appSettings>  **<connectionStrings>**  **<add name="LearningCloudConnection" connectionString="Data Source=MAQ-209\SQLEXPRESS;Initial Catalog=LearningCloud;persist security info=True;Integrated Security=True;" providerName="System.Data.SqlClient" />**  **<!-- <add name="DefaultConnection" connectionString="Data Source=pro-dsi2; Initial Catalog=areadeteste; persist security info=True; Integrated Security=False; User Id=areadeteste; Password=areadeteste;" providerName="System.Data.SqlClient" /> -->**    **<!-- <add name="LearningCloudConnection" connectionString="Data Source=GANDRA\_XPS8500\SQLEXPRESS;Initial Catalog=LearningCloud;persist security info=True;Integrated Security=True;" providerName="System.Data.SqlClient" />-->**  **<!-- <add name="LearningCloudConnection" connectionString="Data Source=(localdb)\v11.0;Initial Catalog=LearningCloud;persist security info=True;Integrated Security=True" providerName="System.Data.SqlClient" />-->**  **</connectionStrings>**  <system.web>  <compilation debug="true" targetFramework="4.5.1"/>  <httpRuntime targetFramework="4.5.1"/>  </system.web>  …  </configuration> |

Na classe de Context implementar override no método **OnModelCreating()**

|  |
| --- |
| **public LearningCloudContext()**  **: base("LearningCloudConnection")**  **{**  **}**  **public DbSet<Aula> Aulas { get; set; }**  **protected override void OnModelCreating(DbModelBuilder modelBuilder)**  **{**  **modelBuilder.Conventions.Remove<PluralizingTableNameConvention>();**  **modelBuilder.Conventions.Remove<OneToManyCascadeDeleteConvention>();**  **modelBuilder.Conventions.Remove<ManyToManyCascadeDeleteConvention>();**  **modelBuilder.Properties()**  **.Where(p => p.Name.Contains("\_Id"))**  **.Configure(p => p.IsKey());**  **modelBuilder.Properties<string>()**  **.Configure(p => p.HasColumnType("varchar"));**  **modelBuilder.Properties<string>()**  **.Configure(p => p.HasMaxLength(100));**  **}** |

Insere o Using de Conventions

**using System.Data.Entity.ModelConfiguration.Conventions;**

Ps: No caso de realizarmos o migration agora o Banco de dados seria criado com as configurações de tamanhos de campos configurada como padrão no override de OnModelCreating

Implementar comportamentos exclusiva para a criação da tabela de Aula.

Criar em Infra Data na pasta EntitiesConfig

Adiciona nova classe na pasta EntitiesConfig

da camada de Dados em Infra - Data (4 - Infra - 4.1 - Data)

* 4 - Infra / 4.1 - Data
* Pasta EntityConfig(Clica com Direito)
  + Add - Class
  + AulaConfiguration.cs
  + definir como public

deve herdar da classe EntityTypeConfiguration<> passando a entidade Aula

|  |
| --- |
| **using System.Data.Entity.ModelConfiguration;**  **using LearningCloud.Domain.Entities;**  namespace LearningCloud.Infra.Data.EntityConfig  {  **public class AulaConfiguration: EntityTypeConfiguration<Aula>**  **{**  **}**  **}** |

cria um construtor (ctor + Tab) e adiciona as configurações para os campos

|  |
| --- |
| **namespace LearningCloud.Infra.Data.EntityConfig**  **{**  **public class AulaConfiguration : EntityTypeConfiguration<Aula>**  **{**  **public AulaConfiguration()**  **{**    **HasKey(aul => aul.Aula\_Id);**  **Property(aul => aul.Aula\_Id)**  **.HasColumnName("aul\_id")**  **.IsRequired();**  **Property(aul => aul.Aula\_Titulo)**  **.HasColumnName("aul\_titulo")**  **.IsRequired()**  **.HasMaxLength(200);**  **Property(aul => aul.Aula\_TipoConteudo)**  **.HasColumnName("aul\_tipoconteudo")**  **.IsRequired()**  **.HasColumnType("char")**  **.HasMaxLength(1);**  **Property(aul => aul.Aula\_Descricao)**  **.HasColumnName("aul\_descricao")**  **.IsOptional()**  **.HasMaxLength(8000);**  **Property(aul => aul.Aula\_Prerequisitos)**  **.HasColumnName("aul\_prerequisitos")**  **.IsOptional()**  **.HasMaxLength(8000);**  **Property(aul => aul.Aula\_Imagem)**  **.HasColumnName("aul\_imagemcapa")**  **.IsOptional()**  **.HasMaxLength(250);**  **Property(aul => aul.Aula\_Status)**  **.HasColumnName("aul\_status")**  **.IsRequired()**  **.HasColumnType("char")**  **.HasMaxLength(1);**  **Property(aul => aul.Aula\_TempoVideo)**  **.HasColumnName("aul\_videotempo")**  **.IsOptional()**  **.HasMaxLength(11);**  **Property(aul => aul.Aula\_Video)**  **.HasColumnName("aul\_videocaminho")**  **.IsOptional()**  **.HasMaxLength(250);**  **Property(aul => aul.Aula\_ConteudoEscrito)**  **.HasColumnName("aul\_conteudoescrito")**  **.IsOptional()**  **.IsMaxLength()**  **.HasColumnType("varchar(max)");**  **Property(aul => aul.Aula\_DataCadastro)**  **.HasColumnName("aul\_datacadastro")**  **.IsRequired()**  **.HasColumnType("datetime2");**  **Property(aul => aul.Aula\_DataAlteracao)**  **.HasColumnName("aul\_dataalteracao")**  **.IsOptional()**  **.HasColumnType("datetime2");**  **//Property(aul => aul.Aula\_CodigoAssinaturaNivel)**  **// .HasColumnName("aul\_fk\_assinaturanivel");**  **//HasRequired(aul => aul.Aula\_AssinaturaNivel)**  **// .WithMany()**  **// .HasForeignKey(aul => aul.Aula\_CodigoAssinaturaNivel);**  **ToTable("LearningCloud\_Aula");**  **}**  **}**  **}** |

Setar as configurações de **AulaConfiguration** no método **OnModelCreating()** na classe de Context

|  |
| --- |
| **protected override void OnModelCreating(DbModelBuilder modelBuilder)**  **{**  **modelBuilder.Conventions.Remove<PluralizingTableNameConvention>();**  **modelBuilder.Conventions.Remove<OneToManyCascadeDeleteConvention>();**  **modelBuilder.Conventions.Remove<ManyToManyCascadeDeleteConvention>();**  **modelBuilder.Properties()**  **.Where(p => p.Name == p.ReflectedType.Name + "\_id")**  **.Configure(p => p.IsKey());**  **modelBuilder.Properties<string>()**  **.Configure(p => p.HasColumnType("varchar"));**  **modelBuilder.Properties<string>()**  **.Configure(p => p.HasMaxLength(100));**  **modelBuilder.Configurations.Add(new AulaConfiguration());**    **}** |

Adiciona o Using

**using LearningCloud.Infra.Data.EntityConfig;**

Habilitar o Migrations para criar o banco de Dados

* Package Manager Console
  + seleciona o *Default project* (LearningCloud.Infra.Data)
  + Enable-Migrations

Na pasta Migrations abrir o arquivo Configuration.cs e alterar o AutomaticMigrationsEnabled para true

|  |
| --- |
| public Configuration()  {  **AutomaticMigrationsEnabled = true;**  } |

ainda no arquivo Configuration.cs limpa os using não utilizados

using System;

using System.Data.Entity;

using System.Linq;

e remove as partes em vermelho

|  |
| --- |
| internal sealed class Configuration : DbMigrationsConfiguration <**LearningCloud.Infra.Data.**Context.LearningCloudContext>  {  ...  }  protected override void Seed(**LearningCloud.Infra.Data.**Context.LearningCloudContext context)  {  ...  } |

ficando

|  |
| --- |
| **internal sealed class Configuration : DbMigrationsConfiguration <Context.LearningCloudContext>**  **{**  **...**  **}**  **protected override void Seed(Context.LearningCloudContext context)**  **{**  **…**  **}** |

Executar o Update database

* Package Manager Console
  + seleciona o Default project (LearningCloud.Infra.Data)
  + Update-Database -Verbose

Ps: Criou o Banco de dados com as configurações de tamanhos de campos da AulaConfiguration.

implementar override no método **SaveChanges()** na classe de Context

|  |
| --- |
| **public override int SaveChanges()**  **{**  **foreach (var entry in this.ChangeTracker.Entries())**  **{**  **if (entry.State != EntityState.Deleted)**  **{**  **string dataCadastro = null;**  **string dataAlteracao = null;**  **foreach (string o in entry.CurrentValues.PropertyNames)**  **{**  **var property = entry.Property(o);**  **if (property.Name.Contains("\_DataCadastro"))**  **{**  **dataCadastro = property.Name;**  **}**  **if (property.Name.Contains("\_DataAlteracao"))**  **{**  **dataAlteracao = property.Name.ToString();**  **}**  **}**  **if (entry.State == EntityState.Added)**  **{**  **if (dataCadastro != null)**  **{**  **entry.Property(dataCadastro).CurrentValue = DateTime.Now;**  **}**  **if (dataAlteracao != null)**  **{**  **entry.Property(dataAlteracao).CurrentValue = null;**  **}**  **}**  **if (entry.State == EntityState.Modified)**  **{**  **if (dataCadastro != null)**  **{**  **entry.Property(dataCadastro).IsModified = false;**  **}**  **if (dataAlteracao != null)**  **{**  **entry.Property(dataAlteracao).CurrentValue = DateTime.Now;**  **}**  **}**  **}**  **}**  **try**  **{**  **return base.SaveChanges();**  **}**  **catch (System.Data.Entity.Validation.DbEntityValidationException e)**  **{**  **foreach (var eve in e.EntityValidationErrors)**  **{**  **Console.WriteLine("Entidade do tipo \"{0}\" no estado \"{1}\" tem os seguintes erros de validação:",**  **eve.Entry.Entity.GetType().Name, eve.Entry.State);**  **foreach (var ve in eve.ValidationErrors)**  **{**  **Console.WriteLine("- Property: \"{0}\", Erro: \"{1}\"",**  **ve.PropertyName, ve.ErrorMessage);**  **}**  **}**  **throw;**  **}**  **}** |

Classe AssinaturaNivel

Adiciona mais uma classe na camada de Domain na pasta Entities

* 3 - Domain
* Pasta Entities (Clica com Direito)
  + Add - Class
  + NomeClasse.cs (AssinaturaNivel.cs)
  + definir como public

exemplo:

|  |
| --- |
| **namespace LearningCloud.Domain.Entities**  **{**  **public class AssinaturaNivel**  **{**  **public int AssinaturaNivel\_Id { get; set; }**  **public string AssinaturaNivel\_Titulo { get; set; }**  **public string AssinaturaNivel\_Descricao { get; set; }**  **public int AssinaturaNivel\_Nivel { get; set; }**  **public string AssinaturaNivel\_Status { get; set; }**  **}**  **}** |

AssinaturaNivelConfiguration

Implementar comportamentos exclusiva para a criação da tabela de AssinaturaNivel.

Criar em Infra Data na pasta EntityConfig

Adiciona nova classe na pasta EntityConfig

da camada de Dados em Infra - Data (4 - Infra - 4.1 - Data)

* 4 - Infra / 4.1 - Data
* Pasta EntityConfig(Clica com Direito)
  + Add - Class
  + AssinaturaNivelConfiguration.cs
  + definir como public

deve herdar da classe EntityTypeConfiguration<> passando a entidade AssinaturaNivel

|  |
| --- |
| **public class AssinaturaNivelConfiguration : EntityTypeConfiguration<AssinaturaNivel>**  **{**  **...**  **}** |

**using System.Data.Entity.ModelConfiguration;**

**using LearningCloud.Domain.Entities;**

cria um construtor (ctor + Tab)

**public AssinaturaNivelConfiguration()**

**{**

**}**

public AssinaturaNivelConfiguration()

{

HasKey(asn => asn.AssinaturaNivel\_Id);

Property(asn => asn.AssinaturaNivel\_Id)

.HasColumnName("asn\_id")

.IsRequired();

Property(asn => asn.AssinaturaNivel\_Titulo)

.HasColumnName("asn\_titulo")

.IsRequired()

.HasMaxLength(150);

Property(asn => asn.AssinaturaNivel\_Descricao)

.HasColumnName("asn\_descricao")

.IsRequired()

.HasMaxLength(500);

Property(asn => asn.AssinaturaNivel\_Nivel)

.HasColumnName("asn\_nivel")

.IsRequired();

Property(asn => asn.AssinaturaNivel\_Status)

.HasColumnName("asn\_status")

.IsRequired()

.HasColumnType("char")

.HasMaxLength(1);

ToTable("LearningCloud\_AssinaturaNivel");

}

Criar um DbSet para criar a entidade no banco para AssinaturaNivel na classe de Context

**public DbSet<AssinaturaNivel> AssinaturaNivel { get; set; }**

public class LearningCloudContext: DbContext

{

public LearningCloudContext()

:base("LearningCloud")

{

}

public DbSet<Aula> Aulas { get; set; }

**public DbSet<AssinaturaNivel> AssinaturaNivel { get; set; }**

}

Na classe de Context implementar no override do método **OnModelCreating()**  as configurações de AssinaturaNivelConfiguration

protected override void OnModelCreating(DbModelBuilder modelBuilder)

{

modelBuilder.Conventions.Remove<PluralizingTableNameConvention>();

modelBuilder.Conventions.Remove<OneToManyCascadeDeleteConvention>();

modelBuilder.Conventions.Remove<ManyToManyCascadeDeleteConvention>();

modelBuilder.Properties()

.Where(p => p.Name == p.ReflectedType.Name + "\_id")

.Configure(p => p.IsKey());

modelBuilder.Properties<string>()

.Configure(p => p.HasColumnType("varchar"));

modelBuilder.Properties<string>()

.Configure(p => p.HasMaxLength(100));

modelBuilder.Configurations.Add(new AulaConfiguration());

**modelBuilder.Configurations.Add(new AssinaturaNivelConfiguration());**

}

Cria FK na classe Aula na camada de Domain

**public int Aula\_CodigoAssinaturaNivel { get; set; }**

**public virtual AssinaturaNivel Aula\_AssinaturaNivel { get; set; }**

using System;

namespace LearningCloud.Domain.Entities

{

public class Aula

{

public int Aula\_Id { get; set; }

public string Aula\_Titulo { get; set; }

public string Aula\_TipoConteudo { get; set; }

public string Aula\_Descricao { get; set; }

public string Aula\_Prerequisitos { get; set; }

public string Aula\_Imagem { get; set; }

public string Aula\_TempoVideo { get; set; }

public string Aula\_Video { get; set; }

public string Aula\_ConteudoEscrito { get; set; }

public string Aula\_Status { get; set; }

**public int Aula\_CodigoAssinaturaNivel { get; set; }**

**public virtual AssinaturaNivel Aula\_AssinaturaNivel { get; set; }**

public DateTime Aula\_DataCadastro{ get; set; }

public DateTime Aula\_DataAlteracao{ get; set; }

}

}

Implementa as configurações para o Fk da tabela Aula com a tabela **AssinaturaNivel** na classe **AulaConfiguration**

using System.Data.Entity.ModelConfiguration;

using LearningCloud.Domain.Entities;

namespace LearningCloud.Infra.Data.EntityConfig

{

public class AulaConfiguration : EntityTypeConfiguration<Aula>

{

public AulaConfiguration()

{

HasKey(aul => aul.aul\_id);

Property(aul => aul.aul\_titulo)

.IsRequired()

.HasMaxLength(200);

Property(aul => aul.aul\_tipoconteudo)

.IsRequired()

.HasColumnType("char")

.HasMaxLength(1);

Property(aul => aul.aul\_descricao)

.IsRequired()

.HasMaxLength(8000);

Property(aul => aul.aul\_indroducao)

.IsRequired()

.HasMaxLength(8000);

Property(aul => aul.aul\_prerequisitos)

.HasMaxLength(8000); // .HasColumnType("varchar(max)");

Property(aul => aul.aul\_imagem)

.HasMaxLength(250);

Property(aul => aul.aul\_status)

.IsRequired()

.HasColumnType("char")

.HasMaxLength(1);

Property(aul => aul.aul\_tempovideo)

.HasMaxLength(11);

Property(aul => aul.aul\_video)

.HasMaxLength(200);

Property(aul => aul.aul\_conteudoartigo)

.HasColumnType("varchar(max)");

Property(aul => aul.aul\_palavraschave)

.HasMaxLength(8000);

Property(aul => aul.aul\_datacadastro)

.IsRequired()

.HasColumnType("datetime2");

Property(aul => aul.aul\_dataalteracao)

.IsOptional()

.HasColumnType("datetime2");

**Property(aul => aul.Aula\_CodigoAssinaturaNivel)**

**.HasColumnName("aul\_fk\_assinaturanivel");**

**HasRequired(aul => aul.Aula\_AssinaturaNivel)**

**.WithMany()**

**.HasForeignKey(aul => aul.Aula\_CodigoAssinaturaNivel);**

}

}

}

implementa o override no método Seed na classe **Configuration** da pasta **Migrations**

para criar parâmetros padrão na tabela de AssinaturaNivel

protected override void Seed(Context.LearningCloudContext context)

{

// This method will be called after migrating to the latest version.

// You can use the DbSet<T>.AddOrUpdate() helper extension method

// to avoid creating duplicate seed data. E.g.

//

// context.People.AddOrUpdate(

// p => p.FullName,

// new Person { FullName = "Andrew Peters" },

// new Person { FullName = "Brice Lambson" },

// new Person { FullName = "Rowan Miller" }

// );

//

**context.AssinaturaNivel.AddOrUpdate(**

**asn => asn.AssinaturaNivel\_Id,**

**new AssinaturaNivel()**

**{**

**AssinaturaNivel\_Id = 1,**

**AssinaturaNivel\_Titulo = "Assinatura Gratuita",**

**AssinaturaNivel\_Descricao = "Assinatura Gratuita - Todos os usuário cadastrados através do site sem assinatura definida.",**

**AssinaturaNivel\_Nivel = 10,**

**AssinaturaNivel\_Status= "A"**

**},**

**new AssinaturaNivel()**

**{**

**AssinaturaNivel\_Id = 2,**

**AssinaturaNivel\_Titulo = "Assinatura Básica",**

**AssinaturaNivel\_Descricao = "Assinatura Básica - Usuários com acessos a conteúdos privilegiados",**

**AssinaturaNivel\_Nivel = 20,**

**AssinaturaNivel\_Status = "I"**

**},**

**new AssinaturaNivel()**

**{**

**AssinaturaNivel\_Id = 3,**

**AssinaturaNivel\_Titulo = "Assinatura Premium",**

**AssinaturaNivel\_Descricao = "Assinatura Premium - Usuários com acessos a todos os conteúdos",**

**AssinaturaNivel\_Nivel = 30,**

**AssinaturaNivel\_Status = "A"**

**});**

}

**adiciona**

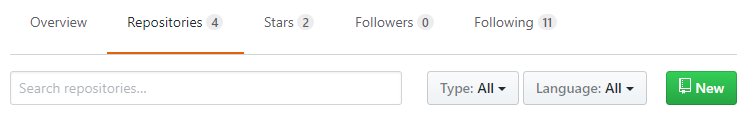
**using LearningCloud.Domain.Entities;**

Executar o Update database

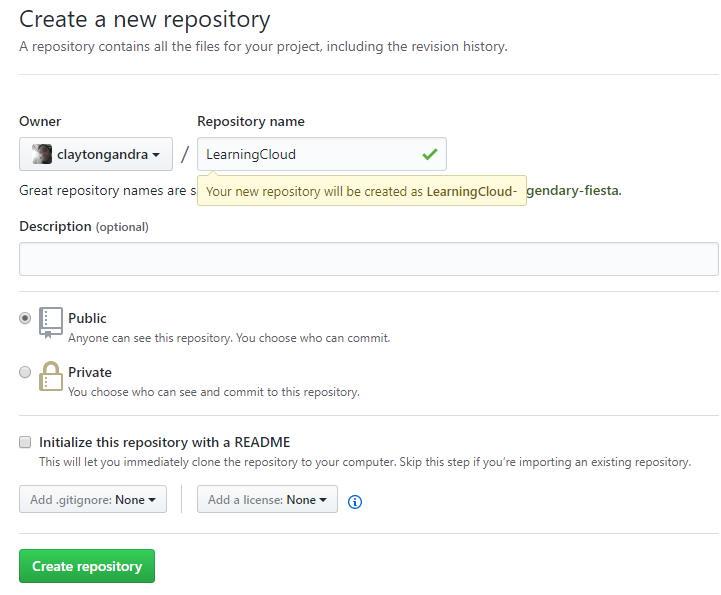
* Package Manager Console
  + seleciona o Default project (LearningCloud.Infra.Data)
  + Update-Database -Verbose **-Force**

**Vamos criar um repositório no Github**

Criar um novo repositório no Git

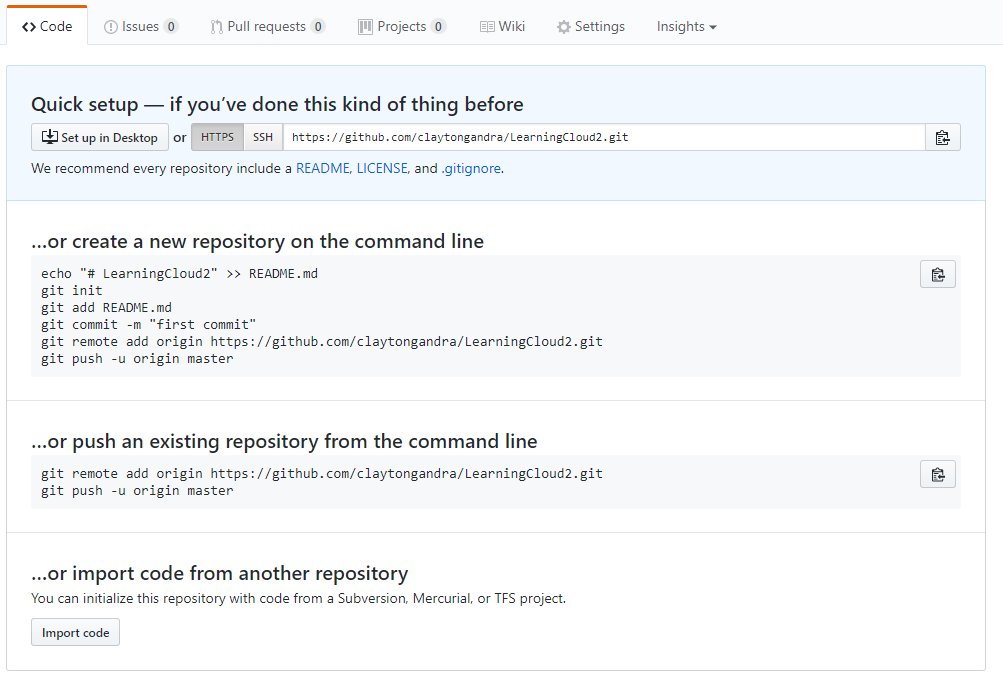


Clique em new



Guarda a URL criada para o repositório

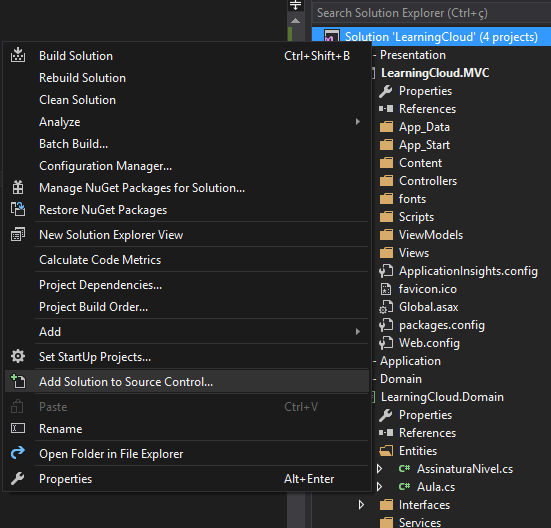
https://github.com/claytongandra/LearningCloud.git



vai no Visual Studio clica com direito em cima da solution



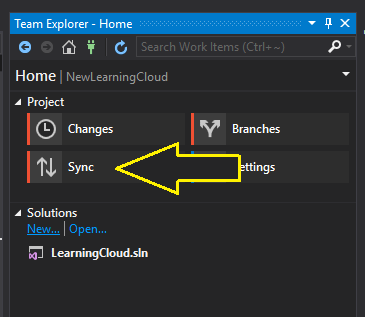
seleciona a opção “Add Solution to Source Control…

l

caso der commit será criado no Git local apenas, então vamos na aba Team Explorer / Home

Clica em Sync

Adiconar a URL do git



**Criar Interfaces de repository na camada de Domain**

**Criar interface de repositório genérico**

Adiciona a interface generica de repositorio na camada de Domain na pasta Interfaces / Repositories

* 3 - Domain
* LearningCloud.Domain
* Pasta Interfaces
* Pasta Repositories (Clica com Direito)
  + Add - New Item
  + Interface
  + **IRepositoryBase**.cs (INomeInterface.cs)
  + definir como public

**Recebendo uma entidade genérica de TEntity, tratando quando for uma class (where TEntity: class)**

**<TEntity> where TEntity: class**

**using System.Collections.Generic;**

**namespace LearningCloud.Domain.Interfaces.Repositories**

**{**

**public interface IRepositoryBase<TEntity> where TEntity: class**

**{**

**void Add(TEntity obj);**

**TEntity GetById(int id);**

**IEnumerable<TEntity> GetAll();**

**void Update(TEntity obj);**

**void Remove(TEntity obj);**

**void Dispose();**

**}**

**}**

Adiciona a interface de repositorio para **Aula** na camada de Domain na pasta Interfaces / Repositories

* 3 - Domain
* Pasta Repositories (Clica com Direito)
  + Add - New Item
  + Interface
  + **IAulaRepository**.cs (INomeInterface.cs)
  + definir como public
* ou
* Pasta Repositories (Clica com Direito)
* Add - New from Template
* Interface

Herdando a **IRepositoryBase** do tipo **<Aula>**

using LearningCloud.Domain.Entities;

namespace LearningCloud.Domain.Interfaces.Repositories

{

public interface IAulaRepository: IRepositoryBase<Aula>

{

**IEnumerable<Aula> GetByStatus(string status);**

}

}

adiciona o using de LearningCloud.Domain.**Entities**;

**using LearningCloud.Domain.Entities;**

**Criar repositórios concretos na camada de Infra / Data**

Adiciona a classe de repositorio para RepositoryBase na camada de Infra.Data na pasta Repositories

* 4 - Infra / 4.1 - Data
* Pasta Repositories (Clica com Direito)
  + Add - Class
  + RepositoryBase.cs
  + definir como public

Recebendo TEntity, com : IDisposable para poder destruir essa instância e implementa a interface de IRepositoryBase<> Recebendo TEntity, quando TEntiti for uma classe (where TEntity : class)

<TEntity> : IDisposable, IRepositoryBase<TEntity> where TEntity : class

incluir

**using System;**

**using LearningCloud.Domain.Interfaces.Repositories;**

using System;

using LearningCloud.Domain.Interfaces.Repositories;

namespace LearningCloud.Infra.Data.Repositories

{

public class RepositoryBase<TEntity> : IDisposable, IRepositoryBase<TEntity> where TEntity : class

{

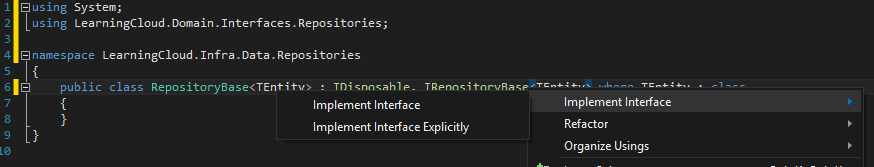
}

}

Implementa interface de IRepositoryBase

Clica com o direito em IRepositoryBase que está sendo implementada na classe RepositoryBase

Implement Interface / Implement Interface



using System;

using LearningCloud.Domain.Interfaces.Repositories;

namespace LearningCloud.Infra.Data.Repositories

{

public class RepositoryBase<TEntity> : IDisposable, IRepositoryBase<TEntity> where TEntity : class

{

public void Add(TEntity obj)

{

throw new NotImplementedException();

}

public TEntity GetById(int id)

{

throw new NotImplementedException();

}

public System.Collections.Generic.IEnumerable<TEntity> GetAll()

{

throw new NotImplementedException();

}

public void Update(TEntity obj)

{

throw new NotImplementedException();

}

public void Remove(TEntity obj)

{

throw new NotImplementedException();

}

public void Dispose()

{

throw new NotImplementedException();

}

}

}

Retira o System.Collections.Generic. do método GetAll()

public **System.Collections.Generic.**IEnumerable<TEntity> GetAll()

{

throw new NotImplementedException();

}

adicionar o using do System.Collections.Generic

**using System.Collections.Generic;**

Vamos implementar uma instância do contexto

Cria um construtor na classe RepositoryBase ctor +(Tab)

e um campo privado do tipo LearningCloudContext

public class RepositoryBase<TEntity> : IDisposable, IRepositoryBase<TEntity> where TEntity : class

{

**protected LearningCloudContext ContextDB { get; private set; }**

**public RepositoryBase()**

**{**

**ContextDB = new LearningCloudContext();**

**}**

public void Add(TEntity obj)

{

throw new NotImplementedException();

}

Incluir o using do Context;

using System;

using LearningCloud.Domain.Interfaces.Repositories;

**using LearningCloud.Infra.Data.Context;**

e vamos implementar os métodos que a interface exige

void Add(TEntity obj);

TEntity GetById(int id);

IEnumerable<TEntity> GetAll();

void Update(TEntity obj);

void Remove(TEntity obj);

void RemoveById(int id);

void Dispose();

public void Add(TEntity obj)

{

ContextDB.Set<TEntity>().Add(obj);

}

public TEntity GetById(int id)

{

return ContextDB.Set<TEntity>().Find(id);

}

public IEnumerable<TEntity> GetAll()

{

return \_contextDB.Set<TEntity>().ToList(); // verificar o AsNoTracking().ToList();

}

public void Update(TEntity obj)

{

ContextDB.Entry(obj).State = EntityState.Modified;

}

public void Remove(TEntity obj)

{

ContextDB.Set<TEntity>().Remove(obj);

}

public void RemoveById(int id)

{

var obj = GetById(id);

Remove(obj);

}

public void Dispose()

{

ContextDB.Dispose();

}

adiciona os usings:

using System.Linq; para o metodo ToList() do GetAll();

using System.Data.Entity; para o **EntityState**.Modified; do Update(TEntity obj)

Foram implementados os metodos de Repository base sem SaveChanges

Implementar Repository de Aula

**Criar repositórios concretos na camada de Infra / Data**

Adiciona a classe de repositorio para AulaRepository na camada de Infra.Data na pasta Repositories

* 4 - Infra / 4.1 - Data
* Pasta Repositories (Clica com Direito)
  + Add - Class
  + AulaRepository.cs
  + definir como public

Herdabdo a classe RepositoryBase<> recebendo <Aula> e implementa a interface de IAulaRepository

incluir

**using LearningCloud.Domain.Entities;**

**using LearningCloud.Domain.Interfaces.Repositories;**

Implementa Interfaces da IAulaRepository na AulaRepository

Clica com o direito em IAulaRepository que está sendo implementada na classe AulaRepository

Implement Interface / Implement Interface

using System.Linq;

using System.Collections.Generic;

using NewLearningCloud.Domain.Entities;

using NewLearningCloud.Domain.Interfaces.Repositories;

namespace NewLearningCloud.Infra.Data.Repositories

{

public class AulaRepository : RepositoryBase<Aula>, IAulaRepository

{

public IEnumerable<Aula> GetByStatus(string status)

{

throw new NotImplementedException();

}

}

}

Vamos implementar o metodo GetByStatus da classe AulaRepository:

public class AulaRepository : RepositoryBase<Aula>, IAulaRepository

{

public IEnumerable<Aula> GetByStatus(string status)

{

**string[] arrayStatus = status.Split(',');**

**return ContextDB.Set<Aula>().Where(a => arrayStatus.Contains(a.Aula\_Status)).ToList();**

}

}

}

incluir

**using System.Linq;**

**Criar estruturas para o Unit Of Work**

Criar Interface para o Unit of Work na camada de Domain

Adiciona a interface de repositorio para **UnitOfWork** na camada de Domain na pasta Interfaces / Repositories

* 3 - Domain
* LearningCloud.Domain
* Pasta Repositories (Clica com Direito)
  + Add - New Item
  + Interface
  + **IUnitOfWorkRepository**.cs (INomeInterface.cs)
  + definir como public

namespace LearningCloud.Domain.Interfaces.Repositories

{

public interface IUnitOfWorkRepository

{

}

}

implementa dois metodos

void BeginTransactionUoW();

void CommitUoW();

ficando:

namespace LearningCloud.Domain.Interfaces.Repositories

{

public interface IUnitOfWorkRepository

{

void BeginTransactionUoW();

void CommitUoW();

}

}

Adiciona a classe concreta de repositorio para UnitOfWorkRepository na camada de Infra.Data na pasta EntityFramework

* 4 - Infra / 4.1 - Data
* Pasta EntityFramework(Clica com Direito)
  + Add - Class
  + UnitOfWorkRepository.cs
  + definir como public

Herdando a interface IUnitOfWorkRepository

incluir

**using LearningCloud.Domain.Interfaces.Repositories;**

Implementa Interfaces da IUnitOfWorkRepository na UnitOfWorkRepository

Clica com o direito em IUnitOfWorkRepository que está sendo implementada na classe UnitOfWorkRepository

Implement Interface / Implement Interface

ficando :

using LearningCloud.Domain.Interfaces.Repositories;

namespace LearningCloud.Infra.Data.EntityFramework

{

public class UnitOfWorkRepository : IUnitOfWorkRepository

{

public void BeginTransactionUoW()

{

throw new System.NotImplementedException();

}

public void CommitUoW()

{

throw new System.NotImplementedException();

}

}

}

Implementar os métodos de BeginTransactionUoW() e CommitUoW()

Com o EntityFramework a transação é aberta automaticamente.

Vamos ter que obter o Context e utilizar esse contexto para fazer o Commit usando o padrão Context Per Request (Um contexto por requisição)

poderíamos utilizar o [StructureMap](https://www.google.com.br/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0ahUKEwiW9uKUorrPAhVIFpAKHX_xCBMQFggcMAA&url=http%3A%2F%2Fstructuremap.github.io%2Fthe-container%2Fnested-containers%2F&usg=AFQjCNHdQs3PAwUIOT9nv9dpZ31fpkT_cw&sig2=Y4JVu1VyZigX-_tLkn_45Q&bvm=bv.134495766,d.Y2I&cad=rja) ou fazer na mão com o HttpContext

Famos fazer com o HttpContext

Adiciona o projeto de IoC em CrossCutting em Infra - CrossCutting(3 - Infra - 3.2 - CrossCutting)

* 3 - Infra - 3.1 - CrossCutting
  + Add - New Project
  + Visual C# - Class Library
  + LearningCloud.Infra.CrossCutting.IoC
  + Deleta a Classe “Class1.cs”

e vamos instalar o SimpleInjector (Install-Package SimpleInjector)

* Package Manager Console
  + seleciona o *Default project* (LearningCloud.Infra.CrossCutting.IoC)
  + Install-Package SimpleInjector

Instalar também o CommonServiceLocator

* Package Manager Console
  + seleciona o *Default project* (LearningCloud.Infra.CrossCutting.IoC)
  + Install-Package CommonServiceLocator

Criar classe de configuração para o SimpleInjector

Adiciona a classe configuração do SimpleInjector na camada de Infra.CrossCutting.IoC chamada BootStrapper

* 3 - Infra / 3.1 - CrossCutting
* Projeto LearningCloud.Infra.CrossCutting.IoC(Clica com Direito)
  + Add - Class
  + BootStrapper.cs
  + definir como public

cria método estático para registar as implementações entre classes concretas e interfaces

public static void RegisterServices(Container container)

{

}

adiciona o using para a referencia do SimpleInjector;

**using SimpleInjector;**

vamos utilizar uma extensão do SimpleInjector para registar a integração com o ASP.NET

SimpleInjector.Integration.Web

* Package Manager Console
  + seleciona o *Default project* (LearningCloud.Infra.CrossCutting.IoC)
  + Install-Package SimpleInjector.Integration.Web

Adicionaremos também as referências dos projetos de Domain e Infra.Data no projeto de IoC

Adiciona a referência do Domain na camada de CrossCutting em Infra - CrossCutting.IoC

* LearningCloud.Infra.CrossCutting.IoC
  + References (Clica com Direito)
  + Add References
  + Marca a opção “LearningCloud.Domain”

Adiciona a referência da Infra.Data na camada de CrossCutting em Infra - CrossCutting.IoC

* LearningCloud.Infra.CrossCutting.IoC
  + References (Clica com Direito)
  + Add References
  + Marca a opção “LearningCloud.Infra.Data”

agora podemos registar as implementações das interfaces para as classes concretas

public class BootStrapper

{

public static void RegisterServices(Container container)

{

**container.Register<IAulaRepository, AulaRepository>(Lifestyle.Scoped**

**);**

**container.Register<IUnitOfWorkRepository, UnitOfWorkRepository>();**

}

}

Lifestyle.Scoped porque implementamos o *IDisposable na* **AulaRepository**

adiciona os usings de

**using LearningCloud.Domain.Interfaces.Repositories;**

**using LearningCloud.Infra.Data.EntityFramework;**

**using LearningCloud.Infra.Data.Repositories;**

E vamos configurar o ServiceLocator para adaptar uma interface para outra ServiceLocator.SetLocatorProvider();

vamos instalar uma implementação pronta para o SimpleInjector no servicelocator

CommonServiceLocator.SimpleInjectorAdapter

* Package Manager Console
  + seleciona o *Default project* (LearningCloud.Infra.CrossCutting.IoC)
  + Install-Package CommonServiceLocator.SimpleInjectorAdapter -Version 2.8.2

confirmar versão https://www.nuget.org/stats/packages/CommonServiceLocator.SimpleInjectorAdapter?groupby=Version

implementa o SimpleInjectorAdapter

namespace LearningCloud.Infra.CrossCutting.IoC

{

public class BootStrapper

{

public static void RegisterServices(Container container)

{

container.RegisterPerWebRequest<IAulaRepository, AulaRepository>();

container.RegisterPerWebRequest<IUnitOfWorkRepository, UnitOfWorkRepository>();

**var adapter = new SimpleInjectorServiceLocatorAdapter(container);**

**ServiceLocator.SetLocatorProvider(() => adapter);**

}

}

}

adiciona os usings

**using Microsoft.Practices.ServiceLocation;**

**using CommonServiceLocator.SimpleInjectorAdapter;**

agora teremos que criar o contextManager para ter a instância do objeto do contexto

Adiciona a classe ContextManager na camada de Infra.Data na pasta EntityFramework

* 3 - Infra / 3.1 - Data
* Pasta EntityFramework(Clica com Direito)
  + Add - Class
  + ContextManager.cs
  + definir como public

essa classe vai devolver o contexto e gerenciar esse context

vai ter que fazer referência ao System.Web na camada de infra

using System.Web;

using System.Data.Entity;

using LearningCloud.Infra.Data.Context;

namespace LearningCloud.Infra.Data.EntityFramework

{

public class ContextManager<TContext> where TContext : DbContext, new()

{

private string ContextKey = "ContextManager.Context";

public ContextManager()

{

ContextKey = "ContextKey." + typeof(TContext).Name;

}

public DbContext GetContext

{

get

{

if (HttpContext.Current.Items[ContextKey] == null)

{

HttpContext.Current.Items[ContextKey] = new TContext();

}

//return (LearningCloudContext)HttpContext.Current.Items[ContextKey];

return HttpContext.Current.Items[ContextKey] as DbContext;

}

}

}

}

agora podemos implementar o BeginTransactionUoW e o ComitUoW na classe UnitOfWorkRepository

Instalar o CommonServiceLocator na camada de Infra.data

* Package Manager Console
  + seleciona o *Default project* (LearningCloud.Infra.Data)
  + Install-Package CommonServiceLocator

e implementa o BeginTransactionUoW

**using LearningCloud.Infra.Data.Context;**

public class UnitOfWorkRepository : IUnitOfWorkRepository

{

**private DbContext \_context;**

public void BeginTransactionUoW()

{

**var contextManager = ServiceLocator.Current.GetInstance<ContextManager<LearningCloudContext>>();**

**\_context = contextManager.GetContext;**

}

public void ComitUoW()

{

}

}

adiciona os usings

**using System.Data.Entity;**

**using Microsoft.Practices.ServiceLocation;**

e implementar o ComitUoW

public class UnitOfWorkRepository : IUnitOfWorkRepository

{

private DbContext \_context;

public void BeginTransactionUoW()

{

var contextManager = ServiceLocator.Current.GetInstance<ContextManager>();

\_context = contextManager.Context;

}

public void ComitUoW()

{

**\_context.SaveChanges();**

}

}

vamos alterar a classe de RepositoryBase para alterar a forma como recebe o context

public class RepositoryBase<TEntity> : IDisposable, IRepositoryBase<TEntity> where TEntity : class

{

protected LearningCloudContext ContextDB { get; private set; }

public RepositoryBase()

{

ContextDB = new LearningCloudContext();

}

public void Add(TEntity obj)

{

ContextDB.Set<TEntity>().Add(obj);

}

para

public class RepositoryBase<TEntity>: IDisposable, IRepositoryBase<TEntity> where TEntity: class

{

protected **DbContext** ContextDB { get; private set;}

public RepositoryBase()

{

**var contextManager = ServiceLocator.Current.GetInstance<ContextManager<LearningCloudContext>>();**

**ContextDB = contextManager.GetContext;**

}

public void Add(TEntity obj)

{

ContextDB.Set<TEntity>().Add(obj);

}

adiciona os Usings

**using Microsoft.Practices.ServiceLocation;**

**using LearningCloud.Infra.Data.EntityFramework;**

será necessário inicializar o mapeamento do SimpleInjector para que ao se iniciar a aplicação ela mapeie o container e lhe permita utilizá-lo

vamos instalar os seguintes pacotes na camada de apresentação MVC:

* Install-Package SimpleInjector
* Install-Package SimpleInjector.Integration.Web.Mvc
* Install-Package CommonServiceLocator
* Install-Package WebActivatorEx

Instalar o SimpleInjector

* Package Manager Console
  + seleciona o *Default project* (LearningCloud.MVC)
  + Install-Package SimpleInjector

Instalar o SimpleInjector.Integration.Web.Mvc

* Package Manager Console
  + seleciona o *Default project* (LearningCloud.MVC)
  + Install-Package SimpleInjector.Integration.Web.Mvc

Instalar o CommonServiceLocator

* Package Manager Console
  + seleciona o *Default project* (LearningCloud.MVC)
  + Install-Package CommonServiceLocator

E Instalar também o WebActivatorEx

* Package Manager Console
  + seleciona o *Default project* (LearningCloud.MVC)
  + Install-Package WebActivatorEx

na pasta de App\_Start (0 - Presentation)

* 0 - Presentation > LearningCloud.MVC
* Pasta App\_Start (Clica com Direito)
* Add
* Class
* Name: **SimpleInjectorInitializer.cs**
* deve ser public
* definir como static

Add reference Infra.CrossCutting.IoC

cria método public static void Initialize()

e cria método private static void InitializeContainer(Container container)

ficando:

using System.Reflection;

using System.Web.Mvc;

using SimpleInjector;

using SimpleInjector.Integration.Web;

using SimpleInjector.Integration.Web.Mvc;

using WebActivatorEx;

using LearningCloud.Infra.CrossCutting.IoC;

using LearningCloud.MVC.App\_Start;

[assembly: PostApplicationStartMethod(typeof(SimpleInjectorInitializer), "Initialize")]

namespace LearningCloud.MVC.App\_Start

{

public static class SimpleInjectorInitializer

{

public static void Initialize()

{

var container = new Container();

container.Options.DefaultScopedLifestyle = new WebRequestLifestyle();

// Chamada dos módulos do Simple Injector

InitializeContainer(container);

//// Necessário para registrar o ambiente do Owin que é dependência do Identity

//// Feito fora da camada de IoC para não levar o System.Web para fora

////container.RegisterPerWebRequest(() =>

////{

//// if (HttpContext.Current != null && HttpContext.Current.Items["owin.Environment"] == null && container.IsVerifying())

//// {

//// return new OwinContext().Authentication;

//// }

//// return HttpContext.Current.GetOwinContext().Authentication;

////}, Lifestyle.Scoped);

container.RegisterMvcControllers(Assembly.GetExecutingAssembly());

container.Verify();

DependencyResolver.SetResolver(new SimpleInjectorDependencyResolver(container));

}

private static void InitializeContainer(Container container)

{

BootStrapper.RegisterServices(container);

}

}

}

Criar Interface para o AssinaturaNivelRepository na camada de Domain

Adiciona a interface de repositorio para **IAssinaturaNivelRepository** na camada de Domain na pasta Interfaces / Repositories

* 3 - Domain
* Pasta Interfaces
* Pasta Repositories (Clica com Direito)
  + Add - New Item
  + Interface
  + **IAssinaturaNivelRepository**.cs (INomeInterface.cs)
  + definir como public

using System.Collections.Generic;

using LearningCloud.Domain.Entities;

namespace LearningCloud.Domain.Interfaces.Repositories

{

public interface IAssinaturaNivelRepository: IRepositoryBase<AssinaturaNivel>

{

IEnumerable<AssinaturaNivel> GetByStatus(string status);

}

}

Adiciona a classe concreta de repositorio para **AssinaturaNivelRepository** na camada de Infra.Data na pasta Repositories

* 4 - Infra / 4.1 - Data
* Pasta Repositories(Clica com Direito)
  + Add - Class
  + **AssinaturaNivelRepository**.cs
  + definir como public

Herdando da RepositoryBase<AssinaturaNivel> implementando a interface I**AssinaturaNivelRepository**

incluir

**using System.Collections.Generic;**

**using System.Linq;**

**using LearningCloud.Domain.Entities;**

**using LearningCloud.Domain.Interfaces.Repositories;**

Implementa Interfaces da **IAssinaturaNivelRepository** na **AssinaturaNivelRepository**

Clica com o direito em **IAssinaturaNivelRepository** que está sendo implementada na classe **AssinaturaNivelRepository**

Implement Interface / Implement Interface

ficando :

using System.Collections.Generic;

using System.Linq;

using LearningCloud.Domain.Entities;

using LearningCloud.Domain.Interfaces.Repositories;

namespace LearningCloud.Infra.Data.Repositories

{

public class AssinaturaNivelRepository: RepositoryBase<AssinaturaNivel>, IAssinaturaNivelRepository

{

public IEnumerable<AssinaturaNivel> GetByStatus(string status)

{

string[] arrayStatus = status.Split(',');

return ContextDB.Set<AssinaturaNivel>().Where(a => arrayStatus.Contains(a.AssinaturaNivel\_Status)).ToList();

}

}

}

Vamos alterar o texto da tag <title> no arquivo \_Layout para exibir o nome da nossa aplicação

de

|  |
| --- |
| <head>  <meta charset="utf-8" />  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>**@ViewBag.Title - My ASP.NET Application**</title>  @Styles.Render("~/Content/css")  @Scripts.Render("~/bundles/modernizr")  </head> |

para:

|  |
| --- |
| <head>  <meta charset="utf-8" />  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>**[LearningCloud] - @ViewBag.Title**</title>  @Styles.Render("~/Content/css")  @Scripts.Render("~/bundles/modernizr")  </head> |

Vamos alterar o texto da tag <footer> no arquivo \_Layout para exibir o nome do criador do sistema

de:

|  |
| --- |
| <footer>  <p>&copy; @DateTime.Now.Year - **My ASP.NET Application**</p>  </footer> |

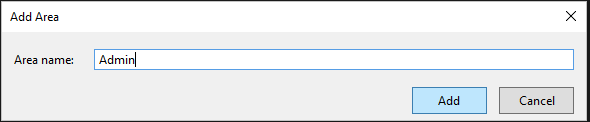
Para:

|  |
| --- |
| <footer>  <p>&copy; @DateTime.Now.Year - **Clayton Gandra**</p>  </footer> |

Vamos Criar uma área “Admin” no projeto MVC

na camada de Apresentação (0 - Presentation)

* 0 - Presentation
* LearningCloud.MVC (Clica com Direito)
* Add
* Area…
* Area name: Admin



Clique no botão Add e note que é criada uma nova pasta chamada área, contendo a pasta Admin, que é o nome da área criada. E como o projeto é MVC, são criadas as mesmas estruturas dos Controllers, Models e Views para esta nova área Admin, Sendo assim, todo Controller referente à administração deverá ser inserido aqui

* + Renomeia a pasta Models para ViewModels

Agora, adicione um novo Controller chamado PanelController dentro da pasta Admin/Controllers,

Adiciona a Controller Panel na area Admin na pasta Controllers

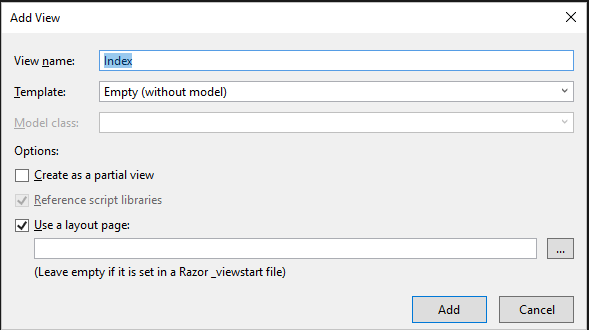
* Admin / Controllers
* Pasta Controllers (Clica com Direito)
  + Add - Controller...
  + MVC 5 Controller - Empty
  + PanelController

contém apenas a Action Index que retorna uma View. Observe o namespace completo, isto influenciará no arquivo de configuração da rota e nas chamadas dos links na página principal do projeto ou qualquer outra página.

namespace LearningCloud.MVC.Areas.Admin.Controllers

|  |
| --- |
| **using System.Web.Mvc;**  **namespace LearningCloud.MVC.Areas.Admin.Controllers**  **{**  **public class PanelController : Controller**  **{**  **// GET: Admin/Panel**  **public ActionResult Index()**  **{**  **return View();**  **}**  **}**  **}** |

Como essa Action Index retorna uma View, precisamos criá-la. Portanto, clique com o botão direito no nome da Action Index e selecione Add View. O nome da View será Index e o template será vazio



é criado automaticamente o arquivo web.config. Aqui é um ponto chave para o entendimento sobre navegação. Também é criado o arquivo AdminAreaRegistration.cs nesta pasta, contendo o método RegisterArea. Note como que a rota está definida, sendo: Admin/controller/action/id.

|  |
| --- |
| using System.Web.Mvc;  namespace LearningCloud.MVC.Areas.Admin  {  public class AdminAreaRegistration : AreaRegistration  {  public override string AreaName  {  get  {  return "Admin";  }  }  public override void RegisterArea(AreaRegistrationContext context)  {  context.MapRoute(  "Admin\_default",  **"Admin/{controller}/{action}/{id}",**  new { action = "Index", id = UrlParameter.Optional }  );  }  }  } |

No arquivo AdminAreaRegistration.cs e inclua o nome do controller = “Panel” na lista de parâmetros default e o name space.

|  |
| --- |
| public override string AreaName  {  get  {  return "Admin";  }  }  public override void RegisterArea(AreaRegistrationContext context)  {  context.MapRoute(  "Admin\_default",  **"Admin/{controller}/{action}/{id}",**  new { **controller = "Panel",** action = "Index", id = UrlParameter.Optional },  **new[] { "LearningCloud.MVC.Areas.Admin.Controllers" }**  );  } |

Na classe RouteConfig da pasta App\_Start

Inclua o name space para o home da pagina fora de admin

|  |
| --- |
| public class RouteConfig  {  public static void RegisterRoutes(RouteCollection routes)  {  routes.IgnoreRoute("{resource}.axd/{\*pathInfo}");  routes.MapRoute(  name: "Default",  url: "{controller}/{action}/{id}",  defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional },  **namespaces: new[] { "LearningCloud.MVC.Controllers" }**  );  }  } |

Vamos criar uma página de leiaute para a área Admin

Adiciona o Layout.cshtml na area Admin na pasta Views / Shared

* Admin / Views
* Pasta Shared (Clica com Direito)
  + Add - New Item
  + MVC 5 Layout Page(Razor)
  + Name: \_Layout.cshtml

|  |
| --- |
| **<!DOCTYPE html>**  **<html>**  **<head>**  **<meta name="viewport" content="width=device-width" />**  **<title>@ViewBag.Title</title>**  **</head>**  **<body>**  **<div>**  **@RenderBody()**  **</div>**  **</body>**  **</html>** |

Alterar para ficar igual ao \_Layout principal

|  |
| --- |
| **<!DOCTYPE html>**  **<html>**  **<head>**  **<meta charset="utf-8" />**  **<meta name="viewport" content="width=device-width, initial-scale=1.0">**  **<title>[LearningCloud] - @ViewBag.Title</title>**  **@Styles.Render("~/Content/css")**  **@Scripts.Render("~/bundles/modernizr")**  **</head>**  **<body>**  **<div class="navbar navbar-inverse navbar-fixed-top">**  **<div class="container">**  **<div class="navbar-header">**  **<button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">**  **<span class="icon-bar"></span>**  **<span class="icon-bar"></span>**  **<span class="icon-bar"></span>**  **</button>**  **@Html.ActionLink("Application name", "Index", "Home", new { area = "" }, new { @class = "navbar-brand" })**  **</div>**  **<div class="navbar-collapse collapse">**  **<ul class="nav navbar-nav">**  **<li>@Html.ActionLink("Home", "Index", "Home")</li>**  **<li>@Html.ActionLink("About", "About", "Home")</li>**  **<li>@Html.ActionLink("Contact", "Contact", "Home")</li>**  **</ul>**  **</div>**  **</div>**  **</div>**  **<div class="container body-content">**  **@RenderBody()**  **<hr />**  **<footer>**  **<p>&copy; @DateTime.Now.Year - Clayton Gandra</p>**  **</footer>**  **</div>**  **@Scripts.Render("~/bundles/jquery")**  **@Scripts.Render("~/bundles/bootstrap")**  **@RenderSection("scripts", required: false)**  **</body>**  **</html>** |

Adiciona a referência do Domain na camada de MVC

* LearningCloud.MVC
  + References (Clica com Direito)
  + Add References
  + Marca a opção “LearningCloud.Domain”

Criar viewmodel de **Aula na pasta dentro da área Admin**

na pasta de ViewModels da **área Admin** (0 - Presentation / Areas / **Admin /** ViewModels )

* 0 - Presentation > LearningCloud.MVC > Areas > **Admin**
* Pasta ViewModels (Clica com Direito)
* Add
* Class
* Name: **AulaViewModel.cs**

copia os campos da classe Aula da entidade da camada de Domain

public int Aula\_Id { get; set; }

public string Aula\_Titulo { get; set; }

public string Aula\_TipoConteudo { get; set; }

public string Aula\_Descricao { get; set; }

public string Aula\_Prerequisitos { get; set; }

public string Aula\_Imagem { get; set; }

public string Aula\_TempoVideo { get; set; }

public string Aula\_Video { get; set; }

public string Aula\_ConteudoEscrito { get; set; }

public string Aula\_Status { get; set; }

public int Aula\_CodigoAssinaturaNivel { get; set; }

public virtual AssinaturaNivel Aula\_AssinaturaNivel { get; set; }

public DateTime Aula\_DataCadastro { get; set; }

public DateTime? Aula\_DataAlteracao { get; set; }

adiciona os Usings  
**using System;**

**using System.ComponentModel;**

**using System.ComponentModel.DataAnnotations;**

**using System.ComponentModel.DataAnnotations.Schema;**

**using System.Web.Mvc;**

**using LearningCloud.Domain.Entities;**

Adiciona os **DataAnnotations** para os campos ficando:

using System;

using System.ComponentModel;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

using LearningCloud.Domain.Entities;

namespace LearningCloud.MVC.Areas.Admin.ViewModels

{

public class AulaViewModel

{

**[Key]**

public int Aula\_Id { get; set; }

**[Required(ErrorMessage = "Preencha o campo Título.")]**

**[StringLength(200, ErrorMessage = "O {0} deve possuir no mínimo {2} e máximo {1} caracteres.", MinimumLength = 2)]**

**[DisplayName("Título")]**

public string Aula\_Titulo { get; set; }

[DisplayName("Tipo Conteúdo")]

[Required(ErrorMessage = "Informe o tipo do conteúdo.")]

public string Aula\_TipoConteudo{ get; set; }

**[AllowHtml]**

**[DisplayName("Descrição")]**

**[AllowHtml]**

**[DisplayName("Descrição")]**

public string Aula\_Descricao{ get; set; }

[AllowHtml]

**[DisplayName("Pré-requisitos")]**

public string Aula\_Prerequisitos{ get; set; }

[DisplayName("Imagem")]

public string Aula\_Imagem{ get; set; }

**[DisplayName("Tempo do Vídeo")]**

public string Aula\_TempoVideo{ get; set; }

**[DisplayName("Vídeo")]**

public string Aula\_Video{ get; set; }

[AllowHtml]

// [Column(TypeName = "varchar(MAX)"), MaxLength]

[DisplayName("Conteúdo da Aula")]

public string Aula\_ConteudoEscrito { get; set; }

**[DisplayName("Status")]**

**[Column(TypeName = "char")]**

**[Required(ErrorMessage = "Informe o status da aula.")]**

public string Aula\_Status{ get; set; }

**[DisplayName("Disponível a partir")]**

**[Required(ErrorMessage = "Selecione o nível de disponibilidade.")]**

public int Aula\_CodigoAssinaturaNivel{ get; set; }

public virtual AssinaturaNivel Aula\_AssinaturaNivel{ get; set; }

[ScaffoldColumn(false)]

public DateTime Aula\_DataCadastro{ get; set; }

[ScaffoldColumn(false)]

public DateTime Aula\_DataAlteracao{ get; set; }

}

}

Criar mapeamento de viewModel com entidade de Domínio

Criar pasta para o AutoMapper

na camada de Apresentação (0 - Presentation)

* 0 - Presentation
* LearningCloud.MVC (Clica com Direito)
* Add
* New Folder
* Name: AutoMapper

Criar três classes na pasta do AutoMapper

a primeira é AutoMapperConfig.cs

na pasta de AutoMapper (0 - Presentation / AutoMapper)

* 0 - Presentation > LearningCloud.MVC > AutoMapper
* Pasta AutoMapper(Clica com Direito)
* Add
* Class
* Name: **AutoMapperConfig.cs**

A outra DomainToViewModelMappingProfile

na pasta de AutoMapper (0 - Presentation / AutoMapper)

* 0 - Presentation > LearningCloud.MVC > AutoMapper
* Pasta AutoMapper(Clica com Direito)
* Add
* Class
* Name: DomainToViewModelMappingProfile**.cs**

e a ultima é ViewModelToDomainMappingProfile

na pasta de AutoMapper (0 - Presentation / AutoMapper)

* 0 - Presentation > LearningCloud.MVC > AutoMapper
* Pasta AutoMapper(Clica com Direito)
* Add
* Class
* Name: ViewModelToDomainMappingProfile**.cs**

Instalar AutoMapper

* Package Manager Console
  + seleciona o *Default project* (LearningCloud.MVC)
  + Install-Package AutoMapper

Na classe DomainToViewModelMappingProfile herdar : **Profile**

adicionar o using do AutoMapper

**using AutoMapper;**

**Criar um construtor para o mapeamento**

public DomainToViewModelMappingProfile()

{

}

configura o mapeamento das entidades de domínio para viewModels

public DomainToViewModelMappingProfile()

{

**CreateMap<Aula, AulaViewModel>();**

}

adicionar os usings de entidades de dominio e viewModels

**using LearningCloud.Domain.Entities;**

**using LearningCloud.MVC.Areas.Admin.ViewModels;**

Ficando:

using AutoMapper;

using LearningCloud.Domain.Entities;

using LearningCloud.MVC.Areas.Admin.ViewModels;

namespace LearningCloud.MVC.AutoMapper

{

public class DomainToViewModelMappingProfile : Profile

{

public DomainToViewModelMappingProfile()

{

CreateMap<Aula, AulaViewModel>();

}

}

}

agora na classe **ViewModelToDomainMappingProfile** faremos o mapeamento ao contrário

Ficando:

using AutoMapper;

using LearningCloud.Domain.Entities;

using LearningCloud.MVC.Areas.Admin.ViewModels;

namespace LearningCloud.MVC.AutoMapper

{

public class ViewModelToDomainMappingProfile : Profile

{

public ViewModelToDomainMappingProfile()

{

CreateMap<**AulaViewModel**, **Aula**>();

}

}

}

Na classe AutoMapperConfig criar metodo **public static void RegisterMappings()**

public static void RegisterMappings()

{

}

adiciona o using do Entity

**using AutoMapper;**

e implementa a inicialização do mapeamento adicionando os perfis de mapeamento das classes que criamos

**Mapper.Initialize(x =>**

**{**

**x.AddProfile<DomainToViewModelMappingProfile>();**

**x.AddProfile<ViewModelToDomainMappingProfile>();**

**});**

**Ficando:**

using AutoMapper;

namespace LearningCloud.MVC.AutoMapper

{

public class AutoMapperConfig

{

public static void RegisterMappings()

{

Mapper.Initialize(x =>

{

x.AddProfile<DomainToViewModelMappingProfile>();

x.AddProfile<ViewModelToDomainMappingProfile>();

});

}

}

}

**Registrar a configuração do** AutoMapper no arquivo Global.asax no método Application\_Start()

ficando

using System.Web.Mvc;

using System.Web.Optimization;

using System.Web.Routing;

namespace LearningCloud.MVC

{

public class MvcApplication : System.Web.HttpApplication

{

protected void Application\_Start()

{

AreaRegistration.RegisterAllAreas();

FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);

RouteConfig.RegisterRoutes(RouteTable.Routes);

BundleConfig.RegisterBundles(BundleTable.Bundles);

**AutoMapperConfig.RegisterMappings();**

}

}

}

Adicione o using do AutoMapper que criamos no MVC

**using LearningCloud.MVC.AutoMapper;**

Vamos criar a Interface de Serviço base na camada de Domain

IServiceBase

Adiciona a interface genérica de serviço na camada de Domain na pasta Interfaces / Services

* 3 - Domain
* LearningCloud.Domain
* Interfaces
* Pasta Services(Clica com Direito)
  + Add - New Item
  + Interface
  + IServiceBase.cs (INomeInterface.cs)
  + definir como public

**Recebendo uma entidade genérica de TEntity, tratando quando for uma class (where TEntity: class)**

<TEntity> where TEntity: class

**using System.Collections.Generic;**

implementando os mesmos métodos da IRepositoryBase<TEntity> where TEntity: class

void Add(TEntity obj);

TEntity GetById(int id);

IEnumerable<TEntity> GetAll();

void Update(TEntity obj);

void Remove(TEntity obj);

void Dispose();

Ficando:

using System.Collections.Generic;

namespace LearningCloud.Domain.Interfaces.Services

{

public interface IServiceBase<TEntity> where TEntity: class

{

void Add(TEntity obj);

TEntity GetById(int id);

IEnumerable<TEntity> GetAll();

void Update(TEntity obj);

void Remove(TEntity obj);

void Dispose();

}

}

Vamos criar as classes concretas de serviço na camada de domínio

Adiciona a classe de serviço na camada de Domain na pasta Services

* 3 - Domain
* LearningCloud.Domain
* Pasta Services(Clica com Direito)
  + Add - Class...
  + ServiceBase.cs (NomeClass.cs)
  + definir como public

Recebendo TEntity, com : IDisposable para poder destruir essa instância e implementa a interface de IServiceBase<> Recebendo TEntity, quando TEntiti for uma classe (where TEntity : class)

**<TEntity> : IDisposable, IServiceBase<TEntity> where TEntity : class**

adiciona os usings

**using System;**

**using LearningCloud.Domain.Interfaces.Services;**

Implementa interface de **IServiceBase**

Clica com o direito em **IServiceBase** que está sendo implementada na classe **ServiceBase**

Implement Interface / Implement Interface

public class ServiceBase<TEntity> : IDisposable, IServiceBase<TEntity> where TEntity : class

{

public void Add(TEntity obj)

{

throw new NotImplementedException();

}

public TEntity GetById(int id)

{

throw new NotImplementedException();

}

public System.Collections.Generic.IEnumerable<TEntity> GetAll()

{

throw new NotImplementedException();

}

public void Update(TEntity obj)

{

throw new NotImplementedException();

}

public void Remove(TEntity obj)

{

throw new NotImplementedException();

}

public void Dispose()

{

throw new NotImplementedException();

}

}

precisamos instanciar o repository, mas não podemos ter dependência com a infra data na camada de domínio, então vamos criar um campo de IrepositoryBase genérico e criar um construtor para injetar a dependência de repository

**private readonly IRepositoryBase<TEntity> \_repository;**

adiciona o using

**using LearningCloud.Domain.Interfaces.Repositories;**

(ctor + Tab)

**public ServiceBase()**

**{**

**}**

passa a interface do repositoryBase por injeção

public ServiceBase(**IRepositoryBase<TEntity> repository**)

{

**\_repository = repository;**

}

e vamos implementar os métodos que a interface exige

public void Add(TEntity obj)

{

**\_repository.Add(obj);**

}

public TEntity GetById(int id)

{

**return \_repository.GetById(id);**

}

public IEnumerable<TEntity> GetAll()

{

**return \_repository.GetAll();**

}

public void Update(TEntity obj)

{

**\_repository.Update(obj);**

}

public void Remove(TEntity obj)

{

**\_repository.Remove(obj);**

}

public void Dispose()

{

**\_repository.Dispose();**

}

Adicionar using para o IEnumerable

**using System.Collections.Generic;**

Vamos criar a Interface de Serviço base na camada de Domain para

IAulaService

Adiciona a interface aula especializada de serviço na camada de Domain na pasta Interfaces / Services

* 3 - Domain > Interfaces
* Pasta Services(Clica com Direito)
  + Add - New Item
  + Interface
  + IAulaService.cs (INomeInterface.cs)
  + definir como public

Herdando de : IServiceBase<Aula> passando <Aula>

adicionar using

**using LearningCloud.Domain.Entities;**

ficando:

using LearningCloud.Domain.Entities;

namespace LearningCloud.Domain.Interfaces.Services

{

public interface IAulaService : IServiceBase<Aula>

{

IEnumerable<Aula> GetByStatus(string status);

}

}

Vamos criar a classe concretas de serviço especializado na camada de domínio

Adiciona a classe AulaService de serviço na camada de Domain na pasta Services

* 3 - Domain
* LearningCloud.Domain
* Pasta Services(Clica com Direito)
  + Add - Class...
  + AulaService.cs (NomeClass.cs)
  + definir como public

Herdando : ServiceBase<Aula> passando <Aula> e implementando a interface IAulaService

public class AulaService : ServiceBase<Aula>, IAulaService

adiciona os usings

**using LearningCloud.Domain.Entities;**

**using LearningCloud.Domain.Interfaces.Services;**

Precisamos criar um construtor com parâmetro para a ServiceBase, ja que essa classe espera um parametro no construtor.

vamos criar um campo somente leitura de IAulaRepository

**private readonly IAulaRepository \_aulaRepository;**

adicona o using

**using LearningCloud.Domain.Interfaces.Repositories;**

cria o construtor e inicializa o campo

(ctor + Tab)

public AulaService(IAulaRepository aulaRepository)

{

\_aulaRepository = aulaRepository;

}

temos que passar o construtor herdando da classe base recebendo o repository que criamos

O ServiceBase espera no construtor dele o **IRepositoryBase**

*public ServiceBase(****IRepositoryBase****<TEntity> repository)*

*{*

*\_repository = repository;*

*}*

nos vamos passar o aulaRepository que criamos, esse repositpry implementa o **IRepositoryBase**

:base(aulaRepository)

Ficando:

using LearningCloud.Domain.Entities;

using LearningCloud.Domain.Interfaces.Repositories;

using LearningCloud.Domain.Interfaces.Services;

namespace LearningCloud.Domain.Services

{

public class AulaService : ServiceBase<Aula>, IAulaService

{

private readonly IAulaRepository \_aulaRepository;

public AulaService(IAulaRepository aulaRepository)

**: base(aulaRepository)**

{

\_aulaRepository = aulaRepository;

}

}

}

(Ps: **AulaService** Não contém um construtor com 0 argumentos)

Vamos implementar os metodos da interface IAulaService

public IEnumerable<Aula> GetByStatus(string status)

{

throw new NotImplementedException();

}

Ficando :

public IEnumerable<Aula> GetByStatus(string status)

{

**return \_aulaRepository.GetByStatus(status);**

}

Vamos criar a Interface de Serviço base na camada de Domain para

IAssinaturaNivelService

Adiciona a interface AssinaturaNivel especializada de serviço na camada de Domain na pasta Interfaces / Services

* 3 - Domain > Interfaces
* Pasta Services(Clica com Direito)
  + Add - New Item
  + Interface
  + IAssinaturaNivelService.cs (INomeInterface.cs)
  + definir como public

Herdando de : IServiceBase<AssinaturaNivel> passando <AssinaturaNivel>

adicionar using

**using LearningCloud.Domain.Entities;**

ficando:

using LearningCloud.Domain.Entities;

namespace LearningCloud.Domain.Interfaces.Services

{

public interface IAssinaturaNivelService : IServiceBase<AssinaturaNivel>

{

}

}

implementa o mesmo método da interface de repository de AssinaturaNivel

public interface IAssinaturaNivelService : IServiceBase<AssinaturaNivel>

{

**IEnumerable<AssinaturaNivel> GetByStatus(string status);**

}

adiciona o using

**using System.Collections.Generic;**

Vamos criar a classe concretas de serviço especializado na camada de domínio

Adiciona a classe **AssinaturaNivel**Service de serviço na camada de Domain na pasta Services

* 3 - Domain
* LearningCloud.Domain
* Pasta Services(Clica com Direito)
  + Add - Class...
  + **AssinaturaNivel**Service.cs (NomeClass.cs)
  + definir como public

Herando : ServiceBase<**AssinaturaNivel**> passando <**AssinaturaNivel**> e implementando a interface **IAssinaturaNivelService**

adiciona os usings

**using LearningCloud.Domain.Entities;**

**using LearningCloud.Domain.Interfaces.Services;**

Precisamos criar um construtor com parâmetro para a ServiceBase, ja que essa classe espera um parâmetro no construtor.

vamos criar um campo somente leitura de  **IAssinaturaNivelRepository**

**private readonly IAssinaturaNivelRepository \_assinaturaNivelRepository;**

adicona o using

**using LearningCloud.Domain.Interfaces.Repositories;**

cria o construtor e inicializa o campo

temos que passar o construtor herdando da classe base recebendo o repository que criamos

Nós vamos passar o  **assinaturaNivelRepository** que criamos

**private readonly IAssinaturaNivelRepository \_assinaturaNivelRepository;**

**public AssinaturaNivelService(IAssinaturaNivelRepository assinaturaNivelRepository)**

**: base(assinaturaNivelRepository)**

**{**

**\_assinaturaNivelRepository = assinaturaNivelRepository;**

**}**

adicona o using

**using LearningCloud.Domain.Interfaces.Repositories;**

Ficando

using LearningCloud.Domain.Entities;

using LearningCloud.Domain.Interfaces.Repositories;

using LearningCloud.Domain.Interfaces.Services;

namespace LearningCloud.Domain.Services

{

public class AssinaturaNivelService : ServiceBase<AssinaturaNivel>, IAssinaturaNivelService

{

private readonly IAssinaturaNivelRepository \_assinaturaNivelRepository;

public AssinaturaNivelService(IAssinaturaNivelRepository assinaturaNivelRepository)

: base(assinaturaNivelRepository)

{

\_assinaturaNivelRepository = assinaturaNivelRepository;

}

}

}

Precisamos implementar os métodos específicos da interface **IAssinaturaNivelService** que herdamos na classe **AssinaturaNivelService**

adiciona o using

**using System.Collections.Generic;**

Implementa interface de **IAssinaturaNivelService**

Clica com o direito em **IAssinaturaNivelService** que está sendo implementada na classe **AssinaturaNivelService**

Implement Interface / Implement Interface

public class AssinaturaNivelService : ServiceBase<AssinaturaNivel>, IAssinaturaNivelService

{

private readonly IAssinaturaNivelRepository \_assinaturaNivelRepository;

public AssinaturaNivelService(IAssinaturaNivelRepository assinaturaNivelRepository)

: base(assinaturaNivelRepository)

{

\_assinaturaNivelRepository = assinaturaNivelRepository;

}

**public IEnumerable<AssinaturaNivel> GetByStatus(string status)**

**{**

**throw new System.NotImplementedException();**

**}**

}

implementa o método do repository que estamos recebendo por injeção de dependência

public IEnumerable<AssinaturaNivel> GetByStatus(string status)

{

**return \_assinaturaNivelRepository.GetByStatus(status);**

}

Vamos para a camada de aplicação

Interfaces

Criar Pasta para organizar a camada de Aplicação (1(2) - Application)

* 1(2) - Application
* LearningCloud.Application (Clica com Direito)
  + Add - New Folder
  + Interfaces

vamos criar as interfaces de serviço dentro da camada de aplicação

IAppServiceBase

Adiciona a interface generica de serviço na camada de Application na pasta Interfaces

* 1(2) - Application
* LearningCloud.Application
* Pasta Interfaces (Clica com Direito)
  + Add - New Item
  + Interface
  + IAppServiceBase.cs (INomeInterface.cs)
  + definir como public

**Recebendo uma entidade genérica de TEntity, tratando quando for uma class (where TEntity: class)**

public interface IAppServiceBase<TEntity> where TEntity : class

**using System.Collections.Generic;**

implementando os métodos de consulta IRepositoryBase mais os métodos da UnitOfWork

TEntity GetById(int id);

IEnumerable<TEntity> GetAll();

void BeginTransactionUoW();

void ComitUoW();

void Dispose();

Ficando:

using System.Collections.Generic;

namespace LearningCloud.Application.Interfaces

{

public interface IAppServiceBase<TEntity> where TEntity : class

{

TEntity GetById(int id);

IEnumerable<TEntity> GetAll();

void BeginTransactionUoW();

void ComitUoW();

void Dispose();

}

}

Adiciona a referência do Domain na camada de Application

* LearningCloud.Application
  + References (Clica com Direito)
  + Add References
  + Marca a opção “LearningCloud.Domain”

vamos criar a interface especializada IAulaAppService de serviço dentro da camada de aplicação

IAulaAppService

Adiciona a interface IAulaAppService especializada de serviço na camada de Application na pasta Interfaces

* 1 - Application
* LearningCloud.Application
* Pasta Interfaces (Clica com Direito)
  + Add - New Item
  + Interface
  + IAulaAppService.cs (INomeInterface.cs)
  + definir como public

Herdando de : IAppServiceBase<Aula> passando <Aula>

adicionar using

**using LearningCloud.Domain.Entities;**

ficando:

using LearningCloud.Domain.Entities;

namespace LearningCloud.Application.Interfaces

{

public interface IAulaAppService: IAppServiceBase<Aula>

{

}

}

vamos implementar os métodos para as funções básicas para aula

using LearningCloud.Domain.Entities;

namespace LearningCloud.Application.Interfaces

{

interface IAulaAppService: IAppServiceBase<Aula>

{

**IEnumerable<Aula> GetByStatusAula(string status);**

**void CreateAula(Aula aula);**

**void UpdateAula(Aula aula);**

**// void RemoveAula(Aula aula);**

**void ChangeStatusAula(Aula aula, string status);**

}

}

agora vamos criar a interface especializada de IAssinaturaNivelAppService dentro da camada de aplicação

Adiciona a interface IAssinaturaNivelAppService especializada de serviço na camada de Application na pasta Interfaces

* 1 - Application
* LearningCloud.Application
* Pasta Interfaces (Clica com Direito)
  + Add - New Item
  + Interface
  + IAssinaturaNivelAppService.cs (INomeInterface.cs)
  + definir como public

Herdando de : IAppServiceBase<AssinaturaNivel> passando <AssinaturaNivel>

adicionar using

**using LearningCloud.Domain.Entities;**

ficando:

using LearningCloud.Domain.Entities;

namespace LearningCloud.Application.Interfaces

{

interface IAssinaturaNivelAppService: IAppServiceBase<AssinaturaNivel>

{

}

}

vamos implementar o método para as funções básicas para AssinaturaNivel

using LearningCloud.Domain.Entities;

namespace LearningCloud.Application.Interfaces

{

interface IAssinaturaNivelAppService: IAppServiceBase<AssinaturaNivel>

{

**IEnumerable<AssinaturaNivel> GetByStatusAssinaturaNivel(string status);**

}

}

adicionar using

**using System.Collections.Generic;**

Criar Pasta para organizar as classes de serviços na camada de Aplicação (1(2) - Application)

* 1(2) - Application
* LearningCloud.Application (Clica com Direito)
  + Add - New Folder
  + Services

Agora podemos criar as classes concretas de Application Sevices

Adiciona a classe de serviço base na camada de Application

* 1 - Application
* LearningCloud.Application
* Pasta Sevices(Clica com Direito)
  + Add - Class...
  + AppServiceBase.cs (NomeClass.cs)
  + definir como public

Recebendo TEntity, com : IDisposable para poder destruir essa instância e implementa a interface de IAppServiceBase<> Recebendo TEntity, quando TEntity for uma classe (where TEntity : class)

**<TEntity> : IDisposable, IAppServiceBase<TEntity> where TEntity : class**

adiciona os usings

**using System;**

**using System.Collections.Generic;**

**using LearningCloud.Application.Interfaces;**

Implementa interface de **IAppServiceBase**

Clica com o direito em **IAppServiceBase** que está sendo implementada na classe **AppServiceBase**

Implement Interface / Implement Interface

ficando:

using System;

using LearningCloud.Application.Interfaces;

namespace LearningCloud.Application.Services

{

public class AppServiceBase<TEntity> : IDisposable, IAppServiceBase<TEntity> where TEntity : class

{

public TEntity GetById(int id)

{

throw new NotImplementedException();

}

public IEnumerable<TEntity> GetAll()

{

throw new NotImplementedException();

}

public void BeginTransactionUoW()

{

throw new NotImplementedException();

}

public void ComitUoW()

{

throw new NotImplementedException();

}

public void Dispose()

{

throw new NotImplementedException();

}

}

}

a camada de aplicação conversa com a camada de domínio e chama os serviços implementados em domínio. mas eu não posso instanciar as classes de serviços diretamente em application para não criar sujeira, vamos implementar através de injeção de dependência. Vamos criar um campo somente leitura do tipo IserviceBase<TEntity>

**private readonly IServiceBase<TEntity> \_serviceBase;**

e vamos criar um construtor inicializando esse campo com a injeção de dependência

**public AppServiceBase(IServiceBase<TEntity> serviceBase)**

**{**

**\_serviceBase = serviceBase;**

**}**

**Adiciona o using de**

**using LearningCloud.Domain.Interfaces.Services;**

Ficando:

using System;

using LearningCloud.Application.Interfaces;

**using LearningCloud.Domain.Interfaces.Services;**

namespace LearningCloud.Application

{

public class AppServiceBase<TEntity> : IDisposable, IAppServiceBase<TEntity> where TEntity : class

{

**private readonly IServiceBase<TEntity> \_serviceBase;**

**public AppServiceBase(IServiceBase<TEntity> serviceBase)**

**{**

**\_serviceBase = serviceBase;**

**}**

public void Add(TEntity obj)

{

throw new NotImplementedException();

}

...

agora vamos implementar os métodos que o IAppServiceBase exige:

**public TEntity GetById(int id)**

**{**

**return \_serviceBase.GetById(id);**

**}**

**public IEnumerable<TEntity> GetAll()**

**{**

**return \_serviceBase.GetAll();**

**}**

public void BeginTransactionUoW()

{

throw new NotImplementedException();

}

public void ComitUoW()

{

throw new NotImplementedException();

}

**public void Dispose()**

**{**

**\_serviceBase.Dispose();**

**}**

Note que ainda não implementamos os metodos **BeginTransactionUoW**() e o **ComitUoW**()

Para implementar esses métodos vamos precisar do ServiceLocator instalado no projeto da camada de Application para instanciar o UnitOfWork

Instalar o CommonServiceLocator na camada de Application

* Package Manager Console
  + seleciona o *Default project* (LearningCloud.Application)
  + Install-Package CommonServiceLocator

CommonServiceLocator instalado na camada de Application podemos implementar os metodos de **BeginTransactionUoW**() e o **ComitUoW**()

vamos criar um campo para instanciar o UnitOfOrk para usarmos nos dois metodos

**private IUnitOfWorkRepository \_unitOfWork;**

adiciona os usings

**using Microsoft.Practices.ServiceLocation;**

**using LearningCloud.Domain.Interfaces.Repositories;**

ficando

**private IUnitOfWorkRepository \_unitOfWork;**

**public void BeginTransactionUoW()**

**{**

**\_unitOfWork = ServiceLocator.Current.GetInstance<IUnitOfWorkRepository>();**

**\_unitOfWork.BeginTransactionUoW();**

**}**

**public void ComitUoW()**

**{**

**\_unitOfWork.CommitUoW();**

**}**

**classe completa ficou:**

**using System;**

**using System.Collections.Generic;**

**using Microsoft.Practices.ServiceLocation;**

**using LearningCloud.Application.Interfaces;**

**using LearningCloud.Domain.Interfaces.Services;**

**using LearningCloud.Domain.Interfaces.Repositories;**

**namespace LearningCloud.Application.Services**

**{**

**public class AppServiceBase<TEntity> : IDisposable, IAppServiceBase<TEntity> where TEntity : class**

**{**

**private readonly IServiceBase<TEntity> \_serviceBase;**

**private IUnitOfWorkRepository \_unitOfWork;**

**public AppServiceBase(IServiceBase<TEntity> serviceBase)**

**{**

**\_serviceBase = serviceBase;**

**}**

**public TEntity GetById(int id)**

**{**

**return \_serviceBase.GetById(id);**

**}**

**public IEnumerable<TEntity> GetAll()**

**{**

**return \_serviceBase.GetAll();**

**}**

**public void BeginTransactionUoW()**

**{**

**\_unitOfWork = ServiceLocator.Current.GetInstance<IUnitOfWorkRepository>();**

**\_unitOfWork.BeginTransactionUoW();**

**}**

**public void ComitUoW()**

**{**

**\_unitOfWork.CommitUoW();**

**}**

**public void Dispose()**

**{**

**\_serviceBase.Dispose();**

**}**

**}**

**}**

Vamos criar a classe concreta **AulaAppService** de serviço de aplicação especializado na camada de Application

Adiciona a classe **AulaAppService** de serviço na camada de Application

* 2 - Application
* LearningCloud.Application
* Pasta Sevices(Clica com Direito)
* Add - Class...
  + **AulaAppService**.cs (NomeClass.cs)
  + definir como public

Herando **:** **AppService**Base<**Aula**> passando <**Aula**> e implementando a interface **IAulaAppService**

**public class AulaAppService : AppServiceBase<Aula>, IAulaAppService**

adiciona os usings

**using LearningCloud.Application.Interfaces;**

**using LearningCloud.Domain.Entities;**

public class AulaAppService : AppServiceBase<Aula>, IAulaAppService

{

}

Implementa Interfaces da AppServiceBase na AulaAppService

Clica com o direito em AppServiceBase que está sendo implementada na classe AulaAppService

Implement Interface / Implement Interface

ficando :

using LearningCloud.Application.Interfaces;

using LearningCloud.Domain.Entities;

namespace LearningCloud.Application.Services

{

class AulaAppService : AppServiceBase<Aula>, IAulaAppService

{

public IEnumerable<Aula> GetByStatusAula(string status)

{

throw new NotImplementedException();

}

public void CreateAula(Aula aula)

{

throw new System.NotImplementedException();

}

public void UpdateAula(Aula aula)

{

throw new System.NotImplementedException();

}

public void ChangeStatusAula(Aula aula, string status)

{

throw new System.NotImplementedException();

}

}

}

vamos criar um campo de IAulaService e criar um construtor para injetar a dependência de IAulaService

**private readonly IAulaService \_aulaService;**

adiciona o using

**using LearningCloud.Domain.Interfaces.Services;**

(ctor + Tab)

**public AulaAppService()**

**{**

**}**

passa a interface do IAulaService por injeção herdando da classe base recebendo a aulaService

public AulaAppService(IAulaService aulaService)

: base(aulaService)

{

\_aulaService = aulaService;

}

e vamos implementar os métodos que a interface exige

public IEnumerable<Aula> GetByStatusAula(string status)

{

**return \_aulaService.GetByStatus(status);**

}

public void CreateAula(Aula aula)

{

**BeginTransactionUoW();**

**\_aulaService.Add(aula);**

**ComitUoW();**

}

public void UpdateAula(Aula aula)

{

**BeginTransactionUoW();**

**\_aulaService.Update(aula);**

**ComitUoW();**

}

public void RemoveAula(Aula aula)

{

**BeginTransactionUoW();**

**\_aulaService.Remove(aula);**

**ComitUoW();**

}

public void ChangeStatusAula(aula aula, string status)

{

**aula.aul\_status = status;**

**UpdateAula(aula);**

**//BeginTransactionUoW();**

**//\_aulaService.Update(aula);**

**//ComitUoW();**

}

Vamos criar a classe concreta **AssinaturaNivelAppService** de serviço de aplicação especializado na camada de Application

Adiciona a classe **AssinaturaNivelAppService** de serviço na camada de Application

* 2 - Application
* LearningCloud.Application
* Pasta Sevices(Clica com Direito)
  + Add - Class...
  + **AssinaturaNivelAppService**.cs (NomeClass.cs)
  + definir como public

Herando : **AppService**Base<**AssinaturaNivel**> passando <**AssinaturaNivel**> e implementando a interface **IAssinaturaNivelAppService**

adiciona os usings

**using LearningCloud.Domain.Entities;**

**using LearningCloud.Application.Interfaces;**

public class **AssinaturaNivelAppService**: AppServiceBase<AssinaturaNivel>, IAssinaturaNivelAppService

{

}

Implementa Interfaces da AppServiceBase na AssinaturaNivelAppService

Clica com o direito em AppServiceBase que está sendo implementada na classe AssinaturaNivelAppService

Implement Interface / Implement Interface

ficando :

using LearningCloud.Domain.Entities;

using LearningCloud.Application.Interfaces;

namespace LearningCloud.Application.Services

{

public class AssinaturaNivelAppService : AppServiceBase<AssinaturaNivel>, IAssinaturaNivelAppService

{

public System.Collections.Generic.IEnumerable<AssinaturaNivel> GetByStatusAssinaturaNivel(string status)

{

throw new System.NotImplementedException();

}

}

}

vamos criar um campo de IAssinaturaNivelService e criar um construtor para injetar a dependência de IAssinaturaNivelService

**private readonly IAssinaturaNivelService \_assinaturaNivelService;**

adiciona o using

**using LearningCloud.Domain.Interfaces.Services;**

(ctor + Tab)

**public AssinaturaNivelAppService()**

**{**

**}**

passa a interface do IAssinaturaNivelService por injeção herdando da classe base recebendo a assinaturaNivelService

public AssinaturaNivelAppService(IAssinaturaNivelService assinaturaNivelService)

: base(assinaturaNivelService)

{

\_assinaturaNivelService = assinaturaNivelService;

}

e vamos implementar o método que a interface exige

public IEnumerable<AssinaturaNivel> GetByStatusAssinaturaNivel(string status)

{

**return \_assinaturaNivelService.GetByStatus(status);**

}

adiciona os usings

**using System.Collections.Generic;**

ficando:

using System.Collections.Generic;

using LearningCloud.Domain.Entities;

using LearningCloud.Application.Interfaces;

using LearningCloud.Domain.Interfaces.Services;

namespace LearningCloud.Application.Services

{

public class AssinaturaNivelAppService : AppServiceBase<AssinaturaNivel>, IAssinaturaNivelAppService

{

private readonly IAssinaturaNivelService \_assinaturaNivelService;

public AssinaturaNivelAppService(IAssinaturaNivelService assinaturaNivelService)

: base(assinaturaNivelService)

{

\_assinaturaNivelService = assinaturaNivelService;

}

public IEnumerable<AssinaturaNivel> GetByStatusAssinaturaNivel(string status)

{

return \_assinaturaNivelService.GetByStatus(status);

}

}

}

Vamos criar um Controller para Aula que irá conversar com a nossa camada de aplicação

AulaController

na pasta de Controllers (0 - Presentation / Areas / **Admin /** Controllers )

* 0 - Presentation > LearningCloud.MVC > Areas > **Admin**
* Pasta Controllers (Clica com Direito)
* Add
* Controller…
* MVC Controller with read/write actions
* [Add]
* Controller Name: AulaController
* [Add]

deixar apenas o using

**using System.Web.Mvc;**

Precisamos da referência da Application na camada de apresentação

Adiciona a referência do Application na camada de MVC

* LearningCloud.MVC
  + References (Clica com Direito)
  + Add References
  + Marca a opção “LearningCloud.Application”

vamos criar os campos de IAulaAppService e IAssinaturaNivelAppService criar um construtor para injetar as dependências desses dois serviços

**private readonly IAulaAppService \_aulaApp;**

**private readonly IAssinaturaNivelAppService \_assinaturaNivelApp;**

adiciona os usings

**using LearningCloud.Application.Interfaces;**

**e cria o construtor com os parâmetros dos serviços de aplicação**

public AulaController(**IAulaAppService aulaApp, IAssinaturaNivelAppService assinaturaNivelApp**)

{

**\_aulaApp = aulaApp;**

**\_assinaturaNivelApp = assinaturaNivelApp;**

}

vamos implementar o ActionResult de Index

// GET: Admin/Aula

public ActionResult Index()

{

IEnumerable<AulaViewModel> listAulaViewModel = Mapper.Map<IEnumerable<Aula>, IEnumerable<AulaViewModel>>(\_aulaApp.GetByStatusAula("A,I"));

return View(listAulaViewModel);

}

adiciona os usings

**using System.Collections.Generic;**

**using LearningCloud.Domain.Entities;**

using LearningCloud.MVC.Areas.Admin.ViewModels;

vamos criar a view para a Index

clica com direito no método View do ActionResult Index()

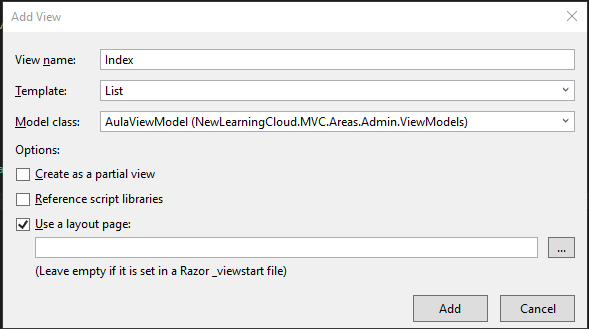
return **View**(aulaViewModel);

Add View…

View name: Index

Template: List

Model class: AulaViewModel (LearningCloud.MVC.Areas.Admin.ViewModels)



vamos implementar o ActionResult de Create [HttpPost]

Adiciona a datanotention [ValidateAntiForgeryToken]

[HttpPost]

**[ValidateAntiForgeryToken]**

public ActionResult Create(FormCollection collection)

{

try

{

// TODO: Add insert logic here

return RedirectToAction("Index");

}

catch

{

return View();

}

}

altera o parâmetro do método Create para receber um AulaViewModel no lugar de FormCollection

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Create(**AulaViewModel aula**)

{

...

e implementando o mapeamento de ViewModel para entidade de Dominio, chamando o método de criação de aula da Applicantion

// POST: Admin/Aula/Create

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Create(AulaViewModel aula)

{

if (ModelState.IsValid)

{

Aula aulaDomain = Mapper.Map<AulaViewModel, Aula>(aula);

\_aulaApp.CreateAula(aulaDomain);

return RedirectToAction("Index");

}

return View(aula);

}

vamos criar a view para a Create

clica com direito no método View do ActionResult Create()

return **View**(aula);

Add View…

View name: Create

Template: Create

Model class: AulaViewModel (LearningCloud.MVC.Areas.Admin.ViewModels)

Add

Instalar **FontAwesome**

* Package Manager Console
  + seleciona o *Default project* (LearningCloud.MVC)
  + **Install-Package FontAwesome**

**criamos os Bundles com o Font-Awesome**

**bundles.Add(new StyleBundle("~/Content/css").Include(**

**"~/Content/bootstrap.css",**

**"~/Content/font-awesome.css",**

**"~/Content/site.css"));**

**para testar**

**registrar as interfaces e classes de application**

**adicionar referencia de appication na camada de crossCutting**

**adicionar usings na Bootsrap**

**using LearningCloud.Domain.Services;**

**using LearningCloud.Domain.Interfaces.Services;**

**using LearningCloud.Application.Services;**

**using LearningCloud.Application.Interfaces;**

**container.Register<IAulaService, AulaService>(Lifestyle.Scoped);**

**container.Register<IAulaRepository, AulaRepository>(Lifestyle.Scoped);**

**container.Register<IAssinaturaNivelAppService, AssinaturaNivelAppService>(Lifestyle.Scoped);**

**container.Register<IAssinaturaNivelService, AssinaturaNivelService>(Lifestyle.Scoped);**

**container.Register<IAssinaturaNivelRepository, AssinaturaNivelRepository>(Lifestyle.Scoped);**

**>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>**

**Erro:**

**No Entity Framework provider found for the ADO.NET provider with invariant name 'System.Data.SqlClient'**

**implementar a verificação no construtor do Context**

**bool instanceExists = System.Data.Entity.SqlServer.SqlProviderServices.Instance != null;**

public class LearningCloudContext : DbContext

{

public LearningCloudContext()

: base("LearningCloud")

{

**bool instanceExists = System.Data.Entity.SqlServer.SqlProviderServices.Instance != null;**

}

**<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Na view Index de Panel**

**@{**

**ViewBag.Title = "Painel Administrativo";**

**}**

<nav class="navbar navbar-inverse">

<div class="container-fluid">

<ul class="nav navbar-nav">

<li><a href="@Url.Action("Index", "Aula", new { Area = "Admin" })"><i class="fa fa-file-video-o">&nbsp;&nbsp;</i>Aulas</a></li>

</ul>

</div>

</nav>

**<div class="jumbotron">**

**<h1>Painel Administrativo</h1>**

**<p class="lead">ASP.NET is a free web framework for building great Web sites and Web applications using HTML, CSS and JavaScript.</p>**

**</div>**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

Vamos alterar a ActionResult Create inicial (não a que recebe o [HttpPost])

// GET: Admin/Aula/Create

public ActionResult Create()

{

return View();

}

para passarmos um SelectList com uma lista de Níveis de assinaturas ativo para a View do formulário de cadastro da aula

Vamos criar uma ViewBag Chamada **assinaturanivel** que receberá o resultado do metodo GetByStatusAssinaturaNivel do serviço de aplicação AssinaturaNivel

**ViewBag.assinaturanivel = new SelectList(\_assinaturaNivelApp.GetByStatusAssinaturaNivel("A"), "AssinaturaNivel\_Id", "AssinaturaNivel\_Titulo");**

**ficando:**

// GET: Admin/Aula/Create

public ActionResult Create()

{

**ViewBag.assinaturanivel = new SelectList(\_assinaturaNivelApp.GetByStatusAssinaturaNivel("A"), "AssinaturaNivel\_Id", "AssinaturaNivel\_Titulo");**

return View();

}

Agora vamos alterar a view para exibir um DropDownList com as AssinaturaNivel

clica com direito no método View do ActionResult Create()

return **View**(aula);

Go To View

na view Creat que será exibida, vamos alterar o EditorFor para um DropDownList que irá receber nossa lista de Níveis de assinatura que criamos na ViewBag.assinaturanivel

<div class="form-group">

@Html.LabelFor(model => model.aul\_status, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.aul\_status, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.aul\_status, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.aul\_fk\_assinaturanivel, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

**@Html.EditorFor(model => model.Aula\_CodigoAssinaturaNivel, new { htmlAttributes = new { @class = "form-control" } })**

@Html.ValidationMessageFor(model => model.aul\_fk\_assinaturanivel, "", new { @class = "text-danger" })

</div>

</div>

alterar para:

**@Html.DropDownList("**assinaturanivel**", null, String.Empty, new { @class = "form-control" })**

<div class="form-group">

@Html.LabelFor(model => model.aul\_fk\_assinaturanivel, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.DropDownList("assinaturanivel", null, String.Empty, new { @class = "form-control" })

@Html.ValidationMessageFor(model => model.aul\_fk\_assinaturanivel, "", new { @class = "text-danger" })

</div>

</div>

já que estamos alterando a View Create, vamos aproveitar e alterar o Status de um campo texto para um radio button com duas opções (Ativo / Inativo) e também o campo Tipo Conteúdo de um campo texto para um radio button com duas opções (Videoaula / Aula Escrita)

<div class="form-group">

@Html.LabelFor(model => model.Aula\_Status, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

**@Html.DropDownListFor(model => model.Aula\_CodigoAssinaturaNivel, (SelectList)ViewBag.assinaturanivel, String.Empty, new { @class = "form-control" })**

@Html.ValidationMessageFor(model => model.Aula\_Status, "", new { @class = "text-danger" })

</div>

</div>

para:

<div class="form-group">

@Html.LabelFor(model => model.Aula\_Status, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

**<label class="radio-inline">@Html.RadioButtonFor(model => model.Aula\_Status, "A", null) Ativo</label>**

**<label class="radio-inline">@Html.RadioButtonFor(model => model.Aula\_Status, "I", null) Inativo</label>**

@Html.ValidationMessageFor(model => model.Aula\_Status, "", new { @class = "text-danger" })

</div>

</div>

E também **TipoConteudo** de

<div class="form-group">

@Html.LabelFor(model => model.Aula\_TipoConteudo, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

**@Html.EditorFor(model => model.Aula\_TipoConteudo, new { htmlAttributes = new { @class = "form-control" } })**

@Html.ValidationMessageFor(model => model.Aula\_TipoConteudo, "", new { @class = "text-danger" })

</div>

</div>

Para

<div class="form-group">

@Html.LabelFor(model => model.Aula\_TipoConteudo, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

**<label class="radio-inline">@Html.RadioButtonFor(model => model.Aula\_TipoConteudo, "E", new { @checked = "checked" }) Aula Escrita</label>**

**<label class="radio-inline">@Html.RadioButtonFor(model => model.Aula\_TipoConteudo, "V", null) Videoaula</label>**

@Html.ValidationMessageFor(model => model.aul\_tipoconteudo, "", new { @class = "text-danger" })

</div>

</div>

Temos que alterar também a ActionResult Create que recebe o post ([HttpPost])

para passar a ViewBag.assinaturanivel também para caso de erro validação de campos o DropDownList seja preenchido com a lista novamente

ficando:

// POST: Admin/Aula/Create

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Create(AulaViewModel aula)

{

if (ModelState.IsValid)

{

var aulaDomain = Mapper.Map<AulaViewModel, Aula>(aula);

\_aulaApp.CreateAula(aulaDomain);

return RedirectToAction("Index");

}

**ViewBag.assinaturanivel = new SelectList(\_assinaturaNivelApp.GetByStatusAssinaturaNivel("A"), "AssinaturaNivel\_Id", "AssinaturaNivel\_Titulo", aula.Aula\_CodigoAssinaturaNivel);**

**return View(aula);**

}

esse mesmo conceito será implementado nas actions de edição (Edit)

**Agora podemos implementar os demais ActionResults da Controller AulaController**

**Details**

**Edit**

**Edit [HttpPost]**

**Delete**

**DeleteConfirmed[HttpPost]**

**teremos também as Actions Inactivate, InactivateConfirmed, Activate.**

Implementa o **Details** método na **AulaController**

**// GET: Admin/Aula/Details/5**

**public ActionResult Details(int id)**

**{**

**Aula aula = \_aulaApp.GetById(id);**

**AulaViewModel aulaViewModel = Mapper.Map<Aula, AulaViewModel>(aula);**

**return View(aulaViewModel);**

**}**

vamos criar a view para a **Details**

clica com direito no método View do ActionResult **Details**(int id)

return **View**();

Add View…

View name: Details

Template: Details

Model class: AulaViewModel (LearningCloud.MVC.Areas.Admin.ViewModels)

Add

Implementa o **Edit** método na **AulaController**

**// GET: Admin/Aula/Edit/5**

**public ActionResult Edit(int id)**

**{**

**Aula aula = \_aulaApp.GetById(id);**

**AssinaturaNivel assinaturaNivelAula = \_assinaturaNivelApp.GetById(aula.Aula\_CodigoAssinaturaNivel);**

**AulaViewModel aulaViewModel = Mapper.Map<Aula, AulaViewModel>(aula);**

**List<AssinaturaNivel> listAssinaturaNivel = new List<AssinaturaNivel>(\_assinaturaNivelApp.GetByStatusAssinaturaNivel("A"));**

**if (assinaturaNivelAula.AssinaturaNivel\_Status == "I")**

**{**

**listAssinaturaNivel.Add(new AssinaturaNivel() { AssinaturaNivel\_Id = assinaturaNivelAula.AssinaturaNivel\_Id, AssinaturaNivel\_Titulo = assinaturaNivelAula.AssinaturaNivel\_Titulo });**

**}**

**SelectList selectlistAssinaturaNivel = new SelectList(listAssinaturaNivel, "AssinaturaNivel\_Id", "AssinaturaNivel\_Titulo", aula.Aula\_CodigoAssinaturaNivel,"");**

**ViewBag.assinaturanivel = selectlistAssinaturaNivel;**

**return View(aulaViewModel);**

**}**

vamos criar a view para a **Edit**

clica com direito no método View do ActionResult **Edit**(int id)

return **View**();

Add View…

View name: Edit

Template: Edit

Model class: AulaViewModel (LearningCloud.MVC.Areas.Admin.ViewModels)

Add

Implementa o método **Edit [HttpPost]** na **AulaController**

**utilizando o [ValidateAntiForgeryToken]**

**// POST: Admin/Aula/Edit/5**

**[HttpPost]**

**[ValidateAntiForgeryToken]**

**public ActionResult Edit(AulaViewModel aula)**

**{**

**if (ModelState.IsValid)**

**{**

**Aula aulaDomain = Mapper.Map<AulaViewModel, Aula>(aula);**

**\_aulaApp.UpdateAula(aulaDomain);**

**return RedirectToAction("Index");**

**}**

**Aula aulaOriginal = \_aulaApp.GetById(aula.Aula\_Id);**

**AssinaturaNivel assinaturaNivelAula = \_assinaturaNivelApp.GetById(aulaOriginal.Aula\_CodigoAssinaturaNivel);**

**List<AssinaturaNivel> listAssinaturaNivel = new List<AssinaturaNivel>(\_assinaturaNivelApp.GetByStatusAssinaturaNivel("A"));**

**if (assinaturaNivelAula.AssinaturaNivel\_Status == "I")**

**{**

**listAssinaturaNivel.Add(new AssinaturaNivel() { AssinaturaNivel\_Id = assinaturaNivelAula.AssinaturaNivel\_Id, AssinaturaNivel\_Titulo = assinaturaNivelAula.AssinaturaNivel\_Titulo });**

**}**

**SelectList selectlistAssinaturaNivel = new SelectList(listAssinaturaNivel, "AssinaturaNivel\_Id", "AssinaturaNivel\_Titulo", aulaOriginal.Aula\_CodigoAssinaturaNivel);**

**ViewBag.assinaturanivel = selectlistAssinaturaNivel;**

**return View(aula);**

**}**

Implementa o método **Delete** na **AulaController**

**// GET: Admin/Aula/Delete/5**

**public ActionResult Delete(int id)**

**{**

**Aula aula = \_aulaApp.GetById(id);**

**AulaViewModel aulaViewModel = Mapper.Map<Aula, AulaViewModel>(aula);**

**return View(aulaViewModel);**

**}**

Implementa o método **Delete [HttpPost]** na **AulaController**

vamos fazer uma página de confirmação para o Deletar, vamos usar o ActionName("Delete") e [ValidateAntiForgeryToken]

**[HttpPost, ActionName("Delete")]**

**[ValidateAntiForgeryToken]**

e mudar o ActionResult para DeleteConfirmed recebendo apenas o Id

**public ActionResult DeleteConfirmed(int id)**

**// POST: Admin/Aula/Delete/5**

**[HttpPost, ActionName("Delete")]**

**[ValidateAntiForgeryToken]**

**public ActionResult DeleteConfirmed(int id)**

**{**

**Aula aula = \_aulaApp.GetById(id);**

**\_aulaApp.RemoveAula(aula);**

**return RedirectToAction("Index");**

**}**

vamos criar a view para a **Delete**

clica com direito no método View do ActionResult **Delete**(int id)

return **View**();

Add View…

View name: Delete

Template: Delete

Model class: AulaViewModel (LearningCloud.MVC.Areas.Admin.ViewModels)

Add

Na View **Delete** vamos mudar o texto de

<h2>Delete</h2><h3>Are you sure you want to delete this?</h3>

Para:

<h1>Excluir Aula <small>Tem certeza que gostaria de excluir essa aula?</small> </h1>

Vamos alterar o label do botão que está como Delete

De:

<input type="submit" value="Delete" class="btn btn-default" /> |

Para:

<input type="submit" value="Excluir" class="btn btn-default" /> **<label>|</label>**

**Vamos ainda criar também as Actions Inactivate, InactivateConfirmed e Activate**

Implementa o método **Inactivate** na **AulaController**

**// GET: Aula/Inactivate/5**

**public ActionResult Inactivate(int id)**

**{**

**Aula aula = \_aulaApp.GetById(id);**

**AulaViewModel aulaViewModel = Mapper.Map<Aula, AulaViewModel>(aula);**

**return View(aulaViewModel);**

**}**

vamos criar a view para a **Inactivate**

clica com direito no método View do ActionResult **Inactivate** (int id)

return **View**();

Add View…

View name: Inactivate

Template: Delete

Model class: AulaViewModel (LearningCloud.MVC.Areas.Admin.ViewModels)

Add

Na View **Inactivate**  vamos mudar o texto de

<h2>Inactivate</h2><h3>Are you sure you want to delete this?</h3>

Para:

<h1> Inativar Aula <small>Tem certeza que gostaria de inativar essa aula?</small> </h1>

Vamos alterar o label do botão que está como Delete

De:

<input type="submit" value="Delete" class="btn btn-default" /> |

Para:

<input type="submit" value="Inativar" class="btn btn-default" /> <label>|</label>

Implementa o método **InactivateConfirmed [HttpPost]** na **AulaController**

vamos fazer uma página de confirmação para o Inativar, vamos usar o ActionName("Inactivate") e [ValidateAntiForgeryToken]

**[HttpPost, ActionName("Inactivate")]**

**[ValidateAntiForgeryToken]**

// POST: Aula/Inactivate/5

[HttpPost, ActionName("Inactivate")]

[ValidateAntiForgeryToken]

public ActionResult InactivateConfirmed(int id)

{

Aula aula = \_aulaApp.GetById(id);

//aula.aul\_status = "I";

//\_aulaApp.UpdateAula(aula);

\_aulaApp.ChangeStatusAula(aula, "I");

return RedirectToAction("Index");

}

Faltou apenas a ActionResult **Activate**

**// GET: Aula/Activate/5**

**public ActionResult Activate(int id)**

**{**

**Aula aula = \_aulaApp.GetById(id);**

**//aula.aul\_status = "A";**

**//\_aulaApp.UpdateAula(aula);**

**\_aulaApp.ChangeStatusAula(aula, "A");**

**return RedirectToAction("Index");**

**}**

Vamos alterar a View **Index** para adicionar os botões de Ativar e Inativar podemos traduzir os labels de Edit, Details e Delete

<td>

@Html.ActionLink("**Alterar**", "Edit", new { id=item.Aula\_Id }) <label>|</label>

@Html.ActionLink("**Detalhes**", "Details", new { id=item.Aula\_Id }) <label>|</label>

@Html.ActionLink("**Excluir**", "Delete", new { id=item.Aula\_Id })

</td>

e traduzir o link de “Create New”

de:

@Html.ActionLink("**Create New**", "Create")

Para

@Html.ActionLink("**Nova Aula**", "Create")

para os botões de Ativar e Inativar, vamos criar um controle para exibir Ativar para aulas inativas e Inativar para aulas ativas.

@Html.ActionLink("Alterar", "Edit", new { id=item.Aula\_Id }) <label>|</label>

@Html.ActionLink("Detalhes", "Details", new { id=item.Aula\_Id }) <label>|</label>

@Html.ActionLink("Excluir", "Delete", new { id=item. })

**@if (item.Aula\_Status == "A")**

**{**

**<label>|</label> @Html.ActionLink("Inativar", "Inactivate", new { id = item.Aula\_Id })**

**}**

**else if (item.Aula\_Status == "I")**

**{**

**<label>|</label> @Html.ActionLink("Ativar", "Activate", new { id = item.Aula\_Id})**

**}**

os Campos **Tipo Conteúdo,Status** e **Disponível a partir** da view de Edit estão diferentes da view de Creat. Vamos copiar esses campos da View Creat para a View Edit

lembrem se que no campo **Disponível a partir** de estamos usando o DropDownList com a ViewBag.assinaturanivel que implementamos na ActionResult de Edit

<div class="form-group">

@Html.LabelFor(model => model.Aula\_Titulo, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

**<label class="radio-inline">@Html.RadioButtonFor(model => model.Aula\_TipoConteudo, "E", new { @checked = "checked" }) Aula Escrita</label>**

**<label class="radio-inline">@Html.RadioButtonFor(model => model.Aula\_TipoConteudo, "V", null) Videoaula</label>**

@Html.ValidationMessageFor(model => model.Aula\_Titulo, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.Aula\_Status, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

**<label class="radio-inline">@Html.RadioButtonFor(model => model.Aula\_Status, "A", null) Ativo</label>**

**<label class="radio-inline">@Html.RadioButtonFor(model => model.Aula\_Status, "I", null) Inativo</label>**

@Html.ValidationMessageFor(model => model.Aula\_Status, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.Aula\_CodigoAssinaturaNivel, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

**@Html.DropDownListFor(model => model.Aula\_CodigoAssinaturaNivel, (SelectList)ViewBag.assinaturanivel, String.Empty, new { @class = "form-control" })**

@Html.ValidationMessageFor(model => model.Aula\_CodigoAssinaturaNivel, "", new { @class = "text-danger" })

</div>

</div>

**Vamos alterar a action de Index para implementarmos uma paginação na lista de aulas utilizando o PagedList**

**Primeiro vamos instalar o pacote do PagedList.Mvc no projeto** LearningCloud.MVC

Instalar EntityFramework (DbContext)

* Package Manager Console
  + seleciona o *Default project* (LearningCloud.MVC)
  + Install-Package **PagedList.Mvc**

Na controller AulaController, ActionResult Index() receberemos um parâmetro inteiro chamado page

public ActionResult Index(**int? page**)

{

...

vamos tambem criar duas variaveis para configurar a paginação

**int paginaTamanho = 10;**

**int paginaNumero = (page ?? 1);**

**paginaTamanho** é a quantidade de registros por página e **paginaNumero** é o número atual da página que estamos navegando (se não tiver recebe 1).

No retorno da View vamos chamar o método .ToPagedList(paginaNumero, paginaTamanho) passando o tamanho da página e a página atual

**Ficando:**

// GET: Admin/Aula

public ActionResult Index(**int? page**)

{

**int paginaTamanho = 10;**

**int paginaNumero = (page ?? 1);**

IEnumerable<AulaViewModel> AulaViewModel = Mapper.Map<IEnumerable<Aula>, IEnumerable<AulaViewModel>>(\_aulaApp.GetByStatusAula("A,I"));

return View(listAulaViewModel**.ToPagedList(paginaNumero, paginaTamanho)**);

}

adiciona o using do **PagedList**

**using PagedList;**

na View Index vamos implementar os botões de navegação para a paginação

Alterar de

@model **IEnumerable**<LearningCloud.MVC.Areas.Admin.ViewModels.AulaViewModel>

Para

@model **IPagedList**<LearningCloud.MVC.Areas.Admin.ViewModels.AulaViewModel>

Adicione os Usings

**@using PagedList;**

**@using PagedList.Mvc;**

No final da View ou onde queira que a paginação seja exibida inclua uma linha (div)

com a implemntação do Helper Razor PagedListPager

**<div class="row" style=" text-align:center">**

**<div class="col-md-12">**

**@Html.PagedListPager(Model, page => Url.Action("Index", "Aula", new { page }))**

**</div>**

**</div>**

Como não estamos mais usando o @model **IEnumerable<>** o **DisplayNameFor** não vai mais funcionar

vamos substituir por textos simples mesmo

<tr>

<th>

@Html.DisplayNameFor(model => model.Aula\_Titulo)

</th>

<th>

@Html.DisplayNameFor(model => model.Aula\_TipoConteudo)

</th>

<th>

@Html.DisplayNameFor(model => model.Aula\_Descricao)

</th>

<th>

@Html.DisplayNameFor(model => model.aul\_prerequisitos)

</th>

<th>

@Html.DisplayNameFor(model => model.Aula\_Prerequisitos)

</th>

<th>

@Html.DisplayNameFor(model => model.Aula\_Imagem)

</th>

<th>

@Html.DisplayNameFor(model => model.Aula\_TempoVideo)

</th>

<th>

@Html.DisplayNameFor(model => model.Aula\_Video)

</th>

<th>

@Html.DisplayNameFor(model => model.Aula\_ConteudoEscrito)

</th>

<th>

@Html.DisplayNameFor(model => model.Aula\_Status)

</th>

<th>

@Html.DisplayNameFor(model => model.Aula\_CodigoAssinaturaNivel)

</th>

<th></th>

</tr>

para:

<tr>

<th>Título</th>

<th>Tipo Conteúdo</th>

<th>Descrição</th>

<th>Pré-requisitos</th>

<th>Imagem</th>

<th>Tempo do Vídeo</th>

<th>Vídeo</th>

<th>Conteúdo da Aula</th>

<th>Status</th>

<th>Disponível a partir de</th>

<th></th>

</tr>

Vamos alterar o visual da tabela que lista as aulas, deixando a listrada

de

<table class="table">

para

<table class="table table-condensed table-striped">

Vamos alterar o valor que é exibido na coluna “Disponível a partir de” que hoje exibe o Fk do relacionamento com a tabela AssinaturaNivel, vamos exibir o título da assinatura

de

@Html.DisplayFor(modelItem => item.**Aula\_CodigoAssinaturaNivel**)

para

@Html.DisplayFor(modelItem => item**.Aula\_AssinaturaNivel.AssinaturaNivel\_Titulo)**

vamos alterar o valor que é exibido na coluna “Tipo Conteúdo” que hoje exibe o caracter V ou A, vamos controlar para exibir Videoaula ou Aula Escrita

alterar de:

<td>

@Html.DisplayFor(modelItem => item.Aula\_TipoConteudo)

</td>

Para:

<td>

@if (item.Aula\_TipoConteudo == "V")

{

<span>Videoaula </span>

}

else if (item.Aula\_TipoConteudo == "E")

{

<span>Aula Escrita</span>

}

</td>

ainda na View Index vamos alterar o valor que é exibido na coluna “Status” que hoje exibe o caracter I ou A, vamos controlar para exibir Inativo ou Ativo

alterar de:

<td>

@Html.DisplayFor(modelItem => item.Aula\_Status)

</td>

Para:

<td>

@if (item.Aula\_Status == "A")

{

<span>Ativo</span>

}

else if (item.Aula\_Status == "I")

{

<span>Inativo</span>

}

</td>

vamos criar um controle para que a aulas inativas apareçam destacadas na lista de Aulas

Para isso vamos criar uma variável do tipo string na view Index

@{

ViewBag.Title = "Index"; ⇐ Vamos alterar esse title também para ViewBag.Title = "**Consultar Aulas**";

**string classeCssInativo = "";**

}

ficando

@{

ViewBag.Title = "Consultar Aulas";

**string classeCssInativo = "";**

}

no Loop que lista as aulas fazer o seguinte controle

@foreach (var item in Model)

{

**classeCssInativo = "";**

**if (item.aul\_status == "I")**

**{**

**classeCssInativo = "warning";**

**}**

<tr **class="@classeCssInativo"**>

<td>

@Html.DisplayFor(modelItem => item.Aula\_Titulo)

</td>

...

Vamos fazer os controles para exibir o título do nível da assinatura e a descrição do status e tipo de conteúdo nas views **Datails , Delete e Inactive**

na view **Datails** alterando o DisplayFor de:

<dt>

@Html.DisplayNameFor(model => model.Aula\_TipoConteudo)

</dt>

<dd>

**@Html.DisplayFor(model => model.Aula\_TipoConteudo)**

</dd>

<dt>

@Html.DisplayNameFor(model => model.Aula\_Status)

</dt>

<dd>

**@Html.DisplayFor(model => model.Aula\_Status)**

</dd>

<dt>

@Html.DisplayNameFor(model => model.Aula\_CodigoAssinaturaNivel)

</dt>

<dd>

@Html.DisplayFor(model => model.**Aula\_CodigoAssinaturaNivel**)

</dd>

Para:

<dt>

@Html.DisplayNameFor(model => model.Aula\_TipoConteudo)

</dt>

<dd>

**@if (Model.Aula\_TipoConteudo == "V")**

**{**

**<span>Videoaula</span>**

**}**

**else if (Model.Aula\_TipoConteudo == "E")**

**{**

**<span>Aula Escrita</span>**

**}**

</dd>

<dt>

@Html.DisplayNameFor(model => model.Aula\_Status)

</dt>

<dd>

**@if (Model.Aula\_Status == "A")**

**{**

**<span>Ativo</span>**

**}**

**else if (Model.Aula\_Status == "I")**

**{**

**<span>Inativo</span>**

**}**

</dd>

<dt>

@Html.DisplayNameFor(model => model.Aula\_CodigoAssinaturaNivel)

</dt>

<dd>

@Html.DisplayFor(model => model.**Aula\_AssinaturaNivel.AssinaturaNivel\_Titulo**)

</dd>

Fazer as mesmas alterações nas views **Delete e Inactive**

Vamos diminuir a quantidade de registros por página na paginação

Na A**ulaController** alterar a quantidade da variável **paginaTamanho**

public ActionResult Index(int? page)

{

int **paginaTamanho = 2**; ⇐ de 10 para 2 apenas para testar com poucos registros

int paginaNumero = (page ?? 1);

**…**

Após cadastrarmos algumas aulas e abrir a página que lista as aulas temos os botões de paginação



Repare que a URL exibida tem um parametro page={n}



Vamos melhorar esse exibição da URL para um formato mais amigável, decorando nosso controller com um atributo chamado Route.

Primeiro vamos alterar o arquivo RouteConfig.cs para adicionar uma extensão de método do parâmetro “routes” do RouteCollection chamado **MapMvcAttributeRoutes()**

**routes.MapMvcAttributeRoutes();**

**Ficando:**

using System.Web.Mvc;

using System.Web.Routing;

namespace LearningCloud.MVC

{

public class RouteConfig

{

public static void RegisterRoutes(RouteCollection routes)

{

routes.IgnoreRoute("{resource}.axd/{\*pathInfo}");

**routes.MapMvcAttributeRoutes();**

routes.MapRoute(

name: "Default",

url: "{controller}/{action}/{id}",

defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional },

namespaces: new[] { "LearningCloud.MVC.Controllers" }

);

}

}

}

No arquivo Global.asax.cs existe a chamada do método “**AreaRegistration.RegisterAllAreas()**” dentro de Application\_Start. Vamos recortar esse “AreaRegistration.RegisterAllAreas()” e colar dentro do arquivo RouteConfig.cs

Recortar de **Global.asax.cs**

public class MvcApplication : System.Web.HttpApplication

{

protected void Application\_Start()

{

**AreaRegistration.RegisterAllAreas();**

FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);

RouteConfig.RegisterRoutes(RouteTable.Routes);

BundleConfig.RegisterBundles(BundleTable.Bundles);

AutoMapperConfig.RegisterMappings();

}

}

Ficando

public class MvcApplication : System.Web.HttpApplication

{

protected void Application\_Start()

{

FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);

RouteConfig.RegisterRoutes(RouteTable.Routes);

BundleConfig.RegisterBundles(BundleTable.Bundles);

AutoMapperConfig.RegisterMappings();

}

}

E Colar em **RouteConfig.cs** ficando:

public class RouteConfig

{

public static void RegisterRoutes(RouteCollection routes)

{

routes.IgnoreRoute("{resource}.axd/{\*pathInfo}");

routes.MapMvcAttributeRoutes();

**AreaRegistration.RegisterAllAreas();**

routes.MapRoute(

name: "Default",

url: "{controller}/{action}/{id}",

defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional },

namespaces: new[] { "NewLearningCloud.MVC.Controllers" }

);

}

}

No controller AulaController vamos decorar a classe com os atributos

**[RouteArea("Admin")]**

**[RoutePrefix("Aula")]**

Ficando

...

namespace NewLearningCloud.MVC.Areas.Admin.Controllers

{

**[RouteArea("Admin")]**

**[RoutePrefix("Aula")]**

public class AulaController : Controller

{

private readonly IAulaAppService \_aulaApp;

private readonly IAssinaturaNivelAppService \_assinaturaNivelApp;

...

E vamos decorar o metodo ActionResult Index com

**[Route("{page?}")]**

Ficando

...

// GET: Admin/Aula

**[Route("{page?}")]**

public ActionResult Index(int? page)

{

int paginaTamanho = 2;

int paginaNumero = (page ?? 1);

IEnumerable<AulaViewModel> listAulaViewModel = Mapper.Map<IEnumerable<Aula>, IEnumerable<AulaViewModel>>(\_aulaApp.GetByStatusAula("A,I"));

return View(listAulaViewModel.ToPagedList(paginaNumero, paginaTamanho));

}

...

Temos também que decorar os métodos ActionResult Create com

**[Route("Criar")]**

Ficando:

// GET: Admin/Aula/Create

**[Route("Criar")]**

public ActionResult Create()

{

ViewBag.assinaturanivel = new SelectList(\_assinaturaNivelApp.GetByStatusAssinaturaNivel("A"), "AssinaturaNivel\_Id", "AssinaturaNivel\_Titulo");

return View();

}

// POST: Admin/Aula/Create

[HttpPost]

**[Route("Criar")]**

[ValidateAntiForgeryToken]

public ActionResult Create(AulaViewModel aula)

...

Note que a partir de agora o número da página é exibida na URL após a /(barra)



E não com ”?page=2”



**criar nova pasta no projeto MVC chamada HtmlHelpers**

Criar Pastas para organizar o Data

* 0 - Presentation
  + LearningCloud.MVC (Clica com Direito)
  + Add - New Folder
  + HtmlHelpers

**classe static StringHelpers**

using System.Globalization;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;

namespace NewLearningCloud.MVC.HtmlHelpers

{

public static class StringHelpers

{

public static string ToSeoUrl(this string url)

{

// make the url lowercase

string encodedUrl = (url ?? "").ToLower();

// replace & with and

encodedUrl = Regex.Replace(encodedUrl, @"\&+", "e");

// remove characters

encodedUrl = encodedUrl.Replace("'", "");

// remove acentos

encodedUrl = StringHelpers.RemoverAcentuacao(encodedUrl);

// remove invalid characters

encodedUrl = Regex.Replace(encodedUrl, @"[^a-z0-9]", "-");

// remove duplicates

encodedUrl = Regex.Replace(encodedUrl, @"-+", "-");

// trim leading & trailing characters

encodedUrl = encodedUrl.Trim('-');

return encodedUrl;

}

public static string RemoverAcentuacao(this string text)

{

return new string(text

.Normalize(NormalizationForm.FormD)

.Where(ch => char.GetUnicodeCategory(ch) != UnicodeCategory.NonSpacingMark)

.ToArray());

}

public static string UpperCaseFirst(this string title)

{

// Check for empty string.

if (string.IsNullOrEmpty(title))

{

return string.Empty;

}

// Return char and concat substring.

return char.ToUpper(title[0]) + title.Substring(1);

}

}

}

**index.cshtml**

Incluir o using

**@using NewLearningCloud.MVC.HtmlHelpers;**

@Html.ActionLink("Detalhes", "Details", new { id = item.Aula\_Id**, titulo = item.Aula\_Titulo.ToSeoUrl()** }) <label>|</label>

No Controler aula (AulaController)

**[Route("Detalhes/{id}/{titulo}")]**

public ActionResult Details(int id, string titulo)

{

…

**if (aula.Aula\_Titulo.ToSeoUrl() != titulo)**

**{**

**return RedirectToAction("Details", new { id, titulo = aula.Aula\_Titulo.ToSeoUrl() });**

**//return HttpNotFound("A aula que você está procurando não foi encontrada!");**

**}**

**…**

Temos também que decorar os métodos ActionResult Edit com

**[Route("Alterar/{id}")]**

Ficando:

**[Route("Alterar/{id}")]**

public ActionResult Edit(int id)

{

**…**

[HttpPost]

**[Route("Alterar/{id}")]**

public ActionResult Edit(AulaViewModel aula)

{

if (ModelState.IsValid)

...

Temos também que decorar os métodos ActionResult Delete com

**[Route("Excluir/{id}")]**

Ficando:

// GET: Admin/Aula/Delete/5

**[Route("Excluir/{id}")]**

public ActionResult Delete(int id)

{

...

// POST: Admin/Aula/Delete/5

[HttpPost, ActionName("Delete")]

**[Route("Excluir/{id}")]**

[ValidateAntiForgeryToken]

public ActionResult DeleteConfirmed(int id)

{

...

Temos também que decorar os métodos ActionResult Inactivate com

**[Route("Inativar/{id}")]**

Ficando:

// GET: Aula/Inactivate/5

**[Route("Inativar/{id}")]**

public ActionResult Inactivate(int id)

{

...

// POST: Aula/Inactivate/5

[HttpPost, ActionName("Inactivate")]

**[Route("Inativar/{id}")]**

[ValidateAntiForgeryToken]

public ActionResult InactivateConfirmed(int id)

{

...

Temos também que decorar os métodos ActionResult Activate com

**[Route("Ativar/{id}")]**

Ficando:

// GET: Aula/Activate/5

**[Route("Ativar/{id}")]**

public ActionResult Activate(int id)

{

Tudo funcionando, note que qualquer alteração sempre volta para a página 1 da paginação. Vamos melhorar isso para sempre voltar na página que disparou a ação. Para isso vamos passar o número da página que chamou a solicitação

Na View Index primeiro vamos alterar os links de **Edit**, **Details**, **Delete**, **Inactivate** e **Activate** para passar a página atual para seu respectivo actions

Alterar

<td>

@Html.ActionLink("Alterar", "Edit", new { id = item.Aula\_Id }) <label>|</label>

@Html.ActionLink("Detalhes", "Details", new { id = item.Aula\_Id, titulo = item.Aula\_Titulo.ToSeoUrl() }) <label>|</label>

@Html.ActionLink("Deletar", "Delete", new { id = item.Aula\_Id })

@if (item.Aula\_Status == "A")

{

<label>|</label> @Html.ActionLink("Inativar", "Inactivate", new { id = item.Aula\_Id })

}

else if (item.Aula\_Status == "I")

{

<label>|</label> @Html.ActionLink("Ativar", "Activate", new { id = item.Aula\_Id })

}

</td>

Para

<td>

@Html.ActionLink("Alterar", "Edit", new { id = item.Aula\_Id**, page = Model.PageNumber** }) **<label>|</label>**

@Html.ActionLink("Detalhes", "Details", new { id = item.Aula\_Id**, page = Model.PageNumber**, titulo = item.Aula\_Titulo.ToSeoUrl() }) **<label>|</label>**

@Html.ActionLink("Deletar", "Delete", new { id = item.aulAula\_Id\_id**, page = Model.PageNumber** })

@if (item.Aula\_Status == "A")

{

<label>|</label> @Html.ActionLink("Inativar", "Inactivate", new { id = item.Aula\_Id**, page = Model.PageNumber** })

}

else if (item.Aula\_Status == "I")

{

<label>|</label> @Html.ActionLink("Ativar", "Activate", new { id = item.Aula\_Id**, page = Model.PageNumber** })

}

</td>

Como recebemos no ViewModel como IPagedList<> com isso podemos usar o **PageNumber.**

Agora podemos alterar as actions da AulaController para receber o número da página. Vamos começar com a **Details**

**Na ActionResult Details da AulaController** vamos acrescentar um parâmetro int chamado page, e uma ViewBag também chamada page

// GET: Admin/Aula/Details/5

public ActionResult Details(int id, **int? page**, string titulo)

{

Aula aula = \_aulaApp.GetById(id);

AulaViewModel aulaViewModel = Mapper.Map<Aula, AulaViewModel>(aula);

**ViewBag.page = page;**

if (aula.Aula\_Titulo.ToSeoUrl() != titulo)

{

return RedirectToAction("Details", new { id, titulo = aula.Aula\_Titulo.ToSeoUrl() });

//return HttpNotFound("A aula que você está procurando não foi encontrada!");

}

return View(aulaViewModel);

}

Na View **Details** vamos alterar o botão voltar para retornar a página que o action Index espera (Aproveitar para traduzir os label dos links e Title)

<p>

@Html.ActionLink("**Edit**", "Edit", new { id = Model.Aula\_Id }) |

**@Html.ActionLink("Back to List", "Index")**

</p>

alterar para

<p>

@Html.ActionLink("**Alterar**", "Edit", new { id = Model.Aula\_Id }) |

@Html.ActionLink("**Voltar**", "Index", **new { page = ViewBag.page }**)

</p>

Recebemos o número da página atraves da **ViewBag.page** que implementamos no Action **Details**

Vamos alterar agora a **Create**

**Na ActionResult Create da AulaController** também vamos acrescentar um parâmetro int chamado page, e uma ViewBag também chamada page

// GET: Admin/Aula/Create

public ActionResult Create(**int? page**)

{

ViewBag.assinaturanivel = new SelectList(\_assinaturaNivelApp.GetByStatusAssinaturaNivel("A"), "AssinaturaNivel\_Id", "AssinaturaNivel\_Titulo");

**ViewBag.page = page;**

return View();

}

Na View Create vamos alterar o botão voltar para retornar o número da página para a Index

de  
 @Html.ActionLink("Back to List", "Index")

Para

@Html.ActionLink("**Voltar**", "Index"**, new { page = ViewBag.page }**)

Vamos alterar a ActionResult Create que recebe o Post [HttpPost], pois podemos deixar que após a criação exiba a primeira página mesmo. Não vamos garantir que o registro criado fique na mesma página que o usuário estava. MAs caso ocorra algum erro vamos usar a página no botão voltar.

// POST: Admin/Aula/Create

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Create(AulaViewModel aula, **int? page**)

{

if (ModelState.IsValid)

{

Aula aulaDomain = Mapper.Map<AulaViewModel, Aula>(aula);

\_aulaApp.CreateAula(aulaDomain);

return RedirectToAction("Index");

}

ViewBag.assinaturanivel = new SelectList(\_assinaturaNivelApp.GetByStatusAssinaturaNivel("A"), "AssinaturaNivel\_Id", "AssinaturaNivel\_Titulo", aula.Aula\_CodigoAssinaturaNivel);

**ViewBag.page = page;**

return View(aula);

}

Vamos alterar agora a **Edit**

**Na ActionResult Edit da AulaController** também vamos acrescentar um parâmetro int chamado page, e uma ViewBag também chamada page

// GET: Admin/Aula/Edit/5

public ActionResult Edit(int id**, int? page**)

{

Aula aula = \_aulaApp.GetById(id);

AssinaturaNivel assinaturaNivelAula = \_assinaturaNivelApp.GetById(aula.Aula\_CodigoAssinaturaNivel);

AulaViewModel aulaViewModel = Mapper.Map<Aula, AulaViewModel>(aula);

List<AssinaturaNivel> listAssinaturaNivel = new List<AssinaturaNivel>(\_assinaturaNivelApp.GetByStatusAssinaturaNivel("A"));

if (assinaturaNivelAula.AssinaturaNivel\_Status == "I")

{

listAssinaturaNivel.Add(new AssinaturaNivel() { AssinaturaNivel\_Id = assinaturaNivelAula.AssinaturaNivel\_Id, AssinaturaNivel\_Titulo = assinaturaNivelAula.AssinaturaNivel\_Titulo });

}

SelectList selectlistAssinaturaNivel = new SelectList(listAssinaturaNivel, "AssinaturaNivel\_Id", "AssinaturaNivel\_Titulo", aula.Aula\_CodigoAssinaturaNivel);

ViewBag.assinaturanivel = selectlistAssinaturaNivel;

**ViewBag.page = page;**

return View(aulaViewModel);

}

Na View **Edit** vamos alterar o botão voltar para retornar o número da página para a Index

de  
 @Html.ActionLink("Back to List", "Index")

Para

@Html.ActionLink("**Voltar**", "Index"**, new { page = ViewBag.page }**)

Ainda na View **Edit**

vamos alterar também o parâmetro do Helper que cria o formulário HTML

**de**

@using (Html.BeginForm())

{

**...**

**para**

@using (Html.BeginForm(**new { page = ViewBag.page }**))

{

**...**

Com Isso vamos passar o número da página para a action **Edit que recebe o Post.** Agora vamos alterar a action **Edit** que recebe o Post [HttpPost], também devemos acrescentar um parâmetro int chamado page, como essa action retorna para a Index que tem o parâmetro de entrada de page, não vamos criar a ViewBag.page, temos apenas que incluir o page no **return RedirectToAction("Index");**

Ficando:

// POST: Admin/Aula/Edit/5

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Edit(AulaViewModel aula**, int? page**)

{

if (ModelState.IsValid)

{

Aula aulaDomain = Mapper.Map<AulaViewModel, Aula>(aula);

\_aulaApp.UpdateAula(aulaDomain);

return RedirectToAction("Index"**, new {page = page }**);

}

Aula aulaOriginal = \_aulaApp.GetById(aula.Aula\_Id);

AssinaturaNivel assinaturaNivelAula = \_assinaturaNivelApp.GetById(aulaOriginal.Aula\_CodigoAssinaturaNivel);

List<AssinaturaNivel> listAssinaturaNivel = new List<AssinaturaNivel>(\_assinaturaNivelApp.GetByStatusAssinaturaNivel("A"));

if (assinaturaNivelAula.AssinaturaNivel\_Status == "I")

{

listAssinaturaNivel.Add(new AssinaturaNivel() { AssinaturaNivel\_Id = assinaturaNivelAula.AssinaturaNivel\_Id, AssinaturaNivel\_Titulo = assinaturaNivelAula.AssinaturaNivel\_Titulo });

}

SelectList selectlistAssinaturaNivel = new SelectList(listAssinaturaNivel, "AssinaturaNivel\_Id", "AssinaturaNivel\_Titulo", aulaOriginal.Aula\_CodigoAssinaturaNivel);

ViewBag.assinaturanivel = selectlistAssinaturaNivel;

**ViewBag.page = page;**

return View(aula);

}

Na ActionResult **Inactivate** da **AulaController** também vamos acrescentar um parâmetro int chamado page, e uma ViewBag também chamada page

ficando

// GET: Aula/Inactivate/5

public ActionResult Inactivate(int id**, int? page**)

{

Aula aula = \_aulaApp.GetById(id);

AulaViewModel aulaViewModel = Mapper.Map<Aula, AulaViewModel>(aula);

**ViewBag.page = page;**

return View(aulaViewModel);

}

Na View **Inactivate** vamos alterar o botão voltar para retornar o número da página para a Index e também o parâmetro do Helper que cria o formulário HTML

**de**

@using (Html.BeginForm())

{

@Html.AntiForgeryToken()

<div class="form-actions no-color">

<input type="submit" value="Inativar" class="btn btn-default" /> |

@Html.ActionLink("Back to List", "Index")

</div>

}

**Para**

@using (Html.BeginForm(**new { page = ViewBag.page }**))

{

@Html.AntiForgeryToken()

<div class="form-actions no-color">

<input type="submit" value="Inativar" class="btn btn-default" /> <label>|</label>

@Html.ActionLink(**"Voltar"**, "Index"**, new { page = ViewBag.page }**)

</div>

}

Com Isso vamos passar o número da página para a action **InactivateConfirmed que recebe o Post.** Agora vamos alterar a action **InactivateConfirmed** que recebe o Post [HttpPost], também devemos acrescentar um parâmetro int chamado page, como essa action retorna para a Index que tem o parâmetro de entrada de page, não vamos criar a ViewBag.page, temos apenas que incluir o page no **return RedirectToAction("Index");**

Ficando:

// POST: Aula/Inactivate/5

[HttpPost, ActionName("Inactivate")]

[ValidateAntiForgeryToken]

public ActionResult InactivateConfirmed(int id**, int? page**)

{

Aula aula = \_aulaApp.GetById(id);

\_aulaApp.ChangeStatusAula(aula, "I");

return RedirectToAction("Index"**, new { page = page }**);

}

Na ActionResult **Activate** da **AulaController** também vamos acrescentar um parâmetro int chamado page e incluir o page no **return RedirectToAction("Index");**

ficando

// GET: Aula/Activate/5

public ActionResult Activate(int id**, int? page**)

{

Aula aula = \_aulaApp.GetById(id);

\_aulaApp.ChangeStatusAula(aula, "A");

return RedirectToAction("Index"**, new { page = page }**);

}

Na ActionResult **Delete** da **AulaController** também vamos acrescentar um parâmetro int chamado page, e uma ViewBag também chamada page

ficando

// GET: Admin/Aula/Delete/5

public ActionResult Delete(int id**, int? page**)

{

Aula aula = \_aulaApp.GetById(id);

AulaViewModel aulaViewModel = Mapper.Map<Aula, AulaViewModel>(aula);

**ViewBag.page = page;**

return View(aulaViewModel);

}

Na View **Delete** vamos alterar o botão voltar para retornar o número da página para a Index

**de**

@Html.ActionLink("Back to List", "Index")

**Para**

@Html.ActionLink("**Voltar**", "Index"**, new { page = ViewBag.page }**)

Não vamos alterar a ActionResult DeleteConfirmed que recebe o Post [HttpPost], pois como o registrto será excluido o numero de páginas poderá ser diferente

Na Action Edit ainda temos um problema. Ela está sendo chamada em dois lugares diferentes, na lista da index e no Detalhe mas o voltar não tem a informação da action que deve retornar

Vamos passar para a action Edit qual action que chamou a ação, além do número da página que já passamos

Na ActionResult Edit vamos criar um parâmetro “returnaction”, que vai receber action que deverá redirecionar após a alteração do registro

// GET: Admin/Aula/Edit/5

public ActionResult Edit(int id, int? page**, string returnaction**)

Vamos criar também um controle no caso do parâmetro vier vazio ou nulo informamos que a returnaction será para Index

**if (returnaction == "" || returnaction == null)**

**{**

**returnaction = "Index";**

**}**

e precisamos criar uma ViewBAg para passar essa informação para a View Edit

**ViewBag.ReturnAction = returnaction;**

Ficando:

// GET: Admin/Aula/Edit/5

public ActionResult Edit(int id, int? page**, string returnaction**)

{

Aula aula = \_aulaApp.GetById(id);

AssinaturaNivel assinaturaNivelAula = \_assinaturaNivelApp.GetById(aula.Aula\_CodigoAssinaturaNivel);

AulaViewModel aulaViewModel = Mapper.Map<Aula, AulaViewModel>(aula);

List<AssinaturaNivel> listAssinaturaNivel = new List<AssinaturaNivel>(\_assinaturaNivelApp.GetByStatusAssinaturaNivel("A"));

//SelectList listAssinaturaNivel = new SelectList(\_assinaturaNivelApp.GetByStatusAssinaturaNivel("A"), "AssinaturaNivel\_Id", "AssinaturaNivel\_Titulo", aula.Aula\_CodigoAssinaturaNivel);

if (assinaturaNivelAula.AssinaturaNivel\_Status == "I")

{

listAssinaturaNivel.Add(new AssinaturaNivel() { AssinaturaNivel\_Id = assinaturaNivelAula.AssinaturaNivel\_Id, AssinaturaNivel\_Titulo = assinaturaNivelAula.AssinaturaNivel\_Titulo });

}

SelectList selectlistAssinaturaNivel = new SelectList(listAssinaturaNivel, "AssinaturaNivel\_Id", "AssinaturaNivel\_Titulo", aula.Aula\_CodigoAssinaturaNivel);

**if (returnaction == "" || returnaction == null)**

**{**

**returnaction = "Index";**

**}**

ViewBag.assinaturanivel = selectlistAssinaturaNivel;

**ViewBag.ReturnAction = returnaction;**

ViewBag.page = page;

return View(aulaViewModel);

}

Na Action Edit que recebe o post [HttpPost] vamos criar o parametro do **returnaction** e no redirect passar o valor desse parametro

Ficando

// POST: Admin/Aula/Edit/5

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Edit(AulaViewModel aula, int? page**, string returnaction**)

{

if (ModelState.IsValid)

{

var aulaDomain = Mapper.Map<AulaViewModel, Aula>(aula);

\_aulaApp.UpdateAula(aulaDomain);

return RedirectToAction(**(string)returnaction,** new { page = page });

}

Aula aulaOriginal = \_aulaApp.GetById(aula.Aula\_Id);

AssinaturaNivel assinaturaNivelAula = \_assinaturaNivelApp.GetById(aulaOriginal.Aula\_CodigoAssinaturaNivel);

List<AssinaturaNivel> listAssinaturaNivel = new List<AssinaturaNivel>(\_assinaturaNivelApp.GetByStatusAssinaturaNivel("A"));

if (assinaturaNivelAula.AssinaturaNivel\_Status == "I")

{

listAssinaturaNivel.Add(new AssinaturaNivel() { AssinaturaNivel\_Id = assinaturaNivelAula.AssinaturaNivel\_Id, AssinaturaNivel\_Titulo = assinaturaNivelAula.AssinaturaNivel\_Titulo });

}

SelectList selectlistAssinaturaNivel = new SelectList(listAssinaturaNivel, "AssinaturaNivel\_Id", "AssinaturaNivel\_Titulo", aulaOriginal.Aula\_CodigoAssinaturaNivel);

ViewBag.assinaturanivel = selectlistAssinaturaNivel;

**ViewBag.ReturnAction = returnaction;**

ViewBag.page = page;

return View(aula);

}

na View Edit vamos alterar o action formulario para passar o valor da ViewBag returnaction

De

@using (Html.BeginForm(new { page = ViewBag.page }))

{

...

Para

@using (Html.BeginForm(new { **returnaction = ViewBag.ReturnAction**, page = ViewBag.page }))

{

...

Vamos alterar também o botão voltar para receber a action que deve voltar dinamicamente

de

@Html.ActionLink("Voltar", "Index", new { page = ViewBag.page })

Para

@Html.ActionLink("Voltar", **(string)ViewBag.ReturnAction,** new { page = ViewBag.page })

Na View Details vamos alterar o link que chama a Edit para passarmos o valor da returnaction

**de**

@Html.ActionLink("Alterar", "Edit", new { id = Model.Aula\_Id }) |

**Para**

@Html.ActionLink("Alterar", "Edit", new { id = Model.Aula\_Id**, page = ViewBag.page, returnaction = "Detalhes/" + Model.Aula\_Id** }) <label>|</label>

**Na** ActionResult Indexda **AulaController** também vamos acrescentar uma ViewBag também chamada page

ficando

public ActionResult Index(int? page)

{

int paginaTamanho = 2;

int paginaNumero = (page ?? 1);

IEnumerable<AulaViewModel> listAulaViewModel = Mapper.Map<IEnumerable<Aula>, IEnumerable<AulaViewModel>>(\_aulaApp.GetAll());

**ViewBag.page = page;**

return View(listAulaViewModel.ToPagedList(paginaNumero, paginaTamanho));

}

**essa ViewBag.page servirá para passar para a ViewIndex a página para ser usada no botão que cria novas aulas**

Na View **Index** vamos alterar o link que chama a **Create** para passarmos o número da página

**View Index**

**de**

**@Html.ActionLink("Create New", "Create")**

**para**

**@Html.ActionLink("Nova Aula", "Create", new { page = ViewBag.page })**

// GET: Admin/Aula/Details/5

[Route("Detalhes/{id}/{titulo**?**}**/{page?}**")]

//[Route("Detalhes/{id}")]

public ActionResult Details(int id, int? page, string titulo)

// GET: Admin/Aula/Create

[Route("Criar**/{page?}**")]

public ActionResult Create(int? page)

// POST: Admin/Aula/Create

[HttpPost]

[Route("Criar**/{page?}**")]

[ValidateAntiForgeryToken]

public ActionResult Create(AulaViewModel aula, int? page)

// GET: Admin/Aula/Edit/5

[Route("Alterar/{id}**/{page?}**")]

public ActionResult Edit(int id, int? page, string returnaction)

// POST: Admin/Aula/Edit/5

[HttpPost]

[Route("Alterar/{id}**/{page?}**")]

public ActionResult Edit(AulaViewModel aula, int? page, string returnaction)

// GET: Admin/Aula/Delete/5

[Route("Excluir/{id}**/{page?}**")]

public ActionResult Delete(int id, int? page)

// POST: Admin/Aula/Delete/5

[HttpPost, ActionName("Delete")]

[Route("Excluir/{id}**/{page?}**")]

[ValidateAntiForgeryToken]

public ActionResult DeleteConfirmed(int id)

// GET: Aula/Inactivate/5

[Route("Inativar/{id}**/{page?}**")]

public ActionResult Inactivate(int id, int? page)

// POST: Aula/Inactivate/5

[HttpPost, ActionName("Inactivate")]

[Route("Inativar/{id}**/{page?}**")]

[ValidateAntiForgeryToken]

public ActionResult InactivateConfirmed(int id, int? page)

Identity

LearningCloud.Infra.CrossCutting.Identity

Adiciona o projeto de Identity em CrossCutting em Infra - CrossCutting(3 - Infra - 3.2 - CrossCutting)

* 3 - Infra - 3.1 - CrossCutting
  + Add - New Project
  + Visual C# - Class Library
  + LearningCloud.Infra.CrossCutting.Identity
  + Deleta a Classe “Class1.cs”

Criar Pastas para organizar o Identity

* LearningCloud.Infra.CrossCutting.Identity (Clica com Direito)
  + Add - New Folder
    - Configuration
  + Add - New Folder
    - ContextIdentity
  + Add - New Folder
    - ViewModels

Criar Pastas para organizar as ViewModels dentro de CrossCutting.Identity

* Models(Clica com Direito)
  + Add - New Folder
    - AccountViewModels
  + Add - New Folder
    - ManageViewModels

Vamos instalar o *EntityFramework*, *Owin, AspNet.Identity.Owin*, *AspNet.Identity.Core* e *AspNet.Identity.EntityFramework*

* Package Manager Console
  + seleciona o *Default project* (LearningCloud.Infra.CrossCutting.**Identity**)
  + Install-Package EntityFramework
  + Install-Package Microsoft.Owin
  + Install-Package Microsoft.AspNet.Identity.Owin
  + Install-Package Microsoft.AspNet.Identity.Core
  + Install-Package Microsoft.AspNet.Identity.EntityFramework

Vamos criar uma classe para o ApplicationUser na pasta ViewModels do projeto CrossCutting.**Identity**

Adiciona a classe ApplicationUser na camada de Infra.CrossCutting.**Identity** na pasta Models

* 3 - Infra / 3.2 - CrossCutting
* Projeto LearningCloud.Infra.CrossCutting.**Identity**
* Pasta ViewModels (Clica com Direito)
  + Add - Class
  + ApplicationUser.cs
  + definir como public
  + Herdar *: IdentityUser*

Adiciona o código da classe ApplicationUser da Models/IdentityModels.cs ficando:

using System.Security.Claims;

using System.Threading.Tasks;

using Microsoft.AspNet.Identity;

using Microsoft.AspNet.Identity.EntityFramework;

namespace LearningCloud.Infra.CrossCutting.Identity.Models

{

public class ApplicationUser : IdentityUser

{

**public async Task<ClaimsIdentity> GenerateUserIdentityAsync(UserManager<ApplicationUser> manager)**

**{**

**// Note the authenticationType must match the one defined in CookieAuthenticationOptions.AuthenticationType**

**var userIdentity = await manager.CreateIdentityAsync(this, DefaultAuthenticationTypes.ApplicationCookie);**

**// Add custom user claims here**

**return userIdentity;**

**}**

}

}

Vamos separar os métodos de configuração que o projeto padrão do MVC com Identity cria na pasta App\_Start da classe IdentityConfig.cs. Cada um desses métodos ficará em uma classe separada na pasta Configuration no projeto do Identity na camada de CrossCutting

assim teremos as seguintes classes

EmailService

SmsService

ApplicationUserManager

ApplicationSignInManager

Adiciona a classe EmailService na camada de Infra.CrossCutting.**Identity** na pasta Configuration

* 3 - Infra / 3.2 - CrossCutting
* Projeto LearningCloud.Infra.CrossCutting.**Identity**
* Pasta Configuration (Clica com Direito)
  + Add - Class
  + EmailService.cs
  + definir como public
  + Herdar *IIdentityMessageService*

Adiciona o código do metodo EmailService da IdentityConfig ficando:

using System.Threading.Tasks;

using Microsoft.AspNet.Identity;

namespace LearningCloud.Infra.CrossCutting.Identity.Configuration

{

public class EmailService : IIdentityMessageService

{

**public Task SendAsync(IdentityMessage message)**

**{**

**// Plug in your email service here to send an email.**

**return Task.FromResult(0);**

**}**

}

}

Adiciona a classe SmsService na camada de Infra.CrossCutting.**Identity** na pasta Configuration

* 3 - Infra / 3.2 - CrossCutting
* Projeto LearningCloud.Infra.CrossCutting.**Identity**
* Pasta Configuration (Clica com Direito)
  + Add - Class
  + SmsService.cs
  + definir como public
  + Herdar *IIdentityMessageService*

Adiciona o código do método SmsService da IdentityConfig ficando:

using System.Threading.Tasks;

using Microsoft.AspNet.Identity;

namespace LearningCloud.Infra.CrossCutting.Identity.Configuration

{

public class SmsService : IIdentityMessageService

{

**public Task SendAsync(IdentityMessage message)**

**{**

**// Plug in your SMS service here to send a text message.**

**return Task.FromResult(0);**

**}**

}

}

Adiciona a classe ApplicationUserManager na camada de Infra.CrossCutting.**Identity** na pasta Configuration

* 3 - Infra / 3.2 - CrossCutting
* Projeto LearningCloud.Infra.CrossCutting.**Identity**
* Pasta Configuration (Clica com Direito)
  + Add - Class
  + ApplicationUserManager.cs
  + definir como public
  + Herdar  *: UserManager<ApplicationUser>*

Adiciona o código do método ApplicationUserManager da IdentityConfig ficando:

using System;

using Microsoft.AspNet.Identity;

using Microsoft.AspNet.Identity.Owin;

using Microsoft.Owin.Security.DataProtection;

using LearningCloud.Infra.CrossCutting.Identity.Models;

namespace LearningCloud.Infra.CrossCutting.Identity.Configuration

{

public class ApplicationUserManager : UserManager<ApplicationUser>

{

**public ApplicationUserManager(IUserStore<ApplicationUser> store)**

**: base(store)**

**{**

**// Configurando validator para nome de usuario**

**UserValidator = new UserValidator<ApplicationUser>(this)**

**{**

**AllowOnlyAlphanumericUserNames = false,**

**RequireUniqueEmail = true**

**};**

**// Logica de validação e complexidade de senha**

**PasswordValidator = new PasswordValidator**

**{**

**RequiredLength = 6,**

**RequireNonLetterOrDigit = false,**

**RequireDigit = false,**

**RequireLowercase = false,**

**RequireUppercase = false,**

**};**

**// Configuração de Lockout**

**UserLockoutEnabledByDefault = true;**

**DefaultAccountLockoutTimeSpan = TimeSpan.FromMinutes(5);**

**MaxFailedAccessAttemptsBeforeLockout = 5;**

**// Providers de Two Factor Autentication**

**RegisterTwoFactorProvider("Código via SMS", new PhoneNumberTokenProvider<ApplicationUser>**

**{**

**MessageFormat = "Seu código de segurança é: {0}"**

**});**

**RegisterTwoFactorProvider("Código via E-mail", new EmailTokenProvider<ApplicationUser>**

**{**

**Subject = "Código de Segurança",**

**BodyFormat = "Seu código de segurança é: {0}"**

**});**

**// Definindo a classe de serviço de e-mail**

**EmailService = new EmailService();**

**// Definindo a classe de serviço de SMS**

**SmsService = new SmsService();**

**var provider = new DpapiDataProtectionProvider("LearningCloud");**

**var dataProtector = provider.Create("ASP.NET Identity");**

**UserTokenProvider = new DataProtectorTokenProvider<ApplicationUser>(dataProtector);**

**}**

}

}

Adiciona a classe ApplicationSignInManager na camada de Infra.CrossCutting.**Identity** na pasta Configuration

* 3 - Infra / 3.2 - CrossCutting
* Projeto LearningCloud.Infra.CrossCutting.**Identity**
* Pasta Configuration (Clica com Direito)
  + Add - Class
  + ApplicationSignInManager.cs
  + definir como public
  + Herdar  *: SignInManager<ApplicationUser, string>*

Adiciona o código do método ApplicationSignInManager da IdentityConfig ficando:

using System.Security.Claims;

using System.Threading.Tasks;

using Microsoft.AspNet.Identity.Owin;

using Microsoft.Owin;

using Microsoft.Owin.Security;

using LearningCloud.Infra.CrossCutting.Identity.Models;

namespace LearningCloud.Infra.CrossCutting.Identity.Configuration

{

public class ApplicationSignInManager : SignInManager<ApplicationUser, string>

{

**public ApplicationSignInManager(ApplicationUserManager userManager, IAuthenticationManager authenticationManager)**

**: base(userManager, authenticationManager)**

**{**

**}**

**public override Task<ClaimsIdentity> CreateUserIdentityAsync(ApplicationUser user)**

**{**

**return user.GenerateUserIdentityAsync((ApplicationUserManager)UserManager);**

**}**

**public static ApplicationSignInManager Create(IdentityFactoryOptions<ApplicationSignInManager> options, IOwinContext context)**

**{**

**return new ApplicationSignInManager(context.GetUserManager<ApplicationUserManager>(), context.Authentication);**

**}**

}

}

Vamos criar uma classe de Context para o Ientity chamada ApplicationDbContext

Adiciona a classe ApplicationDbContext na camada de Infra.CrossCutting.**Identity** na pasta **ContextIdentity**

* 3 - Infra / 3.2 - CrossCutting
* Projeto LearningCloud.Infra.CrossCutting.**Identity**
* Pasta ContextIdentity (Clica com Direito)
  + Add - Class
  + ApplicationDbContext.cs
  + definir como public
  + Herdar  *: IdentityDbContext<ApplicationUser>*
  + Implementado a interfase , IDisposable

Adiciona o código do método ApplicationSignInManager da IdentityConfig ficando:

using System;

using Microsoft.AspNet.Identity.EntityFramework;

using LearningCloud.Infra.CrossCutting.Identity.Models;

namespace LearningCloud.Infra.CrossCutting.Identity.ContextIdentity

{

public class ApplicationDbContext : IdentityDbContext<ApplicationUser>, IDisposable

{

public ApplicationDbContext()

: base("LearningCloud", throwIfV1Schema: false)

{

}

public static ApplicationDbContext Create()

{

return new ApplicationDbContext();

}

}

}

Na camada de Domain vamos criar duas novas entidades na pasta Entities, uma entidade para Usuario e outra para UsuarioAcesso

Adiciona a classe na camada de Domain na pasta Entities

* 3 - Domain
* LearningCloud.Domain
* Pasta Entities (Clica com Direito)
  + Add - Class
  + Usuario.cs (NomeClasse.cs)
  + definir como public

using System;

namespace LearningCloud.Domain.Entities

{

public class Usuario

{

public Usuario()

{

Usuario\_Id = Guid.NewGuid().ToString();

}

public string usu\_id { get; set; }

public string usu\_nome { get; set; }

public string usu\_sobreNome { get; set; }

public DateTime? usu\_dataNascimento { get; set; }

public string usu\_sexo { get; set; }

public int usu\_nivel { get; set; }

public string usu\_status { get; set; }

public DateTime? usu\_dataCadastro { get; set; }

public string usu\_fotoPerfil { get; set; }

}

}

Adiciona a classe na camada de Domain na pasta Entities

* 3 - Domain
* LearningCloud.Domain
* Pasta Entities (Clica com Direito)
  + Add - Class
  + UsuarioAcesso.cs (NomeClasse.cs)
  + definir como public

using System;

namespace LearningCloud.Domain.Entities

{

public class UsuarioAcesso

{

public virtual string uac\_id { get; set; }

public virtual string uac\_email { get; set; }

public virtual bool uac\_emailConfirmed { get; set; }

public virtual string uac\_passwordHash { get; set; }

public virtual string uac\_securityStamp { get; set; }

public virtual string uac\_phoneNumber { get; set; }

public virtual bool uac\_phoneNumberConfirmed { get; set; }

public virtual bool uac\_twoFactorEnabled { get; set; }

public virtual DateTime? uac\_lockoutEndDateUtc { get; set; }

public virtual bool uac\_lockoutEnabled { get; set; }

public virtual int uac\_accessFailedCount { get; set; }

public virtual string uac\_userName { get; set; }

public virtual Usuario uac\_usuario { get; set; }

}

}

Na camada de Data vamos criar duas novas classes de configuração para a criação das tabelas de Usuario UsuarioAcesso na pasta EntityConfig, uma classe para UsuarioConfiguration.cs e outra para UsuarioAcessoConfiguration.cs

Adiciona a classe UsuarioConfiguration na camada de Infra.Data na pasta EntityConfig

* 4 - Infra / 4.1 - Data
* LearningCloud.Infra.Data
* Pasta EntityConfig (Clica com Direito)
  + Add - Class
  + UsuarioConfiguration.cs (NomeClasse.cs)
  + definir como public
  + Herdar  *:* EntityTypeConfiguration<Usuario>

using System.Data.Entity.ModelConfiguration;

using LearningCloud.Domain.Entities;

namespace LearningCloud.Infra.Data.EntityConfig

{

public class UsuarioConfiguration : EntityTypeConfiguration<Usuario>

{

**public UsuarioConfiguration()**

**{**

**HasKey(usu => usu.Usuario\_Id);**

**Property(usu => usu.Usuario\_Id)**

**.HasColumnName("usu\_id")**

**.IsRequired()**

**.HasMaxLength(128);**

**Property(usu => usu.Usuario\_Nome)**

**.HasColumnName("usu\_nome")**

**.IsRequired()**

**.HasMaxLength(256);**

**Property(usu => usu.Usuario\_SobreNome)**

**.HasColumnName("usu\_sobrenome")**

**.IsRequired()**

**.HasMaxLength(256);**

**Property(usu => usu.Usuario\_DataNascimento)**

**.HasColumnName("usu\_dataNascimento")**

**.IsRequired()**

**.HasColumnType("datetime2");**

**Property(usu => usu.Usuario\_Genero)**

**//.IsRequired()**

**.HasColumnType("char")**

**.HasMaxLength(1);**

**Property(usu => usu.Usuario\_GeneroDescricao)**

**.HasColumnName("usu\_generoDescricao")**

**.HasMaxLength(256);**

**Property(usu => usu.Usuario\_FotoPerfil)**

**.HasColumnName("usu\_fotoPerfil")**

**.HasMaxLength(256);**

**Property(usu => usu.Usuario\_Status)**

**.HasColumnName("usu\_status")**

**.IsRequired()**

**.HasColumnType("char")**

**.HasMaxLength(1);**

**Property(usu => usu.Usuario\_DataCadastro)**

**.HasColumnName("usu\_dataCadastro")**

**.IsRequired()**

**.HasColumnType("datetime2");**

**ToTable("LearningCloud\_Usuario");**

**}**

}

}

**public UsuarioAcessoConfiguration()**

**{**

**HasKey(uac => uac.UsuarioAcesso\_Id);**

**Property(uac => uac.UsuarioAcesso\_Id)**

**.HasColumnName("uac\_id");**

**Property(uac => uac.UsuarioAcesso\_UserName)**

**.HasColumnName("uac\_userName");**

**Property(uac => uac.UsuarioAcesso\_Email)**

**.HasColumnName("uac\_email");**

**Property(uac => uac.UsuarioAcesso\_EmailConfirmed)**

**.HasColumnName("uac\_emailConfirmed");**

**Property(uac => uac.UsuarioAcesso\_PasswordHash)**

**.HasColumnName("uac\_passwordHash");**

**Property(uac => uac.UsuarioAcesso\_SecurityStamp)**

**.HasColumnName("uac\_securityStamp");**

**Property(uac => uac.UsuarioAcesso\_PhoneNumber)**

**.HasColumnName("uac\_phoneNumber");**

**Property(uac => uac.UsuarioAcesso\_PhoneNumberConfirmed)**

**.HasColumnName("uac\_phoneNumberConfirmed");**

**Property(uac => uac.UsuarioAcesso\_TwoFactorEnabled)**

**.HasColumnName("uac\_twoFactorEnabled");**

**Property(uac => uac.UsuarioAcesso\_LockoutEndDateUtc)**

**.HasColumnName("uac\_lockoutEndDateUtc")**

**.HasColumnType("datetime2");**

**Property(uac => uac.UsuarioAcesso\_LockoutEnabled)**

**.HasColumnName("uac\_lockoutEnabled");**

**Property(uac => uac.UsuarioAcesso\_AccessFailedCount)**

**.HasColumnName("uac\_accessFailedCount");**

**Property(uac => uac.UsuarioAcesso\_Nivel)**

**.HasColumnName("uac\_nivelAcesso");**

**Property(uac => uac.UsuarioAcesso\_CodigoUsuario)**

**.HasColumnName("uac\_fk\_usuario");**

**HasRequired(uac => uac.UsuarioAcesso\_Usuario)**

**.WithMany()**

**.HasForeignKey(uac => uac.UsuarioAcesso\_CodigoUsuario);**

**ToTable("NewLearningCloud\_UsuarioAcesso");**

**}**

Vamos alterar o arquivo ApplicationDbContext para criar a estrutura de tabelas de usuario relacionada a tabela de user AspNetUsers (padrão do entity), mas vamos renomear esse padrão para LearningCloud\_UsuarioAcesso assim como as demais tabelas de Roles, UserRoles, Logins e Claims (LearningCloud\_Roles, LearningCloud\_UsuariosRoles, LearningCloud\_UsuariosLogins e LearningCloud\_UsuariosClaims )

Adiciona a referencia do Domain e Data na camada de Identity em Infra.CrossCutting.Identity

* LearningCloud.Infra.CrossCutting.Identity
  + References (Clica com Direito)
  + Add References
  + Solution
  + Marca a opção “LearningCloud.Domain”
  + Marca a opção “LearningCloud.Infra.Data”

**using System;**

**using System.Data.Entity;**

**using Microsoft.AspNet.Identity.EntityFramework;**

**using System.Data.Entity.ModelConfiguration.Conventions;**

using LearningCloud.Infra.CrossCutting.Identity.Models;

**using LearningCloud.Domain.Entities;**

**using LearningCloud.Infra.Data.EntityConfig;**

public class ApplicationDbContext : IdentityDbContext<ApplicationUser>, IDisposable

{

public ApplicationDbContext()

: base("LearningCloud", throwIfV1Schema: false)

{

}

**public DbSet<Usuario> LearningCloud\_Usuario { get; set; }**

public static ApplicationDbContext Create()

{

return new ApplicationDbContext();

}

**protected override void OnModelCreating(DbModelBuilder modelBuilder)**

**{**

**modelBuilder.Conventions.Remove<PluralizingTableNameConvention>();**

**modelBuilder.Conventions.Remove<OneToManyCascadeDeleteConvention>();**

**modelBuilder.Conventions.Remove<ManyToManyCascadeDeleteConvention>();**

**modelBuilder.Properties()**

**.Where(p => p.Name.Contains("\_id"))**

**.Configure(p => p.IsKey());**

**modelBuilder.Properties<string>()**

**.Configure(p => p.HasColumnType("varchar"));**

**modelBuilder.Properties<string>()**

**.Configure(p => p.HasMaxLength(100));**

**base.OnModelCreating(modelBuilder);**

**modelBuilder.Configurations.Add(new UsuarioConfiguration());**

**// modelBuilder.HasDefaultSchema("");**

**modelBuilder.Entity<ApplicationUser>().ToTable("NewLearningCloud\_UsuarioAcesso").Property(p => p.Id).HasColumnName("uac\_id");**

**modelBuilder.Entity<ApplicationUser>().ToTable("NewLearningCloud\_UsuarioAcesso").Property(p => p.Email).HasColumnName("uac\_email").IsRequired();**

**modelBuilder.Entity<ApplicationUser>().ToTable("NewLearningCloud\_UsuarioAcesso").Property(p => p.EmailConfirmed).HasColumnName("uac\_emailConfirmed");**

**modelBuilder.Entity<ApplicationUser>().ToTable("NewLearningCloud\_UsuarioAcesso").Property(p => p.PasswordHash).HasColumnName("uac\_passwordHash");**

**modelBuilder.Entity<ApplicationUser>().ToTable("NewLearningCloud\_UsuarioAcesso").Property(p => p.SecurityStamp).HasColumnName("uac\_securityStamp");**

**modelBuilder.Entity<ApplicationUser>().ToTable("NewLearningCloud\_UsuarioAcesso").Property(p => p.PhoneNumber).HasColumnName("uac\_phoneNumber");**

**modelBuilder.Entity<ApplicationUser>().ToTable("NewLearningCloud\_UsuarioAcesso").Property(p => p.PhoneNumberConfirmed).HasColumnName("uac\_phoneNumberConfirmed");**

**modelBuilder.Entity<ApplicationUser>().ToTable("NewLearningCloud\_UsuarioAcesso").Property(p => p.TwoFactorEnabled).HasColumnName("uac\_twoFactorEnabled");**

**modelBuilder.Entity<ApplicationUser>().ToTable("NewLearningCloud\_UsuarioAcesso").Property(p => p.LockoutEndDateUtc).HasColumnName("uac\_lockoutEndDateUtc");**

**modelBuilder.Entity<ApplicationUser>().ToTable("NewLearningCloud\_UsuarioAcesso").Property(p => p.LockoutEnabled).HasColumnName("uac\_lockoutEnabled");**

**modelBuilder.Entity<ApplicationUser>().ToTable("NewLearningCloud\_UsuarioAcesso").Property(p => p.AccessFailedCount).HasColumnName("uac\_accessFailedCount");**

**modelBuilder.Entity<ApplicationUser>().ToTable("NewLearningCloud\_UsuarioAcesso").Property(p => p.UsuarioAcesso\_Nivel).HasColumnName("uac\_nivelAcesso"); //.IsOptional();**

**modelBuilder.Entity<ApplicationUser>().ToTable("NewLearningCloud\_UsuarioAcesso").Property(p => p.UserName).HasColumnName("uac\_userName");**

**modelBuilder.Entity<ApplicationUser>().ToTable("NewLearningCloud\_UsuarioAcesso").HasRequired(x => x.UsuarioAcesso\_Usuario).WithRequiredDependent().Map(p => p.MapKey("uac\_fk\_usuario"));**

**modelBuilder.Entity<IdentityUserRole>().ToTable("NewLearningCloud\_UsuariosRoles").Property(p => p.UserId).HasColumnName("uro\_usuarioId");**

**modelBuilder.Entity<IdentityUserRole>().ToTable("NewLearningCloud\_UsuariosRoles").Property(p => p.RoleId).HasColumnName("uro\_roleId");**

**modelBuilder.Entity<IdentityUserLogin>().ToTable("NewLearningCloud\_UsuariosLogins").Property(p => p.UserId).HasColumnName("ulo\_usuarioId");**

**modelBuilder.Entity<IdentityUserLogin>().ToTable("NewLearningCloud\_UsuariosLogins").Property(p => p.LoginProvider).HasColumnName("ulo\_loginProvider");**

**modelBuilder.Entity<IdentityUserLogin>().ToTable("NewLearningCloud\_UsuariosLogins").Property(p => p.ProviderKey).HasColumnName("ulo\_providerKey");**

**modelBuilder.Entity<IdentityUserClaim>().ToTable("NewLearningCloud\_UsuariosClaims").Property(p => p.Id).HasColumnName("ucl\_id");**

**modelBuilder.Entity<IdentityUserClaim>().ToTable("NewLearningCloud\_UsuariosClaims").Property(p => p.UserId).HasColumnName("ucl\_usuarioId");**

**modelBuilder.Entity<IdentityUserClaim>().ToTable("NewLearningCloud\_UsuariosClaims").Property(p => p.ClaimType).HasColumnName("ucl\_claimType");**

**modelBuilder.Entity<IdentityUserClaim>().ToTable("NewLearningCloud\_UsuariosClaims").Property(p => p.ClaimValue).HasColumnName("ucl\_claimValue");**

**modelBuilder.Entity<IdentityRole>().ToTable("NewLearningCloud\_Roles").Property(p => p.Id).HasColumnName("rol\_id");**

**modelBuilder.Entity<IdentityRole>().ToTable("NewLearningCloud\_Roles").Property(p => p.Name).HasColumnName("rol\_name");**

**}**

**public override int SaveChanges()**

**{**

**foreach (var entry in this.ChangeTracker.Entries())**

**{**

**if (entry.State != EntityState.Deleted)**

**{**

**string dataCadastro = null;**

**string dataAlteracao = null;**

**foreach (string o in entry.CurrentValues.PropertyNames)**

**{**

**var property = entry.Property(o);**

**if (property.Name.Contains("\_DataCadastro"))**

**{**

**dataCadastro = property.Name;**

**}**

**if (property.Name.Contains("\_DataAlteracao"))**

**{**

**dataAlteracao = property.Name.ToString();**

**}**

**}**

**if (entry.State == EntityState.Added)**

**{**

**if (dataCadastro != null)**

**{**

**entry.Property(dataCadastro).CurrentValue = DateTime.Now;**

**}**

**if (dataAlteracao != null)**

**{**

**entry.Property(dataAlteracao).CurrentValue = null;**

**}**

**}**

**if (entry.State == EntityState.Modified)**

**{**

**if (dataCadastro != null)**

**{**

**entry.Property(dataCadastro).IsModified = false;**

**}**

**if (dataAlteracao != null)**

**{**

**entry.Property(dataAlteracao).CurrentValue = DateTime.Now;**

**}**

**}**

**}**

**}**

**try**

**{**

**return base.SaveChanges();**

**}**

**catch (System.Data.Entity.Validation.DbEntityValidationException e)**

**{**

**foreach (var eve in e.EntityValidationErrors)**

**{**

**Console.WriteLine("Entidade do tipo \"{0}\" no estado \"{1}\" tem os seguintes erros de validação:",**

**eve.Entry.Entity.GetType().Name, eve.Entry.State);**

**foreach (var ve in eve.ValidationErrors)**

**{**

**Console.WriteLine("- Property: \"{0}\", Erro: \"{1}\"",**

**ve.PropertyName, ve.ErrorMessage);**

**}**

**}**

**throw;**

**}**

**//return base.SaveChanges();**

**}**

}

Alterar classe ApplicationUser da camada de CrossCutting.Identity na pasta Models para adicionar um campo de referência para a tabela de Usuário na tabela de usuario acesso

using System.Security.Claims;

using System.Threading.Tasks;

using Microsoft.AspNet.Identity;

using Microsoft.AspNet.Identity.EntityFramework;

**using LearningCloud.Domain.Entities;**

namespace LearningCloud.Infra.CrossCutting.Identity.Models

{

public class ApplicationUser : IdentityUser

{

public async Task<ClaimsIdentity> GenerateUserIdentityAsync(UserManager<ApplicationUser> manager)

{

// Note the authenticationType must match the one defined in CookieAuthenticationOptions.AuthenticationType

var userIdentity = await manager.CreateIdentityAsync(this, DefaultAuthenticationTypes.ApplicationCookie);

// Add custom user claims here

return userIdentity;

}

**public int UsuarioAcesso\_Nivel { get; set; }**

**public virtual Usuario UsuarioAcesso\_Usuario { get; set; }**

}

}

Habilitar o Migrations para criar o banco de Dados

* Package Manager Console
  + seleciona o *Default project* (LearningCloud.Infra.CrossCutting.Identity)
  + Enable-Migrations

Na pasta Migrations abrir o arquivo Configuration.cs e alterar o AutomaticMigrationsEnabled para true

public Configuration()

{

**AutomaticMigrationsEnabled = true;**

}

ainda no arquivo Configuration.cs limpa os using não utilizados

using System;

using System.Data.Entity;

using System.Linq;

e remove as partes em vermelho

internal sealed class Configuration : DbMigrationsConfiguration <LearningCloud.Infra.CrossCutting.Identity.ContextIdentity.ApplicationDbContext>

{

...

}

protected override void Seed(LearningCloud.Infra.CrossCutting.Identity.ContextIdentity.ApplicationDbContext context)

{

...

}

ficando

internal sealed class Configuration : DbMigrationsConfiguration<ContextIdentity.ApplicationDbContext>

{

...

}

protected override void SeedContextIdentity.ApplicationDbContext context)

{

…

}

Executar o Update database

* Package Manager Console
  + seleciona o Default project (LearningCloud.Infra.CrossCutting.Identity)
  + Update-Database -Verbose -Force

Adiciona a classe de ViewModel RegisterViewModel na camada de Infra.CrossCutting.**Identity** na pasta ViewModels/ AccountViewModels

* 3 - Infra / 3.2 - CrossCutting
* Projeto LearningCloud.Infra.CrossCutting.**Identity**
* Pasta ViewModels/ AccountViewModels (Clica com Direito)
  + Add - Class
  + RegisterViewModel.cs
  + definir como public

Adiciona o seguinte código ficando:

using System;

using System.ComponentModel.DataAnnotations;

namespace LearningCloud.Infra.CrossCutting.Identity.Models.AccountViewModels

{

public class RegisterViewModel

{

**[Required]**

**[Display(Name = "Usuário")]**

**public string UsuarioAcesso\_UserName { get; set; }**

**[Required]**

**[Display(Name = "Nome")]**

**public string Usuario\_Nome { get; set; }**

**[Required]**

**[Display(Name = "Sobrenome")]**

**public string Usuario\_SobreNome { get; set; }**

**[Required]**

**[EmailAddress]**

**[Display(Name = "E-mail")]**

**public string UsuarioAcesso\_Email { get; set; }**

**[Required]**

**[DataType(DataType.Date)]**

**[Display(Name = "Data Nascimento")]**

**[DisplayFormat(ApplyFormatInEditMode = true, DataFormatString = "{0:dd/MM/yyyy}")]**

**public DateTime Usuario\_DataNascimento { get; set; }**

**[Required]**

**[StringLength(10, ErrorMessage = "Selecione o genero.")]**

**[Display(Name = "Genero")]**

**public string Usuario\_Genero { get; set; }**

**public string Usuario\_GeneroDescricao { get; set; }**

**[Required]**

**[StringLength(100, ErrorMessage = "A {0} deve conter pelo menos {2} caracteres.", MinimumLength = 6)]**

**[DataType(DataType.Password)]**

**[Display(Name = "Senha")]**

**public string Password { get; set; }**

**[Required(ErrorMessage = "Preencha o campo Confirmação da Senha.")]**

**[DataType(DataType.Password)]**

**[Display(Name = "Confirmação da Senha")]**

**[Compare("Password", ErrorMessage = "A senha e a confirmação da senha estão diferentes.")]**

**public string ConfirmPassword { get; set; }**

}

}

Vamos instalar o *EntityFramework*, *AspNet.Identity.Core* e *AspNet.Identity.EntityFramework na camada de* ***IoC***

* Package Manager Console
  + seleciona o *Default project* (LearningCloud.Infra.CrossCutting.**IoC**)

Install-Package Microsoft.AspNet.Identity.Core

* + Install-Package Microsoft.AspNet.Identity.EntityFramework
  + Install-Package Microsoft.AspNet.Identity.Owin
  + Install-Package Microsoft.Owin

Adiciona a referência do CrossCutting.Identity na camada de CrossCutting.IoC

* LearningCloud.Infra.CrossCutting.IoC
  + References (Clica com Direito)
  + Add References
  + Marca a opção “LearningCloud.Infra.CrossCutting.Identity”

Na classe BootStrapper da camada de IoC incluir os registros para as implementações do Identity

**using NewLearningCloud.Infra.CrossCutting.Identity.ContextIdentity;**

**using Microsoft.AspNet.Identity;**

**using NewLearningCloud.Infra.CrossCutting.Identity.ViewModels;**

**using Microsoft.AspNet.Identity.EntityFramework;**

**using NewLearningCloud.Infra.CrossCutting.Identity.Configuration;**

public class BootStrapper

{

public static void RegisterServices(Container container)

{

container.Register<IUnitOfWorkRepository, UnitOfWorkRepository>();

container.RegisterPerWebRequest<IAulaRepository, AulaRepository>();

container.RegisterPerWebRequest<IAulaAppService, AulaAppService>();

container.RegisterPerWebRequest<IAulaService, AulaService>();

container.RegisterPerWebRequest<IAssinaturaNivelRepository, AssinaturaNivelRepository>();

container.RegisterPerWebRequest<IAssinaturaNivelAppService, AssinaturaNivelAppService>();

container.RegisterPerWebRequest<IAssinaturaNivelService, AssinaturaNivelService>();

**//\*\*\*\*\* Registro de dependências para o Identity \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**container.Register<ApplicationDbContext>(Lifestyle.Scoped);**

**container.Register<IUserStore<ApplicationUser>>(() => new UserStore<ApplicationUser>(new ApplicationDbContext()),Lifestyle.Scoped);**

**container.Register<IRoleStore<IdentityRole, string>>(() => new RoleStore<IdentityRole>(), Lifestyle.Scoped);**

**container.Register<ApplicationUserManager>(Lifestyle.Scoped);**

**container.Register<ApplicationSignInManager>(Lifestyle.Scoped);**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

var adapter = new SimpleInjectorServiceLocatorAdapter(container);

ServiceLocator.SetLocatorProvider(() => adapter);

}

}

Vamos alterar o arquivo **SimpleInjectorInitializer** na camada de Presentation que está na pasta App\_Start

Ficando:

**using System.Web;**

public static class SimpleInjectorInitializer

{

public static void Initialize()

{

var container = new Container();

container.Options.DefaultScopedLifestyle = new WebRequestLifestyle();

// Chamada dos módulos do Simple Injector

InitializeContainer(container);

**// Necessário para registrar o ambiente do Owin que é dependência do Identity**

**// Feito fora da camada de IoC para não levar o System.Web para fora**

**container.Register(() =>**

**{**

**if (HttpContext.Current != null && HttpContext.Current.Items["owin.Environment"] == null && container.IsVerifying)**

**{**

**return new OwinContext().Authentication;**

**}**

**return HttpContext.Current.GetOwinContext().Authentication;**

**}, Lifestyle.Scoped);**

container.RegisterMvcControllers(Assembly.GetExecutingAssembly());

container.Verify();

DependencyResolver.SetResolver(new SimpleInjectorDependencyResolver(container));

}

Vamos ter que instalar o Owin *camada de* ***Presentation***

* Package Manager Console
  + seleciona o *Default project* (LearningCloud.MVC)
  + Install-Package Microsoft.Owin
  + Install-Package Microsoft.Owin.Host.SystemWeb
  + Install-Package Microsoft.Owin.Security
  + Install-Package Microsoft.Owin.Security.Cookies
  + Install-Package Microsoft.AspNet.Identity.Core
  + Install-Package Microsoft.AspNet.Identity.EntityFramework
  + Install-Package Microsoft.AspNet.Identity.Owin

Adicionar a referencia do projeto LearningCloud.Infra.CrossCutting.Identity na camada MVC

Adiciona a referencia do CrossCutting.Identity na camada de MVC

* LearningCloud.MVC
  + References (Clica com Direito)
  + Add References
  + Marca a opção “LearningCloud.Infra.CrossCutting.Identity”

Vamos criar a classe Startup.Auth na pasta App\_Start na camada de MVC

na pasta de App\_Start (0 - Presentation)

* 0 - Presentation > LearningCloud.MVC
* Pasta App\_Start (Clica com Direito)
* Add
* Class
* name: Startup.Auth**.cs**
* definir como partial

Adicionar código do projeto padrão

Ficando

using System;

using System.Web.Mvc;

using Owin;

using Microsoft.AspNet.Identity;

using Microsoft.AspNet.Identity.Owin;

using Microsoft.Owin;

using Microsoft.Owin.Security.Cookies;

using Microsoft.Owin.Security.DataProtection;

using LearningCloud.Infra.CrossCutting.Identity.Configuration;

using LearningCloud.Infra.CrossCutting.Identity.ViewModels;

//[assembly: OwinStartup(typeof(LearningCloud.MVC.Startup))]

namespace LearningCloud.MVC

{

public partial class Startup

{

public static IDataProtectionProvider DataProtectionProvider { get; set; }

// For more information on configuring authentication, please visit http://go.microsoft.com/fwlink/?LinkId=301864

public void ConfigureAuth(IAppBuilder app)

{

// Configure the db context, user manager and signin manager to use a single instance per request

app.CreatePerOwinContext(() => DependencyResolver.Current.GetService<ApplicationUserManager>());

// Enable the application to use a cookie to store information for the signed in user

// and to use a cookie to temporarily store information about a user logging in with a third party login provider

// Configure the sign in cookie

app.UseCookieAuthentication(new CookieAuthenticationOptions

{

AuthenticationType = DefaultAuthenticationTypes.ApplicationCookie,

LoginPath = new PathString("/Conta/Entrar"),

Provider = new CookieAuthenticationProvider

{

// Enables the application to validate the security stamp when the user logs in.

// This is a security feature which is used when you change a password or add an external login to your account.

OnValidateIdentity = SecurityStampValidator.OnValidateIdentity<ApplicationUserManager, ApplicationUser>(

validateInterval: TimeSpan.FromMinutes(30),

regenerateIdentity: (manager, user) => user.GenerateUserIdentityAsync(manager))

}

});

app.UseExternalSignInCookie(DefaultAuthenticationTypes.ExternalCookie);

// Enables the application to temporarily store user information when they are verifying the second factor in the two-factor authentication process.

app.UseTwoFactorSignInCookie(DefaultAuthenticationTypes.TwoFactorCookie, TimeSpan.FromMinutes(5));

// Enables the application to remember the second login verification factor such as phone or email.

// Once you check this option, your second step of verification during the login process will be remembered on the device where you logged in from.

// This is similar to the RememberMe option when you log in.

app.UseTwoFactorRememberBrowserCookie(DefaultAuthenticationTypes.TwoFactorRememberBrowserCookie);

// Uncomment the following lines to enable logging in with third party login providers

//app.UseMicrosoftAccountAuthentication(

// clientId: "",

// clientSecret: "");

//app.UseTwitterAuthentication(

// consumerKey: "",

// consumerSecret: "");

//app.UseFacebookAuthentication(

// appId: "",

// appSecret: "");

//app.UseGoogleAuthentication(new GoogleOAuth2AuthenticationOptions()

//{

// ClientId = "",

// ClientSecret = ""

//});

}

}

}

adicionar a classe Startup no projeto MVC

Vamos criar a classe Startup.Auth na pasta App\_Start na camada de MVC

no projeto LearningCloud.MVC (0 - Presentation)

* 0 - Presentation > LearningCloud.MVC (Clica com Direito)
* Add
* New Item...
* OWIN Startup Class
* Item name: Startup
* definir como partial

Adicionando a chamada do metodo

**ConfigureAuth(app);**

Ficando:

using System;

using System.Threading.Tasks;

using Microsoft.Owin;

using Owin;

using LearningCloud.MVC;

[assembly: OwinStartup(typeof(Startup))]

namespace LearningCloud.MVC

{

public partial class Startup

{

public void Configuration(IAppBuilder app)

{

**ConfigureAuth(app);**

}

}

}

Criar Controller Account

AccountController

na pasta de Controllers (0 - Presentation / Controllers )

* 0 - Presentation > LearningCloud.MVC
* Pasta Controllers (Clica com Direito)
* Add
* Controller…
* MVC 5 Controller - Empty
* [Add]
* Controller Name: AccountController
* [Add]

Remover os seguintes códigos em vermelho

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace LearningCloud.MVC.Controllers

{

public class AccountController: Controller

{

// GET: Default

public ActionResult Index()

{

return View();

}

}

}

Vamos acrescentar a decoração [Authorize] na classe e criar dois campos e um construtor para injetarmos esses campos do tipo ApplicationSignInManager e ApplicationUserManager

namespace LearningCloud.MVC.Controllers

{

**[Authorize]**

public class AccountController : Controller

{

**private ApplicationSignInManager \_signInManager;**

**private ApplicationUserManager \_userManager;**

**public AccountController(ApplicationUserManager userManager, ApplicationSignInManager signInManager)**

**{**

**\_userManager = userManager;**

**\_signInManager = signInManager;**

**}**

}

}

Adiciona o using

**using LearningCloud.Infra.CrossCutting.Identity.Configuration;**

No Final da classe de controller vamos criar uma #region com os seguintes metodos:

namespace LearningCloud.MVC.Controllers

{

[Authorize]

public class AccountController : Controller

{

private ApplicationSignInManager \_signInManager;

private ApplicationUserManager \_userManager;

public AccountController(ApplicationUserManager userManager, ApplicationSignInManager signInManager)

{

\_userManager = userManager;

\_signInManager = signInManager;

}

**#region Helpers**

**// Used for XSRF protection when adding external logins**

**private const string XsrfKey = "XsrfId";**

**private IAuthenticationManager AuthenticationManager**

**{**

**get**

**{**

**return HttpContext.GetOwinContext().Authentication;**

**}**

**}**

**private void AddErrors(IdentityResult result)**

**{**

**foreach (var error in result.Errors)**

**{**

**ModelState.AddModelError("", error);**

**}**

**}**

**private ActionResult RedirectToLocal(string returnUrl)**

**{**

**if (Url.IsLocalUrl(returnUrl))**

**{**

**return Redirect(returnUrl);**

**}**

**return RedirectToAction("Index", "Home");**

**}**

**#endregion**

}

}

Adiciona os usings

**using Microsoft.Owin.Security;**

**using Microsoft.AspNet.Identity;**

Vamos criar um método para sobrescrever o Dispose para destruir as instâncias que injetamos no controller

|  |
| --- |
| **protected override void Dispose(bool disposing)**  **{**  **if (disposing)**  **{**  **if (\_userManager != null)**  **{**  **\_userManager.Dispose();**  **\_userManager = null;**  **}**  **if (\_signInManager != null)**  **{**  **\_signInManager.Dispose();**  **\_signInManager = null;**  **}**  **}**  **base.Dispose(disposing);**  **}**  #region Helpers |

Vamos criar dois ActionResult (Get e Post) para cadastrar usuários (Register)

namespace LearningCloud.MVC.Controllers

{

[Authorize]

public class AccountController : Controller

{

private ApplicationSignInManager \_signInManager;

private ApplicationUserManager \_userManager;

public AccountController(ApplicationUserManager userManager, ApplicationSignInManager signInManager)

{

\_userManager = userManager;

\_signInManager = signInManager;

}

**// GET: /Account/Register**

**[AllowAnonymous]**

**public ActionResult Register()**

**{**

**return View();**

**}**

**//**

**// POST: /Account/Register**

**[HttpPost]**

**[AllowAnonymous]**

**[ValidateAntiForgeryToken]**

**public async Task<ActionResult> Register(RegisterViewModel model)**

**{**

**if (ModelState.IsValid)**

**{**

**var user = new ApplicationUser**

**{**

**UserName = model.UsuarioAcesso\_UserName,**

**Email = model.UsuarioAcesso\_Email,**

**UsuarioAcesso\_Nivel = 50,**

**UsuarioAcesso\_Usuario = new Domain.Entities.Usuario**

**{**

**Usuario\_Nome = model.Usuario\_Nome,**

**Usuario\_SobreNome = model.Usuario\_SobreNome,**

**Usuario\_DataNascimento = model.Usuario\_DataNascimento,**

**Usuario\_DataCadastro = DateTime.Now.ToUniversalTime(),**

**Usuario\_Status = "A"**

**}**

**};**

**var result = await \_userManager.CreateAsync(user, model.ConfirmPassword);**

**if (result.Succeeded)**

**{**

**await \_signInManager.SignInAsync(user, isPersistent: false, rememberBrowser: false);**

**var code = await \_userManager.GenerateEmailConfirmationTokenAsync(user.Id);**

**var callbackUrl = Url.Action("ConfirmEmail", "Account", new { userId = user.Id, code = code }, protocol: Request.Url.Scheme);**

**await \_userManager.SendEmailAsync(user.Id, "Confirme sua Conta", "Por favor confirme sua conta clicando neste link: <a href='" + callbackUrl + "'></a>");**

**ViewBag.Link = callbackUrl;**

**return View("DisplayEmail");**

**}**

**AddErrors(result);**

**}**

**// If we got this far, something failed, redisplay form**

**return View(model);**

**}**

…

Adiciona os usings

**using System.Threading.Tasks;**

**using NewLearningCloud.Infra.CrossCutting.Identity.ViewModels;**

**using LearningCloud.Infra.CrossCutting.Identity.Models.AccountViewModels;**

O controller completo ficará como a seguir:

**using System;**

**using System.Web;**

**using System.Web.Mvc;**

**using System.Threading.Tasks;**

**using Microsoft.AspNet.Identity;**

**using Microsoft.Owin.Security;**

**using LearningCloud.Infra.CrossCutting.Identity.Configuration;**

**using LearningCloud.Infra.CrossCutting.Identity.Models;**

**using LearningCloud.Infra.CrossCutting.Identity.Models.AccountViewModels;**

**namespace LearningCloud.MVC.Controllers**

**{**

**[Authorize]**

**public class AccountController : Controller**

**{**

**private ApplicationSignInManager \_signInManager;**

**private ApplicationUserManager \_userManager;**

**public AccountController(ApplicationUserManager userManager, ApplicationSignInManager signInManager)**

**{**

**\_userManager = userManager;**

**\_signInManager = signInManager;**

**}**

**// GET: /Account/Register**

**[AllowAnonymous]**

**public ActionResult Register()**

**{**

**return View();**

**}**

//

**// POST: /Account/Register**

**[HttpPost]**

**[AllowAnonymous]**

**[ValidateAntiForgeryToken]**

**public async Task<ActionResult> Register(RegisterViewModel model)**

**{**

**if (ModelState.IsValid)**

**{**

**var user = new ApplicationUser**

**{**

**UserName = model.UsuarioAcesso\_UserName,**

**Email = model.UsuarioAcesso\_Email,**

**UsuarioAcesso\_Nivel = 50,**

**UsuarioAcesso\_Usuario = new Domain.Entities.Usuario**

**{**

**Usuario\_Nome = model.Usuario\_Nome,**

**Usuario\_SobreNome = model.Usuario\_SobreNome,**

**Usuario\_DataNascimento = model.Usuario\_DataNascimento,**

**Usuario\_DataCadastro = DateTime.Now.ToUniversalTime(),**

**Usuario\_Status = "A"**

**}**

**};**

**var result = await \_userManager.CreateAsync(user, model.ConfirmPassword);**

**if (result.Succeeded)**

**{**

**await \_signInManager.SignInAsync(user, isPersistent: false, rememberBrowser: false);**

**return RedirectToAction("DisplayEmail", new { Id = user.Id });**

**}**

**AddErrors(result);**

**}**

**// If we got this far, something failed, redisplay form**

**return View(model);**

**}**

**// GET: /Account/DisplayEmail/Id**

**[Route("Email/{Id}")]**

**public async Task<ActionResult> DisplayEmail(string Id)**

**{**

**var code = await \_userManager.GenerateEmailConfirmationTokenAsync(Id);**

**var callbackUrl = Url.Action("ConfirmEmail", "Account", new { userId = Id, code = code }, protocol: Request.Url.Scheme);**

**await \_userManager.SendEmailAsync(Id, "Confirme sua Conta", "Por favor confirme sua conta clicando neste link: <a href='" + callbackUrl + "'></a>");**

**ViewBag.Link = callbackUrl;**

**return View();**

**}**

**#region Helpers**

**// Used for XSRF protection when adding external logins**

**private const string XsrfKey = "XsrfId";**

**private IAuthenticationManager AuthenticationManager**

**{**

**get**

**{**

**return HttpContext.GetOwinContext().Authentication;**

**}**

**}**

**private void AddErrors(IdentityResult result)**

**{**

**foreach (var error in result.Errors)**

**{**

**ModelState.AddModelError("", error);**

**}**

**}**

**private ActionResult RedirectToLocal(string returnUrl)**

**{**

**if (Url.IsLocalUrl(returnUrl))**

**{**

**return Redirect(returnUrl);**

**}**

**return RedirectToAction("Index", "Home");**

**}**

**#endregion**

**}**

**}**

vamos criar a view para a Register

clica com direito no método View do ActionResult Register()

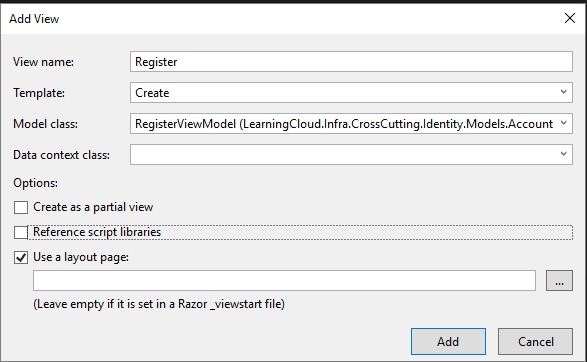
return **View**();

Add View…

View name: Register

Template: Create

Model class: RegisterViewModel (LearningCloud.Infra.CrossCutting.Identity.Models...)



Ficando:

@model LearningCloud.Infra.CrossCutting.Identity.Models.AccountViewModels.RegisterViewModel

@{

ViewBag.Title = "Cadastrar";

}

<h2>Register</h2>

@using (Html.BeginForm())

{

@Html.AntiForgeryToken()

<div class="form-horizontal">

<h4>RegisterViewModel</h4>

<hr />

@Html.ValidationSummary(true, "", new { @class = "text-danger" })

<div class="form-group">

@Html.LabelFor(model => model.UsuarioAcesso\_UserName, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.UsuarioAcesso\_UserName, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.UsuarioAcesso\_UserName, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.Usuario\_Nome, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Usuario\_Nome, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Usuario\_Nome, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.Usuario\_SobreNome, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Usuario\_SobreNome, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Usuario\_SobreNome, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.UsuarioAcesso\_Email, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.UsuarioAcesso\_Email, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.UsuarioAcesso\_Email, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.Usuario\_DataNascimento, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Usuario\_DataNascimento, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Usuario\_DataNascimento, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.Usuario\_Genero, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Usuario\_Genero, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Usuario\_Genero, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.Usuario\_GeneroDescricao, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Usuario\_GeneroDescricao, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Usuario\_GeneroDescricao, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.Password, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Password, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Password, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.ConfirmPassword, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.ConfirmPassword, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.ConfirmPassword, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

<div class="col-md-offset-2 col-md-10">

<input type="submit" value="Create" class="btn btn-default" />

</div>

</div>

</div>

}

<div>

@Html.ActionLink("Back to List", "Index")

</div>

Na Controller AcountController vamos criar as actions para confirmar e-mail e logoff

**// GET: /Account/ConfirmEmail**

**[AllowAnonymous]**

**[Route("Email/Confirmar/{userId}")]**

**public async Task<ActionResult> ConfirmEmail(string userId, string code)**

**{**

**if (userId == null || code == null)**

**{**

**return View("Error");**

**}**

**var result = await \_userManager.ConfirmEmailAsync(userId, code);**

**return View(result.Succeeded ? "ConfirmEmail" : "Error");**

**}**

**// POST: /Account/LogOff**

**[HttpPost]**

**[ValidateAntiForgeryToken]**

**public ActionResult LogOff()**

**{**

**AuthenticationManager.SignOut();**

**return RedirectToAction("Index", "Home");**

**}**

adicionar **using System.Net;**

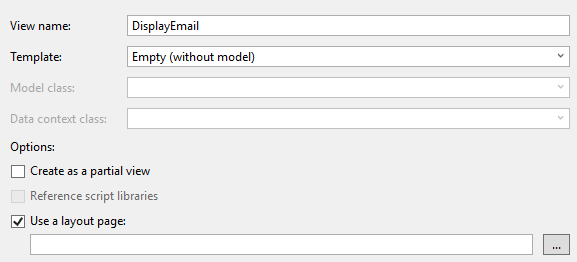
vamos criar a view para a DisplayEmail

clica com direito na pasta Views / Account

Add View…

View name: DisplayEmail

Template: Empty (without model)



Ficando:

@{

ViewBag.Title = "Verificar E-mail";

}

<section class="content-header">

<h1>@ViewBag.Title <small></small> </h1>

<ol class="breadcrumb">

<li><a href="#"><i class="fa fa-dashboard"></i> Level</a></li>

<li class="active">Here</li>

</ol>

</section>

<section class="content">

<div class="box box-default">

<div class="box-header with-border">

<h3 class="box-title">zzzzzzzzz</h3>

</div>

<div class="box-body">

<div class="container">

<div class="row">

<div class="col-lg-12 col-md-12">

<p class="text-info">

Verifique seu e-mail e confirme seu endereço.

</p>

<p class="text-danger">

DEMO: Caso o e-mail não chegue clique neste link: <a href="@ViewBag.Link">link</a>

</p>

<p class="text-danger">

Exemplo do link que vai no e-mail: <br />

@ViewBag.Link

</p>

</div>

</div>

</div>

</div><!-- /.box-body -->

</div><!-- /.box box-default -->

</section><!-- /.content -->

vamos criar a view para a ConfirmEmail

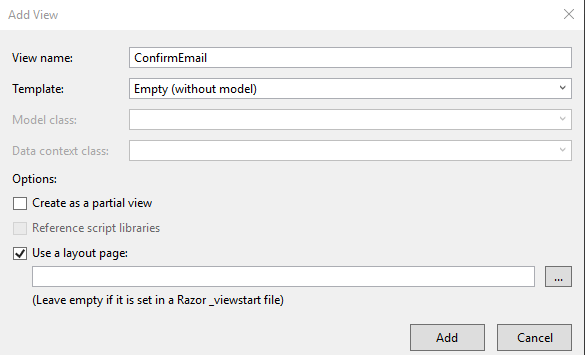
clica com direito no método View do Task<ActionResult> ConfirmEmail(string userId, string code)

return **View**(result.Succeeded ? "ConfirmEmail" : "Error");

Add View…

View name: ConfirmEmail

Template: Empty (without model)



altere a view para ficar como a seguir:

**@{**

**ViewBag.Title = "Confirmar E-mail";**

**}**

**<h2>@ViewBag.Title.</h2>**

**<section class="content-header">**

**<h1> @ViewBag.Title. <small></small> </h1>**

**<ol class="breadcrumb">**

**<li><a href="#"><i class="fa fa-dashboard"></i> Level</a></li>**

**<li class="active">Here</li>**

**</ol>**

**</section>**

**<section class="content">**

**<div class="box box-default">**

**<div class="box-header with-border">**

**<h3 class="box-title">zzzzzzzzz</h3>**

**</div>**

**<div class="box-body">**

**<div class="container">**

**<div class="row">**

**<div class="col-lg-12 col-md-12">**

**<p>**

**Obrigado por confirmar seu e-mail. Por favor @Html.ActionLink("Clique aqui para continuar", "Index", "Home", routeValues: null, htmlAttributes: new { id = "loginLink" })**

**</p>**

**</div>**

**</div>**

**</div>**

**</div><!-- /.box-body -->**

**</div><!-- /.box box-default -->**

**</section><!-- /.content -->**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

criar \_LoginPartial.cshtml na pasta de View / Shared do portal inicial

Add

New Item

MVC 5 Partial Page (Razor)

Name: \_LoginPartial.cshtml

|  |
| --- |
| **@if (Request.IsAuthenticated)**  **{**  **using (Html.BeginForm("LogOff", "Account", FormMethod.Post, new { id = "logoutForm", @class = "navbar-right" }))**  **{**  **@Html.AntiForgeryToken()**  **}**  **<ul id="top-menu" class="nav navbar-nav navbar-right">**  **<li><a href="#" id="mu-search-icon"><i class="fa fa-search"></i></a></li>**  **<li class="dropdown">**  **<a class="dropdown-toggle" data-toggle="dropdown" href='#' id="userLink">**  **<i class="fa fa-user">&nbsp;&nbsp;</i>@User.Identity.Name <i class="fa fa-angle-down"></i>**  **</a>**  **<ul class="dropdown-menu dropdown-user">**  **<li>**  **<div class="text-center">**  **<span class="">Entrou como:</span>**  **<span class="user-name"><small>@User.Identity.Name</small></span>**  **</div>**  **</li>**  **<li class="divider"></li>**  **<li>**  **<a href="#">**  **<i class="fa fa-id-card-o">&nbsp;&nbsp;</i>Minha Conta**  **</a>**  **</li>**  **<li>**  **<a href="#">**  **<i class="fa fa-cogs">&nbsp;&nbsp;</i>Alterar Senha**  **</a>**  **</li>**  **<li class="divider"></li>**  **<li>**  **<a href="javascript:document.getElementById('logoutForm').submit()">**  **<i class="fa fa-sign-out">&nbsp;&nbsp;</i>Sair**  **</a>**  **</li>**  **</ul>**  **</li>**  **<li class="dropdown">**  **<a class="dropdown-toggle count-info" data-hover="dropdown" data-toggle="dropdown" href="#" id="settingsLink">**  **<i class="fa fa-cog">&nbsp;</i>**  **<i class="fa fa-angle-down"></i>**  **</a>**  **<ul class="dropdown-menu dropdown-user">**  **<li><a href="@Url.Action("Index", "Home", new { Area = "Admin" })"><i class="fa fa-cogs">&nbsp;&nbsp;</i> Painel Administrativo</a></li>**  **</ul>**  **</li>**  **</ul>**  **}**  **else**  **{**  **<ul id="top-menu" class="nav navbar-nav navbar-right">**  **<li><a href="#" id="mu-search-icon"><i class="fa fa-search"></i></a></li>**  **<li><a href="@Url.Action("Login", "Account", new { Area = "" })"><i class="fa fa-sign-in">&nbsp;&nbsp;</i>Entrar</a></li>**  **<li><a href="@Url.Action("Register", "Account", new { Area = "" })"><i class="fa fa-user-plus">&nbsp;&nbsp;</i>Cadastrar</a></li>**  **</ul>**  **}** |

Altera o seguinte trecho de código da \_Layout.cshtml da página principal (Nâo o da Admin)

|  |
| --- |
| **<!DOCTYPE html>**  **<html>**  **<head>**  **<meta charset="utf-8" />**  **<meta name="viewport" content="width=device-width, initial-scale=1.0">**  **<title>[LearningCloud] - @ViewBag.Title</title>**  **@Styles.Render("~/Content/css")**  **@Scripts.Render("~/bundles/modernizr")**  **</head>**  **<body>**  **<div class="navbar navbar-inverse navbar-fixed-top">**  **<div class="container">**  **<div class="navbar-header">**  **<button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">**  **<span class="icon-bar"></span>**  **<span class="icon-bar"></span>**  **<span class="icon-bar"></span>**  **</button>**  **@Html.ActionLink("[LearningCloud]", "Index", "Home", new { area = "" }, new { @class = "navbar-brand" })**  **</div>**  **<div class="navbar-collapse collapse">**  **<ul class="nav navbar-nav">**  **<li>@Html.ActionLink("Home", "Index", "Home")</li>**  **<li>@Html.ActionLink("About", "About", "Home")</li>**  **<li>@Html.ActionLink("Contact", "Contact", "Home")</li>**  **</ul>**  **@Html.Partial("\_LoginPartial")**  **</div>**  **</div>**  **</div>**  **<div class="container body-content">**  **@RenderBody()**  **<hr />**  **<footer>**  **<p>&copy; @DateTime.Now.Year - Clayton Gandra</p>**  **</footer>**  **</div>**  **@Scripts.Render("~/bundles/jquery")**  **@Scripts.Render("~/bundles/bootstrap")**  **@RenderSection("scripts", required: false)**  **</body>**  **</html>** |

Na área Admin colocar o [Authorize] nas classes de controller AulaController e HomeController

**[Authorize]**

public class PanelController : Controller

{

...

**[Authorize]**

public class AulaController : Controller

{

...

Alterar o Link que vai para a Panel Admin inicial colocando o nome da aplicação e a área, colocar o @User.Identity.Name no \_layout.cshtm a da área de Admin

para exibir o username de quem está logado e vamos criar um link para voltar para a página inicial fora de Admin

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8" />  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>[LearningCloud] - @ViewBag.Title</title>  @Styles.Render("~/Content/css")  @Scripts.Render("~/bundles/modernizr")  </head>  <body>  <div class="navbar navbar-inverse navbar-fixed-top">  <div class="container">  <div class="navbar-header">  <button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">  <span class="icon-bar"></span>  <span class="icon-bar"></span>  <span class="icon-bar"></span>  </button>  **@Html.ActionLink("[LearningCloud]", "Index", "Panel", new { area = "Admin" }, new { @class = "navbar-brand" })**  </div>  <div class="navbar-collapse collapse">  **<ul class="nav navbar-nav">**  **@\*(Removido os links) \*@**  **</ul>**  <ul id="top-menu" class="nav navbar-nav navbar-right">  <li>  <a href="#" class="dropdown-toggle" data-toggle="dropdown">  <span>**<i class="fa fa-id-badge">&nbsp;&nbsp;</i>@User.Identity.Name**</span>  </a>  </li>  **<li><a href="@Url.Action("Index", "Home", new { Area = "" })"><i class="fa fa-desktop"></i> Portal de cursos</a></li>**  </ul>    </div>  </div>  </div>  <div class="container body-content">  @RenderBody()  <hr />  <footer>  <p>&copy; @DateTime.Now.Year - Clayton Gandra</p>  </footer>  </div>  @Scripts.Render("~/bundles/jquery")  @Scripts.Render("~/bundles/bootstrap")  @RenderSection("scripts", required: false)  </body>  </html> |

Vamos criar um novo Context para mapiar as classes de usuario e usuarioacesso

Infra.data pasta Context

UsuarioLearningCloudContext

definir como public herdar : DbContext

Ficando:

**using System.Data.Entity;**

**using LearningCloud.Domain.Entities;**

**using LearningCloud.Infra.Data.EntityConfig;**

**namespace LearningCloud.Infra.Data.Context**

**{**

**class UsuarioLearningCloudContext : DbContext**

**{**

**public UsuarioLearningCloudContext()**

**: base("LearningCloud")**

**{**

**}**

**public DbSet<Usuario> LearningCloud\_Usuario { get; set; }**

**public DbSet<UsuarioAcesso> LearningCloud\_UsuarioAcesso { get; set; }**

**protected override void OnModelCreating(DbModelBuilder modelBuilder)**

**{**

**modelBuilder.Configurations.Add(new UsuarioConfiguration());**

**modelBuilder.Configurations.Add(new UsuarioAcessoConfiguration());**

**base.OnModelCreating(modelBuilder);**

**}**

**}**

**}**

Criar interface de usuarioAcesso na camada de dominio em

Domain / Interfaces / Repositories

IUsuarioAcessoRepository

definir como public herdar IDisposable

using System;

using System.Collections.Generic;

using LearningCloud.Domain.Entities;

namespace LearningCloud.Domain.Interfaces.Repositories

{

public interface IUsuarioAcessoRepository : IDisposable

{

UsuarioAcesso GetAcessoByUsuarioId(string id);

string GetLoginByEmailOrUser(string login);

}

}

Vamos criar a classe concreta do repositório de usuárioAcesso na camada de Infra.Data na pasta Repositories

UsuarioAcessoRepository

definir como public herdando de : IUsuarioAcessoRepository

Implementar Interfaces da herança

**using LearningCloud.Domain.Interfaces.Repositories;**

**namespace LearningCloud.Infra.Data.Repositories**

**{**

**class UsuarioAcessoRepository : IUsuarioAcessoRepository**

**{**

**public Domain.Entities.UsuarioAcesso GetAcessoByUsuarioId(string id)**

**{**

**throw new System.NotImplementedException();**

**}**

**public string GetLoginByEmailOrUser(string login)**

**{**

**throw new System.NotImplementedException();**

**}**

**public void Dispose()**

**{**

**throw new System.NotImplementedException();**

**}**

**}**

**}**

vamos criar um construtor para injetarmos o contexto de banco de dados

ficando:

**using System.Data.Entity;**

**using Microsoft.Practices.ServiceLocation;**

using LearningCloud.Domain.Interfaces.Repositories;

**using LearningCloud.Infra.Data.Context;**

**using LearningCloud.Infra.Data.EntityFramework;**

namespace LearningCloud.Infra.Data.Repositories

{

class UsuarioAcessoRepository : IUsuarioAcessoRepository

{

**protected DbContext IdentityContextDB { get; private set; }**

**public UsuarioAcessoRepository()**

**{**

**var contextManager = ServiceLocator.Current.GetInstance<ContextManager<UsuarioLearningCloudContext>>();**

**IdentityContextDB = contextManager.GetContext;**

**}**

public Domain.Entities.UsuarioAcesso GetAcessoByUsuarioId(string id)

{

throw new System.NotImplementedException();

}

public string GetLoginByEmailOrUser(string login)

{

throw new System.NotImplementedException();

}

public void Dispose()

{

throw new System.NotImplementedException();

}

}

}

VAmos implementar as funcionalidades dos métodos

adicionar os usings

**using System.Linq;**

**using LearningCloud.Domain.Entities;**

public Domain.Entities.UsuarioAcesso GetAcessoByUsuarioId(string id)

{

**return IdentityContextDB.Set<UsuarioAcesso>().Find(id);**

}

public string GetLoginByEmailOrUser(string login)

{

**UsuarioAcesso retornoQueryUser = (from UserLogin in IdentityContextDB.Set<UsuarioAcesso>()**

**where (UserLogin.UsuarioAcesso\_Email == login || UserLogin.UsuarioAcesso\_UserName == login)**

**select UserLogin).SingleOrDefault();**

**return retornoQueryUser.UsuarioAcesso\_UserName;**

}

public void Dispose()

{

**IdentityContextDB.Dispose();**

}

classe completa:

**using System.Data.Entity;**

**using System.Linq;**

**using Microsoft.Practices.ServiceLocation;**

**using LearningCloud.Domain.Interfaces.Repositories;**

**using LearningCloud.Domain.Entities;**

**using LearningCloud.Infra.Data.Context;**

**using LearningCloud.Infra.Data.EntityFramework;**

**namespace LearningCloud.Infra.Data.Repositories**

**{**

**class UsuarioAcessoRepository : IUsuarioAcessoRepository**

**{**

**protected DbContext IdentityContextDB { get; private set; }**

**public UsuarioAcessoRepository()**

**{**

**var contextManager = ServiceLocator.Current.GetInstance<ContextManager<UsuarioLearningCloudContext>>();**

**IdentityContextDB = contextManager.GetContext;**

**}**

**public Domain.Entities.UsuarioAcesso GetAcessoByUsuarioId(string id)**

**{**

**return IdentityContextDB.Set<UsuarioAcesso>().Find(id);**

**}**

**public string GetLoginByEmailOrUser(string login)**

**{**

**UsuarioAcesso retornoQueryUser = (from UserLogin in IdentityContextDB.Set<UsuarioAcesso>()**

**where (UserLogin.uac\_email == login || UserLogin.uac\_userName == login)**

**select UserLogin).SingleOrDefault();**

**return retornoQueryUser.uac\_userName;**

**}**

**public void Dispose()**

**{**

**IdentityContextDB.Dispose();**

**}**

**}**

**}**

VAmos criar a interface de serviço para usuario acesso na camada de Domain

LearningCloud.Domain >> Interface >> Services

IUsuarioAcessoService

definir como public

criar dois métodos

UsuarioAcesso GetAcessoByUsuarioId(string id);

string GetLoginByEmailOrUser(string login);

Ficando:

**using LearningCloud.Domain.Entities;**

**namespace LearningCloud.Domain.Interfaces.Services**

**{**

**public interface IUsuarioAcessoService**

**{**

**UsuarioAcesso GetAcessoByUsuarioId(string id);**

**string GetLoginByEmailOrUser(string login);**

**}**

**}**

Vamos criar a classe concreta de serviço na camada de domain

pasta Services

LearningCloud.Domain >> Services

definir como public herdando de : IUsuarioAcessoService

implementar a interface de IUsuarioAcessoService

using LearningCloud.Domain.Interfaces.Services;

namespace LearningCloud.Domain.Services

{

public class UsuarioAcessoService : IUsuarioAcessoService

{

public Entities.UsuarioAcesso GetAcessoByUsuarioId(string id)

{

throw new System.NotImplementedException();

}

public string GetLoginByEmailOrUser(string login)

{

throw new System.NotImplementedException();

}

}

}

Vamos criar um construtor para injetar a interface de repositório

**using LearningCloud.Domain.Interfaces.Repositories;**

using LearningCloud.Domain.Interfaces.Services;

namespace LearningCloud.Domain.Services

{

public class UsuarioAcessoService : IUsuarioAcessoService

{

**private readonly IUsuarioAcessoRepository \_usuarioAcessoRepository;**

**public UsuarioAcessoService(IUsuarioAcessoRepository usuarioAcessoRepository)**

**{**

**\_usuarioAcessoRepository = usuarioAcessoRepository;**

**}**

public Entities.UsuarioAcesso GetAcessoByUsuarioId(string id)

{

throw new System.NotImplementedException();

}

public string GetLoginByEmailOrUser(string login)

{

throw new System.NotImplementedException();

}

}

}

Vamos implementar os métodos

public Entities.UsuarioAcesso GetAcessoByUsuarioId(string id)

{

**return \_usuarioAcessoRepository.GetAcessoByUsuarioId(id);**

}

public string GetLoginByEmailOrUser(string login)

{

**return \_usuarioAcessoRepository.GetLoginByEmailOrUser(login);**

}

Agora vamos criar a interface de UsuarioAcesso na camada de aplicação

LearningCloud >> Application >> Interfaces

IUsuarioAcessoAppService

definir como public

Ficando:

**using LearningCloud.Domain.Entities;**

**namespace LearningCloud.Application.Interfaces**

**{**

**public interface IUsuarioAcessoAppService**

**{**

**UsuarioAcesso GetAcessoByUsuarioId(string id);**

**string GetLoginByEmailOrUser(string login);**

**}**

**}**

Vamos criar a classe concreta de camada de aplicação na pasta Sevices

LearningCloud >> Application >> Services

definir como public herdando de : IUsuarioAcessoAppService

Implementando a interface e criando o construtor para injetar a interface de serviço de domaian

Ficando:

**using LearningCloud.Application.Interfaces;**

**using LearningCloud.Domain.Interfaces.Services;**

**namespace LearningCloud.Application.Services**

**{**

**public class UsuarioAcessoAppService : IUsuarioAcessoAppService**

**{**

**private readonly IUsuarioAcessoService \_usuarioAcessoService;**

**public UsuarioAcessoAppService(IUsuarioAcessoService usuarioAcessoService)**

**{**

**\_usuarioAcessoService = usuarioAcessoService;**

**}**

**public Domain.Entities.UsuarioAcesso GetAcessoByUsuarioId(string id)**

**{**

**return \_usuarioAcessoService.GetAcessoByUsuarioId(id);**

**}**

**public string GetLoginByEmailOrUser(string login)**

**{**

**return \_usuarioAcessoService.GetLoginByEmailOrUser(login);**

**}**

**}**

**}**

Vamos criar uma ViewModel para Login na camada de CrossCutting.Identity na pasta ViewModels/AccountViewModels

**using System.ComponentModel.DataAnnotations;**

**namespace LearningCloud.Infra.CrossCutting.Identity.ViewModels.AccountViewModels**

**{**

**public class LoginViewModel**

**{**

**[Required(ErrorMessage = "Informe seu Usuário ou E-mail.")]**

**[Display(Name = "Usuário ou Email")]**

**public string UsuarioAcesso\_UserName { get; set; }**

**[Required(ErrorMessage = "Informe sua senha.")]**

**[DataType(DataType.Password)]**

**[Display(Name = "Senha")]**

**public string UsuarioAcesso\_Password { get; set; }**

**[Display(Name = "Mantenha-me conectado.")]**

**public bool UsuarioAcesso\_RememberMe { get; set; }**

**}**

**}**

Injetar o interface de serviço de aplicação no construtor da AccountController

private ApplicationSignInManager \_signInManager;

private ApplicationUserManager \_userManager;

**private readonly IUsuarioAcessoAppService \_usuarioAcessoApp;**

public AccountController(ApplicationUserManager userManager, ApplicationSignInManager signInManager, **IUsuarioAcessoAppService usuarioAcessoApp**)

{

\_userManager = userManager;

\_signInManager = signInManager;

**\_usuarioAcessoApp = usuarioAcessoApp;**

}

criar ActionResult para login e

**using Microsoft.AspNet.Identity.Owin;**

**// GET: /Account/Login**

**[AllowAnonymous]**

**[Route("Entrar")]**

**public ActionResult Login(string returnUrl)**

**{**

**ViewBag.ReturnUrl = returnUrl;**

**if (User.Identity.IsAuthenticated)**

**{**

**return RedirectToAction("Index", "Home");**

**}**

**return View();**

**}**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**// POST: /Account/Login**

**[HttpPost]**

**[AllowAnonymous]**

**[Route("Entrar")]**

**[ValidateAntiForgeryToken]**

**public async Task<ActionResult> Login(LoginViewModel model, string returnUrl)**

**{**

**if (!ModelState.IsValid)**

**{**

**return View(model);**

**}**

**string userName = \_usuarioAcessoApp.GetLoginByEmailOrUser(model.UserName);**

**if (userName != null)**

**{**

**model.UsuarioAcesso\_UserName = userName;**

**//////var user = await \_userManager.FindByNameAsync(userName);**

**//////if (user != null)**

**//////{**

**////// if (!await \_userManager.IsEmailConfirmedAsync(user.Id))**

**////// {**

**////// ViewBag.StatusMessage = "Você deve confirmar seu e-mail para poder entrar.";**

**////// return View("Error");**

**////// }**

**//////}**

**}**

**// This doesn't count login failures towards account lockout**

**// To enable password failures to trigger account lockout, change to shouldLockout: true**

**var result = await \_signInManager.PasswordSignInAsync(model.UserName, model.Password, model.RememberMe, shouldLockout: true);**

**switch (result)**

**{**

**case SignInStatus.Success:**

**return RedirectToLocal(returnUrl);**

**case SignInStatus.LockedOut:**

**return View("Lockout");**

**case SignInStatus.RequiresVerification:**

**return RedirectToAction("SendCode", new { ReturnUrl = returnUrl, RememberMe = model.RememberMe });**

**case SignInStatus.Failure:**

**default:**

**ModelState.AddModelError("", "Login ou Senha incorretos.");**

**model.UsuarioAcesso\_Password = null;**

**return View(model);**

**}**

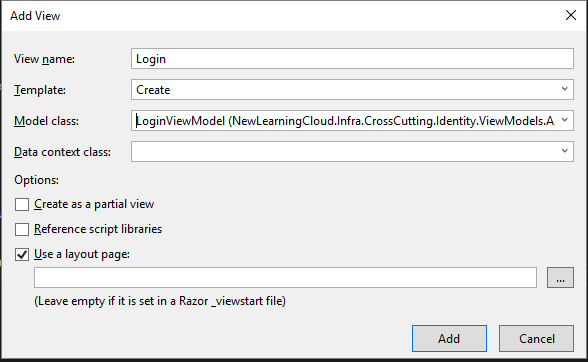
**}**

registrar interface e classes para usuario acesso no IoC

**container.Register<IUsuarioAcessoRepository, UsuarioAcessoRepository>(Lifestyle.Scoped);**

**container.Register<IUsuarioAcessoAppService, UsuarioAcessoAppService>();**

**container.Register<IUsuarioAcessoService, UsuarioAcessoService>();**



Adiciona o using na view

@using **LearningCloud.Infra.CrossCutting.Identity.ViewModels.AccountViewModels**

alterar html de view de:

**<h2>Login</h2>**

@using (Html.BeginForm())

{

@Html.AntiForgeryToken()

**<div class="form-horizontal">**

<h4>LoginViewModel</h4>

<hr />

…

<div class="form-group">

<div class="col-md-offset-2 col-md-10">

<input type="submit" value="Create" class="btn btn-default" />

</div>

</div>

**</div>**

}

Para:

**<div class="row">**

**<div class="col-md-5">**

**<section id="loginForm">**

@using (Html.BeginForm(**"Login", "Account", new { ReturnUrl = ViewBag.ReturnUrl }, FormMethod.Post, new { @class = "form-horizontal", role = "form" })**)

{

@Html.AntiForgeryToken()

<h4>**Utilize uma conta local para login.**</h4>

<hr />

…

<div class="form-group">

<div class="col-md-offset-4 col-md-8">

<input type="submit" value="**Login**" class="btn btn-default" />

</div>

</div>

**<p>**

**@Html.ActionLink("Registrar uma nova conta?", "Register")**

**</p>**

**<p>**

**@Html.ActionLink("Esqueceu sua senha?", "ForgotPassword")**

**</p>**

**}**

**</section>**

**</div>**

**<div class="col-md-1">**

**</div>**

**<div class="col-md-5">**

**<section id="socialLoginForm">**

**@\*@Html.Partial("\_ExternalLoginsListPartial", new ExternalLoginListViewModel { ReturnUrl = ViewBag.ReturnUrl })\*@**

**</section>**

**</div>**

**</div>**

adicionar a model ExternalLoginListViewModel na camada de CrossCutting.Identity pasta AccountViewModels

namespace LearningCloud.Infra.CrossCutting.Identity.Models.AccountViewModels

{

public class ExternalLoginListViewModel

{

public string ReturnUrl { get; set; }

}

}

adicionar o arquivo

**\_ExternalLoginsListPartial.cshtml na pasta Views / Account**

@model LearningCloud.Infra.CrossCutting.Identity.ViewModels.AccountViewModels.ExternalLoginListViewModel

@using Microsoft.Owin.Security

<h4>Ou utilize outro serviço para fazer o login:</h4>

<hr />

@{

var loginProviders = Context.GetOwinContext().Authentication.GetExternalAuthenticationTypes();

if (loginProviders.Count() == 0) {

<div>

<p>

There are no external authentication services configured. See <a href="http://go.microsoft.com/fwlink/?LinkId=403804">this article</a>

for details on setting up this ASP.NET application to support logging in via external services.

</p>

</div>

}

else {

using (Html.BeginForm("ExternalLogin", "Account", new { ReturnUrl = Model.ReturnUrl })) {

@Html.AntiForgeryToken()

<div id="socialLoginList">

<p>

@foreach (AuthenticationDescription p in loginProviders)

{

<button type="submit" class="btn btn-default" id="@p.AuthenticationType" name="provider" value="@p.AuthenticationType" title="Entrar utilizando sua conta do @p.Caption">@p.AuthenticationType</button>

}

</p>

</div>

}

}

}

Tirar comentario na view Login,cshtml

@Html.Partial("\_ExternalLoginsListPartial", new ExternalLoginListViewModel { ReturnUrl = ViewBag.ReturnUrl })

login com facebook

Install-Package Microsoft.Owin.Security.Facebook

Alterar a Startup.AUth.cs

public void ConfigureAuth(IAppBuilder app)

{

...

// Enables the application to remember the second login verification factor such as phone or email.

// Once you check this option, your second step of verification during the login process will be remembered on the device where you logged in from.

// This is similar to the RememberMe option when you log in.

app.UseTwoFactorRememberBrowserCookie(DefaultAuthenticationTypes.TwoFactorRememberBrowserCookie);

**const string xmlSchemaString = "http://www.w3.org/2001/XMLSchema#string";**

**string facebookAppId = "217851345348916";**

**string facebookAppSecret = "3a4369eecd8d978baa476a1fb9e964c5";**

**var facebookOptions = new FacebookAuthenticationOptions();**

**facebookOptions.AppId = facebookAppId;**

**facebookOptions.AppSecret = facebookAppSecret;**

**facebookOptions.Scope.Add("email");**

**//facebookOptions.Scope.Add("user\_birthday");**

**facebookOptions.BackchannelHttpHandler = new FacebookBackChannelHandler();**

**facebookOptions.UserInformationEndpoint = "https://graph.facebook.com/v2.7/me?fields=id,name,email,first\_name,last\_name";**

**facebookOptions.Provider = new FacebookAuthenticationProvider()**

**{**

**OnAuthenticated = async facebookContext =>**

**{**

**// Save every additional claim we can find in the user**

**foreach (JProperty property in facebookContext.User.Children())**

**{**

**var claimType = string.Format("urn:facebook:{0}", property.Name);**

**string claimValue = (string)property.Value;**

**if (!facebookContext.Identity.HasClaim(claimType, claimValue))**

**facebookContext.Identity.AddClaim(new Claim(claimType, claimValue, xmlSchemaString, "External"));**

**}**

**}**

**};**

**facebookOptions.SignInAsAuthenticationType = DefaultAuthenticationTypes.ExternalCookie;**

**app.UseFacebookAuthentication(facebookOptions);**

**}**

**public class FacebookBackChannelHandler : System.Net.Http.HttpClientHandler**

**{**

**protected override async System.Threading.Tasks.Task<System.Net.Http.HttpResponseMessage> SendAsync(System.Net.Http.HttpRequestMessage request, System.Threading.CancellationToken cancellationToken)**

**{**

**// Replace the RequestUri so it's not malformed**

**if (!request.RequestUri.AbsolutePath.Contains("/oauth"))**

**{**

**request.RequestUri = new Uri(request.RequestUri.AbsoluteUri.Replace("?access\_token", "&access\_token"));**

**}**

**return await base.SendAsync(request, cancellationToken);**

**}**

**}**

}

}

Usiing

**using Microsoft.Owin.Security.Facebook;**

**using Newtonsoft.Json.Linq;**

**using System.Security.Claims;**

chave e id do facebook

Login com Google

Install-Package Microsoft.Owin.Security.Google

Alterar a Startup.AUth.cs

public void ConfigureAuth(IAppBuilder app)

{

…

facebookOptions.Provider = new FacebookAuthenticationProvider()

{

OnAuthenticated = async facebookContext =>

{

// Save every additional claim we can find in the user

foreach (JProperty property in facebookContext.User.Children())

{

var claimType = string.Format(FacebookClaimBaseName, property.Name);

string claimValue = (string)property.Value;

if (!facebookContext.Identity.HasClaim(claimType, claimValue))

facebookContext.Identity.AddClaim(new Claim(claimType, claimValue, xmlSchemaString, "External"));

}

}

};

facebookOptions.SignInAsAuthenticationType = DefaultAuthenticationTypes.ExternalCookie;

app.UseFacebookAuthentication(facebookOptions);

**string googleClientId = "359134327838-37rl59j06tjpc29ltbsrmpem66sk809a.apps.googleusercontent.com";**

**string googleClientSecret = "9TM5TEiUS6e2Egjqh5deumdp";**

**var googleAuthenticationOptions = new GoogleOAuth2AuthenticationOptions**

**{**

**ClientId = googleClientId,**

**ClientSecret = googleClientSecret,**

**Provider = new GoogleOAuth2AuthenticationProvider()**

**{**

**//OnAuthenticated = async googleContext =>**

**// {**

**// // string profileClaimName = string.Format("urn:google:{0}", "profile");**

**// foreach (JProperty property in googleContext.User.Children())**

**// {**

**// var claimType = string.Format("urn:google:{0}", property.Name);**

**// string claimValue = (string)property.Value;**

**// if (!googleContext.Identity.HasClaim(claimType, claimValue))**

**// googleContext.Identity.AddClaim(new Claim(claimType, claimValue,**

**// xmlSchemaString, "External"));**

**// }**

**// }**

**}**

**};**

**googleAuthenticationOptions.Scope.Add("https://www.googleapis.com/auth/plus.login");**

**googleAuthenticationOptions.Scope.Add("https://www.googleapis.com/auth/userinfo.email");**

**app.UseGoogleAuthentication(googleAuthenticationOptions);**

Using

**using Microsoft.Owin.Security.Google;**

adicionar a action ExternalLogin na AccountController

**// POST: /Account/ExternalLogin**

**[HttpPost]**

**[AllowAnonymous]**

**[Route("Externa/Entrar")]**

**[ValidateAntiForgeryToken]**

**public ActionResult ExternalLogin(string provider, string returnUrl)**

**{**

**// Request a redirect to the external login provider**

**return new ChallengeResult(provider, Url.Action("ExternalLoginCallback", "Account", new { ReturnUrl = returnUrl }));**

**}**

adicionar na #region Helpers a interna class ChallengeResult : HttpUnauthorizedResult

**internal class ChallengeResult : HttpUnauthorizedResult**

**{**

**public ChallengeResult(string provider, string redirectUri)**

**: this(provider, redirectUri, null)**

**{**

**}**

**public ChallengeResult(string provider, string redirectUri, string userId)**

**{**

**LoginProvider = provider;**

**RedirectUri = redirectUri;**

**UserId = userId;**

**}**

**public string LoginProvider { get; set; }**

**public string RedirectUri { get; set; }**

**public string UserId { get; set; }**

**public override void ExecuteResult(ControllerContext context)**

**{**

**var properties = new AuthenticationProperties { RedirectUri = RedirectUri };**

**if (UserId != null)**

**{**

**properties.Dictionary[XsrfKey] = UserId;**

**}**

**context.HttpContext.GetOwinContext().Authentication.Challenge(properties, LoginProvider);**

**}**

**}**

adicionar a action ExternalLoginCallback na AccountController

**// GET: /Account/ExternalLoginCallback**

**[AllowAnonymous]**

**public async Task<ActionResult> ExternalLoginCallback(string returnUrl)**

**{**

**var loginInfo = await AuthenticationManager.GetExternalLoginInfoAsync();**

**if (loginInfo == null)**

**{**

**return RedirectToAction("Login");**

**}**

**// Sign in the user with this external login provider if the user already has a login**

**var result = await \_signInManager.ExternalSignInAsync(loginInfo, isPersistent: false);**

**switch (result)**

**{**

**case SignInStatus.Success:**

**return RedirectToLocal(returnUrl);**

**case SignInStatus.LockedOut:**

**return View("Lockout");**

**case SignInStatus.RequiresVerification:**

**return RedirectToAction("SendCode", new { ReturnUrl = returnUrl, RememberMe = false });**

**case SignInStatus.Failure:**

**default:**

**// Se ele nao tem uma conta solicite que crie uma**

**ViewBag.ReturnUrl = returnUrl;**

**ViewBag.LoginProvider = loginInfo.Login.LoginProvider;**

**//get firstname, lastname and email address from login provider**

**var providerKey = loginInfo.Login.ProviderKey;**

**var loginProvider = loginInfo.Login.LoginProvider;**

**var addtionalDetail = await ExternalClaimsManager.GetAddtionalLoginDetailsAsync(**

**loginProvider,**

**AuthenticationManager.GetExternalIdentityAsync(DefaultAuthenticationTypes.ExternalCookie)**

**);**

**//// Pegar a informação do login externo.**

**var info = await AuthenticationManager.GetExternalLoginInfoAsync();**

**if (info == null)**

**{**

**return View("ExternalLoginFailure");**

**}**

**var user = new ApplicationUser**

**{**

**UserName = loginInfo.DefaultUserName,**

**Email = loginInfo.Email,**

**UsuarioAcesso\_Nivel = 50,**

**EmailConfirmed = true,**

**UsuarioAcesso\_Usuario = new Domain.Entities.Usuario**

**{**

**Usuario\_Nome = addtionalDetail.FirstName,**

**Usuario\_SobreNome = addtionalDetail.LastName,**

**Usuario\_DataCadastro = DateTime.Now.ToUniversalTime(),**

**Usuario\_Status = "A"**

**}**

**};**

**//\_userManager.UserValidator = new CustomUserValidator<ApplicationUser>(\_userManager);**

**//if (String.IsNullOrEmpty(user.UserName))**

**//{**

**// ModelState.AddModelError("UserName", "Preencha o campo Usuário.");**

**//}**

**//else**

**//{**

**var existingAccount = await \_userManager.FindByNameAsync(user.UserName);**

**if (existingAccount != null && existingAccount.Id != user.Id)**

**ModelState.AddModelError("UsuarioAcesso\_UserName", "O Usuário '" + user.UserName + "' já existe.");**

**var existingEmail = await \_userManager.FindByEmailAsync(user.Email);**

**if (existingEmail != null && existingEmail.Id != user.Id)**

**{**

**ModelState.AddModelError("UsuarioAcesso\_Email", "O Email '" + user.Email + "' já existe.");**

**user.EmailConfirmed = false;**

**}**

**//}**

**if (ModelState.IsValid)**

**{**

**var resultCreate = await \_userManager.CreateAsync(user);**

**if (resultCreate.Succeeded)**

**{**

**resultCreate = await \_userManager.AddLoginAsync(user.Id, info.Login);**

**if (resultCreate.Succeeded)**

**{**

**await \_signInManager.SignInAsync(user, isPersistent: false, rememberBrowser: false);**

**return RedirectToLocal(returnUrl);**

**}**

**}**

**AddErrors(resultCreate);**

**}**

**return View("ExternalLoginConfirmation", new ExternalLoginConfirmationViewModel { UsuarioAcesso\_UserName = user.UserName, Usuario\_Nome = user.UsuarioAcesso\_Usuario.Usuario\_Nome, Usuario\_SobreNome = user.UsuarioAcesso\_Usuario.Usuario\_SobreNome, UsuarioAcesso\_Email = user.Email });**

**}**

**}**

criar classe ExternalClaimsManager na pasta Configuration

LearningCloud.Infra.CrossCutting.Identity

**using System;**

**using System.Linq;**

**using System.Collections.Generic;**

**using System.Security.Claims;**

**using System.Threading.Tasks;**

**namespace NewLearningCloud.Infra.CrossCutting.Identity.Configuration**

**{**

**public static class ExternalClaimsManager**

**{**

**static Dictionary<string, ExternalClaimReader> \_externalPrvdrMngrs = new Dictionary<string, ExternalClaimReader>()**

**{**

**{"Facebook", new FacebookExternalClaimReader()},**

**{"Google", new GoogleExternalClaimReader()},**

**{"Twitter", new TwitterExternalClaimReader()},**

**{"Microsoft", new MicrosoftExternalClaimReader()}**

**};**

**static ExternalClaimsManager()**

**{**

**}**

**public class AddtionalLoginDetail**

**{**

**public string FirstName { get; set; }**

**public string LastName { get; set; }**

**public string EmailAddress { get; set; }**

**public AddtionalLoginDetail()**

**{**

**}**

**}**

**public async static Task<AddtionalLoginDetail> GetAddtionalLoginDetailsAsync(string loginProvider, Task<ClaimsIdentity> claimsIdentityTask)**

**{**

**return await \_externalPrvdrMngrs[loginProvider].GetAddtionalLoginDetailsAsync(claimsIdentityTask);**

**}**

**private abstract class ExternalClaimReader**

**{**

**public abstract Task<AddtionalLoginDetail> GetAddtionalLoginDetailsAsync(Task<ClaimsIdentity> claimsIdentityTask);**

**protected string GetClaimValue(IEnumerable<Claim> claims, string claimTypeName)**

**{**

**if (claims == null) throw new ArgumentNullException("claims");**

**if (claimTypeName == null) throw new ArgumentNullException("claimTypeName");**

**var value = String.Empty;**

**try**

**{**

**var claim = claims.First(x => x.Type.Contains(claimTypeName));**

**if (claim != null)**

**value = claim.Value;**

**}**

**catch (InvalidOperationException) { }**

**return value;**

**}**

**}**

**private class FacebookExternalClaimReader : ExternalClaimReader**

**{**

**public override async Task<AddtionalLoginDetail> GetAddtionalLoginDetailsAsync(Task<ClaimsIdentity> claimsIdentityTask)**

**{**

**return await Task.Run(() =>**

**{**

**var claims = claimsIdentityTask.Result.Claims;**

**var ald = new AddtionalLoginDetail**

**{**

**FirstName = GetClaimValue(claims, "first\_name"),// firstName,**

**LastName = GetClaimValue(claims, "last\_name"),//lastName,**

**EmailAddress = GetClaimValue(claims, "emailaddress")**

**};**

**return ald;**

**}**

**);**

**}**

**}**

**private class GoogleExternalClaimReader : ExternalClaimReader**

**{**

**public override async Task<AddtionalLoginDetail> GetAddtionalLoginDetailsAsync(Task<ClaimsIdentity> claimsIdentityTask)**

**{**

**return await Task.Run(() =>**

**{**

**var claims = claimsIdentityTask.Result.Claims;**

**var ald = new AddtionalLoginDetail**

**{**

**EmailAddress = GetClaimValue(claims, ClaimTypes.Email),**

**FirstName = GetClaimValue(claims, ClaimTypes.GivenName),**

**LastName = GetClaimValue(claims, ClaimTypes.Surname)**

**};**

**return ald;**

**}**

**);**

**}**

**}**

**private class TwitterExternalClaimReader : ExternalClaimReader**

**{**

**public override async Task<AddtionalLoginDetail> GetAddtionalLoginDetailsAsync(Task<ClaimsIdentity> claimsIdentityTask)**

**{**

**return await Task.Run(() =>**

**{**

**var claims = claimsIdentityTask.Result.Claims;**

**var ald = new AddtionalLoginDetail**

**{**

**EmailAddress = GetClaimValue(claims, "urn:twitter:name")**

**};**

**return ald;**

**}**

**);**

**}**

**}**

**private class MicrosoftExternalClaimReader : ExternalClaimReader**

**{**

**public override async Task<AddtionalLoginDetail> GetAddtionalLoginDetailsAsync(Task<ClaimsIdentity> claimsIdentityTask)**

**{**

**return await Task.Run(() =>**

**{**

**var claims = claimsIdentityTask.Result.Claims;**

**var firstName = String.Empty;**

**var lastName = String.Empty;**

**var name = GetClaimValue(claims, "urn:microsoft:name");**

**if (!String.IsNullOrEmpty(name))**

**{**

**var names = name.Split(' ');**

**firstName = names[0];**

**lastName = String.Join(" ", names.Skip(1));**

**}**

**var ald = new AddtionalLoginDetail**

**{**

**EmailAddress = GetClaimValue(claims, "emailaddress"),**

**FirstName = firstName,**

**LastName = lastName**

**};**

**var birthdayString = GetClaimValue(claims, "birthday");**

**if (!String.IsNullOrEmpty(birthdayString))**

**{**

**var birthday = DateTime.ParseExact(((string)birthdayString.ToString()), "mm/dd/yyyy", null);**

**//ald.DobDay = birthday.Day;**

**//ald.DobMonth = birthday.Month;**

**}**

**return ald;**

**}**

**);**

**}**

**}**

**}**

**}**

adicionar as actions na AccountController

ExternalLoginConfirmation

ExternalLoginFailure

**// POST: /Account/ExternalLoginConfirmation**

**[HttpPost]**

**[AllowAnonymous]**

**[ValidateAntiForgeryToken]**

**public async Task<ActionResult> ExternalLoginConfirmation(ExternalLoginConfirmationViewModel model, string returnUrl)**

**{**

**if (User.Identity.IsAuthenticated)**

**{**

**return RedirectToAction("Index", "Manage");**

**}**

**if (ModelState.IsValid)**

**{**

**// Pegar a informação do login externo.**

**var info = await AuthenticationManager.GetExternalLoginInfoAsync();**

**if (info == null)**

**{**

**return View("ExternalLoginFailure");**

**}**

**ViewBag.ReturnUrl = returnUrl;**

**ViewBag.LoginProvider = info.Login.LoginProvider;**

**var user = new ApplicationUser**

**{**

**UserName = model.UsuarioAcesso\_UserName,**

**Email = model.UsuarioAcesso\_Email,**

**UsuarioAcesso\_Nivel = 50,**

**UsuarioAcesso\_Usuario = new Domain.Entities.Usuario**

**{**

**Usuario\_Nome = model.Usuario\_Nome,**

**Usuario\_SobreNome = model.Usuario\_SobreNome,**

**Usuario\_DataCadastro = DateTime.Now.ToUniversalTime(),**

**Usuario\_Status = "A"**

**}**

**};**

**var existingAccount = await \_userManager.FindByNameAsync(user.UserName);**

**if (existingAccount != null && existingAccount.Id != user.Id)**

**ModelState.AddModelError("UsuarioAcesso\_UserName", "O Usuário '" + user.UserName + "' já existe.");**

**var existingEmail = await \_userManager.FindByEmailAsync(user.Email);**

**if (existingEmail != null && existingEmail.Id != user.Id)**

**{**

**ModelState.AddModelError("UsuarioAcesso\_Email", "O Email '" + user.Email + "' já existe.");**

**user.EmailConfirmed = false;**

**}**

**if (ModelState.IsValid)**

**{**

**var result = await \_userManager.CreateAsync(user);**

**if (result.Succeeded)**

**{**

**result = await \_userManager.AddLoginAsync(user.Id, info.Login);**

**if (result.Succeeded)**

**{**

**await \_signInManager.SignInAsync(user, isPersistent: false, rememberBrowser: false);**

**return RedirectToLocal(returnUrl);**

**}**

**}**

**AddErrors(result);**

**}**

**}**

**ViewBag.ReturnUrl = returnUrl;**

**return View(model);**

**}**

**//**

**// GET: /Account/ExternalLoginFailure**

**[AllowAnonymous]**

**public ActionResult ExternalLoginFailure()**

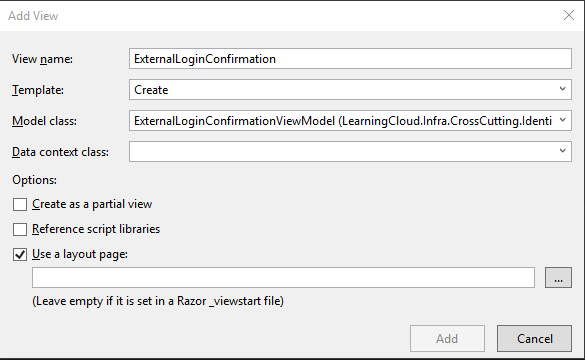
**{**

**return View();**

**}**

adiciona a view ExternalLoginConfirmation

ExternalLoginConfirmation Tempalte Create Model class: (ExternalLoginConfirmationViewModel)



altera o @using (Html.BeginForm(

**@using (Html.BeginForm()**

para:

**@using (Html.BeginForm("ExternalLoginConfirmation", "Account", new { ReturnUrl = ViewBag.ReturnUrl }, FormMethod.Post, new { @class = "form-horizontal", role = "form" }))**

**{**

**@Html.AntiForgeryToken()**

Adicionar a viewModel ExternalLoginConfirmationViewModel na camada de CrossCutting.Identity pasta Models / AccountViewModels

definir como public

**using System.ComponentModel.DataAnnotations;**

**namespace LearningCloud.Infra.CrossCutting.Identity.ViewModels.AccountViewModels**

**{**

**public class ExternalLoginConfirmationViewModel**

**{**

**[Required(ErrorMessage = "Preencha o campo Usuário.")]**

**[StringLength(100, ErrorMessage = "O {0} deve conter pelo menos {2} caracteres.", MinimumLength = 6)]**

**[Display(Name = "Usuário")]**

**public string UsuarioAcesso\_UserName { get; set; }**

**[Required(ErrorMessage = "Preencha o campo Nome.")]**

**[StringLength(100, ErrorMessage = "O {0} deve conter pelo menos {2} caracteres.", MinimumLength = 3)]**

**[Display(Name = "Nome")]**

**public string Usuario\_Nome { get; set; }**

**[Required(ErrorMessage = "Preencha o campo Sobrenome.")]**

**[Display(Name = "Sobrenome")]**

**public string Usuario\_SobreNome { get; set; }**

**[Required(ErrorMessage = "Preencha o campo Email.")]**

**[EmailAddress]**

**[Display(Name = "E-mail")]**

**public string UsuarioAcesso\_Email { get; set; }**

**}**

**}**

criar Views

ExternalLoginFailure (Empty)

adiciona o código

**@{**

**ViewBag.Title = "Falha no Login";**

**}**

**<div class="container">**

**<div class="row">**

**<div class="col-lg-12 col-md-12">**

**<section class="content-header">**

**<h1> @ViewBag.Title <small></small> </h1>**

**<ol class="breadcrumb">**

**<li><a href="#"><i class="fa fa-dashboard"></i> Level</a></li>**

**<li class="active">Here</li>**

**</ol>**

**</section>**

**<section class="content">**

**<div class="box box-default">**

**<div class="box-header with-border">**

**<h3 class="box-title">zzzzzzzzz</h3>**

**</div>**

**<div class="box-body">**

**<hgroup>**

**<h2>@ViewBag.Title.</h2>**

**<h3 class="text-danger">Falha ao tentar realizar o login com o serviço selecionado.</h3>**

**</hgroup>**

**</div><!-- /.box-body -->**

**</div><!-- /.box box-default -->**

**</section><!-- /.content -->**

**</div><!-- /.col-lg-12 col-md-12 -->**

**</div><!-- /.row -->**

**</div><!-- /.container -->**

criar Views Lockout(Empty)

adiciona o código

|  |
| --- |
| **@{**  **ViewBag.Title = "Conta bloqueada";**  **}**  **<hgroup>**  **<h1 class="text-danger">@ViewBag.Title.</h1>**  **<h2 class="text-danger">Esta conta foi bloqueada, tente novamente em alguns minutos.</h2>**  **</hgroup>** |

Alterar a view Error

|  |
| --- |
| **@{**  **ViewBag.Title = "Algo de errado não está certo.";**  **if (string.IsNullOrEmpty(ViewBag.StatusMessage))**  **{**  **ViewBag.StatusMessage = "Ocorreu uma exceção ao processar sua solicitação.";**  **}**  **}**  **<div class="row">**  **<div class="col-md-2 col-md-offset-1" style="padding:50px 0 50px 50px; border:1px none #ff0000">**  **<img src="~/Content/Images/img\_icons/icons8-Tornado-100.png" width="100" height="100" />**  **</div>**  **<div class="col-md-9" style="border:1px none #4cff00">**  **<hr />**  **<h1>Oops! Erro</h1>**  **<h3>Algo de errado não está certo.</h3>**  **<h5 class="text-danger">@Html.Raw(@ViewBag.StatusMessage)</h5>**  **@if (TempData["CallbackError"] != null)**  **{**  **List<string> callbackError = (List<string>)TempData["CallbackError"];**  **<ul>**  **@foreach (var error in callbackError)**  **{**  **<li>@error</li>**  **}**  **</ul>**  **}**  **</div>**  **</div>** |

Vamos alterar a ApplicationUser para exibir o nome do usuário no lugar do user name.

criado Claims

public class ApplicationUser : IdentityUser

{

public async Task<ClaimsIdentity> GenerateUserIdentityAsync(UserManager<ApplicationUser> manager)

{

// Note the authenticationType must match the one defined in CookieAuthenticationOptions.AuthenticationType

var userIdentity = await manager.CreateIdentityAsync(this, DefaultAuthenticationTypes.ApplicationCookie);

// Add custom user claims here

**//userIdentity.AddClaim(new Claim("FullName", UsuarioAcesso\_Usuario.Usuario\_Nome + " " + UsuarioAcesso\_Usuario.Usuario\_SobreNome));**

**userIdentity.AddClaims(new[] {**

**new Claim("FullName", UsuarioAcesso\_Usuario.Usuario\_Nome + " " + UsuarioAcesso\_Usuario.Usuario\_SobreNome),**

**new Claim("FirstName",UsuarioAcesso\_Usuario.Usuario\_Nome),**

**new Claim("Surname", UsuarioAcesso\_Usuario.Usuario\_SobreNome)**

**});**

return userIdentity;

}

public int UsuarioAcesso\_Nivel { get; set; }

public virtual Usuario UsuarioAcesso\_Usuario { get; set; }

}

Na pasta Configuration da camada de Identity vamos criar um arquivo chamado IdentityExtensions

**LearningCloud.Infra.CrossCutting.Identity.Configuration**

**using System.Security.Claims;**

**using System.Security.Principal;**

**namespace NewLearningCloud.Infra.CrossCutting.Identity.Configuration**

**{**

**public static class IdentityExtensions**

**{**

**public static string GetInfoUser(this IPrincipal user, string claimName)**

**{**

**if (user.Identity.IsAuthenticated)**

**{**

**var claimsIdentity = user.Identity as ClaimsIdentity;**

**if (claimsIdentity != null)**

**{**

**foreach (var claim in claimsIdentity.Claims)**

**{**

**if (claim.Type == claimName)**

**return claim.Value;**

**}**

**}**

**return "";**

**}**

**else**

**return "";**

**}**

**}**

**}**

nas Views \_LoginPartial e \_Layout da Admin trocar

**@User.Identity.Name**

para

**@User.GetInfoUser("FullName")**

depois no Account cria ForgotPassword, ForgotPasswordConfirmation e criar a ForgotPasswordViewModel

ForgotPassword

**// GET: /Account/ForgotPassword**

**[AllowAnonymous]**

**[Route("Senha/Recuperar")]**

**public ActionResult ForgotPassword()**

**{**

**return View();**

**}**

ForgotPassword Post

|  |
| --- |
| **// POST: /Account/ForgotPassword**  **[HttpPost]**  **[AllowAnonymous]**  **[Route("Senha/Recuperar")]**  **[ValidateAntiForgeryToken]**  **public async Task<ActionResult> ForgotPassword(ForgotPasswordViewModel model)**  **{**  **if (ModelState.IsValid)**  **{**  **ApplicationUser user = await \_userManager.FindByEmailAsync(model.UsuarioAcesso\_Email);**  **if (user == null)**  **{**  **ViewBag.Status = -1;**  **ViewBag.Message = "Não existe cadastro com o e-mail informado.";**  **return View("ForgotPasswordConfirmation");**  **}**  **if (string.IsNullOrEmpty(user.PasswordHash))**  **{**  **IList<UserLoginInfo> userLogins = await \_userManager.GetLoginsAsync(user.Id);**  **if (userLogins.Count > 0)**  **{**  **model.CurrentLogins = userLogins;**  **}**  **ViewBag.Status = -1;**  **ViewBag.Message = "A conta solicitada não possui senha.";**  **return View("ForgotPasswordConfirmation", model);**  **}**    **//////if (user == null || !(await \_userManager.IsEmailConfirmedAsync(user.Id)))**  **//////{**  **////// // Não revelar se o usuario nao existe ou nao esta confirmado**  **////// return View("ForgotPasswordConfirmation");**  **//////}**  **string code = await \_userManager.GeneratePasswordResetTokenAsync(user.Id);**  **string callbackUrl = Url.Action("ResetPassword", "Account", new { userId = user.Id, code = code }, protocol: Request.Url.Scheme);**  **await \_userManager.SendEmailAsync(user.Id, "Esqueci minha senha", "Por favor altere sua senha clicando aqui: <a href='" + callbackUrl + "'></a>");**  **ViewBag.Link = callbackUrl;**  **ViewBag.Status = 0;**  **ViewBag.Message = "DEMO: Caso o link não chegue: ";**  **ViewBag.LinkAcesso = callbackUrl;**  **return View("ForgotPasswordConfirmation");**  **}**  **// No caso de falha, reexibir a view.**  **return View(model);**  **}** |

CRiar na pasta ViewModels/AccountViewModels da camada Infra.CrossCutting.Identity a classe ForgotPasswordViewModel

|  |
| --- |
| **using System.ComponentModel.DataAnnotations;**  **namespace LearningCloud.Infra.CrossCutting.Identity.ViewModels.AccountViewModels**  **{**  **public class ForgotPasswordViewModel**  **{**  **[Required(ErrorMessage = "Preencha o campo Email.")]**  **[EmailAddress]**  **[Display(Name = "E-mail")]**  **public string UsuarioAcesso\_Email { get; set; }**  **[ScaffoldColumn(false)]**  **public IList<UserLoginInfo> CurrentLogins { get; set; }**  **}**  **}** |

definir como public

criar a View ForgotPassword (Empty com model LearningCloud.Infra.CrossCutting.Identity.ViewModels.AccountViewModels.ForgotPasswordViewModel)

|  |
| --- |
| **@model LearningCloud.Infra.CrossCutting.Identity.ViewModels.AccountViewModels.ForgotPasswordViewModel**  **@{**  **ViewBag.Title = "Esqueceu sua senha?";**  **}**  **<h2>@ViewBag.Title.</h2>**  **@using (Html.BeginForm("ForgotPassword", "Account", FormMethod.Post, new { @class = "form-horizontal", role = "form" }))**  **{**  **@Html.AntiForgeryToken()**  **<h4>Enter your email.</h4>**  **<hr />**  **@Html.ValidationSummary("", new { @class = "text-danger" })**  **<div class="form-group">**  **@Html.LabelFor(m => m.UsuarioAcesso\_Email, new { @class = "col-md-2 control-label" })**  **<div class="col-md-10">**  **@Html.TextBoxFor(m => m.UsuarioAcesso\_Email, new { @class = "form-control" })**  **</div>**  **</div>**  **<div class="form-group">**  **<div class="col-md-offset-2 col-md-10">**  **<input type="submit" class="btn btn-default" value="Enviar" />**  **</div>**  **</div>**  **}**  **@section Scripts {**  **@Scripts.Render("~/bundles/jqueryval")**  **}** |

Criar a action ForgotPasswordConfirmation

|  |
| --- |
| **// GET: /Account/ForgotPasswordConfirmation**  **[AllowAnonymous]**  **[Route("Senha/Recuperar/Confirmar")]**  **public ActionResult ForgotPasswordConfirmation()**  **{**  **return View();**  **}** |

Criar a View ForgotPasswordConfirmation (Empty (without model))

|  |
| --- |
| **@model LearningCloud.Infra.CrossCutting.Identity.ViewModels.AccountViewModels.ForgotPasswordViewModel**  **@{**  **ViewBag.Title = "Confirmação de esquecimento de senha";**  **}**  **<hgroup class="title">**  **<h1>@ViewBag.Title.</h1>**  **</hgroup>**  **<div>**    **@if (@ViewBag.Status < 0)**  **{**  **<p class="text-danger">**  **<h4>@ViewBag.Message</h4>**  **</p>**  **if (Model !=null && Model.CurrentLogins.Count > 0)**  **{**  **<p class="text-danger">**  **A conta está associada ao(s) seguinte(s) provedores:**  **</p>**  **foreach (var account in Model.CurrentLogins)**  **{**  **<h3><span class="label label-default">@account.LoginProvider</span></h3>**  **}**  **}**  **}**  **else**  **{**  **<p>**  **Por favor, verifique seu e-mail para trocar sua senha.**  **</p>**  **<p class="text-danger">**  **<h4>@ViewBag.Message <a href="@ViewBag.LinkAcesso">Link de acesso</a></h4>**  **</p>**  **}**  **</div>** |

Criara as action ResetPassword e ResetPasswordConfirmation e a viewmodel ResetPasswordViewModel

ActionResult ResetPassword Get

|  |
| --- |
| **// GET: /Account/ResetPassword**  **[AllowAnonymous]**  **[Route("Senha/Redefinir")]**  **public ActionResult ResetPassword(string code)**  **{**  **return code == null ? View("Error") : View();**  **}** |

ActionResult ResetPassword Post

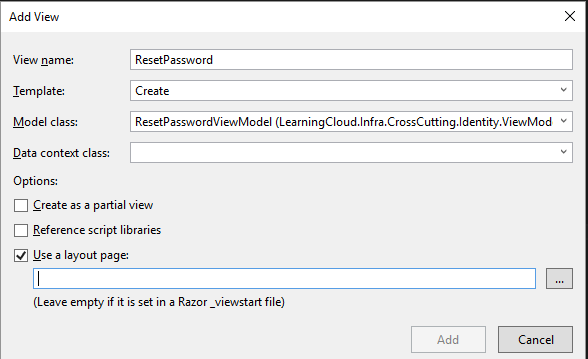
|  |
| --- |
| **// POST: /Account/ResetPassword**  **[HttpPost]**  **[AllowAnonymous]**  **[Route("Senha/Redefinir")]**  **[ValidateAntiForgeryToken]**  **public async Task<ActionResult> ResetPassword(ResetPasswordViewModel model)**  **{**  **if (!ModelState.IsValid)**  **{**  **return View(model);**  **}**  **var user = await \_userManager.FindByEmailAsync(model.UsuarioAcesso\_Email);**  **if (user == null)**  **{**  **// Não revelar se o usuario nao existe ou nao esta confirmado**  **return RedirectToAction("ResetPasswordConfirmation", "Account");**  **}**  **var result = await \_userManager.ResetPasswordAsync(user.Id, model.Code, model.Password);**  **if (result.Succeeded)**  **{**  **return RedirectToAction("ResetPasswordConfirmation", "Account");**  **}**  **AddErrors(result);**  **return View();**  **}** |

viewmodel ResetPasswordViewModel na pasta ViewModels/AccountViewModels na camada Infra.CrossCutting.Identity

|  |
| --- |
| **using System.ComponentModel.DataAnnotations;**  **namespace NewLearningCloud.Infra.CrossCutting.Identity.ViewModels.AccountViewModels**  **{**  **public class ResetPasswordViewModel**  **{**  **[Required(ErrorMessage = "Preencha o campo Email.")]**  **[EmailAddress]**  **[Display(Name = "E-mail")]**  **public string UsuarioAcesso\_Email { get; set; }**  **[Required(ErrorMessage = "Preencha o campo Senha.")]**  **[StringLength(100, ErrorMessage = "A {0} deve conter pelo menos {2} caracteres.", MinimumLength = 6)]**  **[DataType(DataType.Password)]**  **[Display(Name = "Senha")]**  **public string UsuarioAcesso\_Password { get; set; }**    **[Required(ErrorMessage = "Preencha o campo Confirmação da Senha.")]**  **[DataType(DataType.Password)]**  **[Display(Name = "Confirmação da Senha")]**  **[Compare("UsuarioAcesso\_Password", ErrorMessage = "A senha e a confirmação da senha estão diferentes.")]**  **public string UsuarioAcesso\_ConfirmPassword { get; set; }**  **[ScaffoldColumn(false)]**  **public string Code { get; set; }**  **}**  **}** |

Definir como public

Criara a view ResetPassword (Create com **ResetPasswordViewModel (LearningCloud.Infra.CrossCutting.Identity.ViewModels.AccountViewModel**s))



Alterar os seguintes pontos em destaque na View

|  |
| --- |
| @model LearningCloud.Infra.CrossCutting.Identity.ViewModels.AccountViewModels.ResetPasswordViewModel  @{  ViewBag.Title = **"Redefinir Senha"**;  }  **<h2> @ViewBag.Title</h2>**  **@using (Html.BeginForm("ResetPassword", "Account", FormMethod.Post, new { @class = "form-horizontal", role = "form" }))**  **{**  **@Html.AntiForgeryToken()**    **<div class="form-horizontal">**  **<h4>Crie uma nova senha.</h4>**  **<hr />**  **@Html.ValidationSummary(true, "", new { @class = "text-danger" })**  **@Html.HiddenFor(model => model.Code)**  **<div class="form-group">**  **@Html.LabelFor(model => model.UsuarioAcesso\_Email, htmlAttributes: new { @class = "control-label col-md-2" })**  **<div class="col-md-10">**  **@Html.EditorFor(model => model.UsuarioAcesso\_Email, new** **{ htmlAttributes = new { @class = "form-control" } })**  **@Html.ValidationMessageFor(model => model.UsuarioAcesso\_Email, "", new { @class = "text-danger" })**  **</div>**  **</div>**  **<div class="form-group">**  **@Html.LabelFor(model => model.UsuarioAcesso\_Password, htmlAttributes: new { @class = "control-label col-md-2" })**  **<div class="col-md-10">**  **@Html.EditorFor(model => model.UsuarioAcesso\_Password, new { htmlAttributes = new { @class = "form-control" } })**  **@Html.ValidationMessageFor(model => model.UsuarioAcesso\_Password, "", new { @class = "text-danger" })**  **</div>**  **</div>**  **<div class="form-group">**  **@Html.LabelFor(model => model.UsuarioAcesso\_ConfirmPassword, htmlAttributes: new { @class = "control-label col-md-2" })**  **<div class="col-md-10">**  **@Html.EditorFor(model => model.UsuarioAcesso\_ConfirmPassword, new { htmlAttributes = new { @class = "form-control" } })**  **@Html.ValidationMessageFor(model => model.UsuarioAcesso\_ConfirmPassword, "", new { @class = "text-danger" })**  **</div>**  **</div>**  **<div class="form-group">**  **<div class="col-md-offset-2 col-md-10">**  **<input type="submit" value="Enviar" class="btn btn-default" />**  **</div>**  **</div>**  **</div>**  **}**  **<div>**  **@Html.ActionLink("Back to List", "Index")**  **</div>** |

ActionResult ResetPasswordConfirmation Get

|  |
| --- |
| **// GET: /Account/ResetPasswordConfirmation**  **[AllowAnonymous]**  **[Route("Senha/Redefinir/Confirmar")]**  **public ActionResult ResetPasswordConfirmation()**  **{**  **return View();**  **}** |

Criar a view (Empty (without model))

|  |
| --- |
| **@{**  **ViewBag.Title = "Confirmação de redefinição de senha";**  **}**  **<div>**  **<p>**  **Sua senha foi redefinida. Por favor @Html.ActionLink("clique aqui para efetuar login", "Login", "Account", routeValues: null, htmlAttributes: new { id = "loginLink" })**  **</p>**  **</div>** |

Criar as ActionResults SendCode, VerifyCode e as ViewModels SendCodeViewModel e VerifyCodeViewModel

ActionResult SendCode Get

|  |
| --- |
| **// GET: /Account/SendCode**  **[AllowAnonymous]**  **[Route("Codigo")]**  **public async Task<ActionResult> SendCode(string returnUrl, bool rememberMe)**  **{**  **var userId = await \_signInManager.GetVerifiedUserIdAsync();**  **if (userId == null)**  **{**  **List<string> callbackError = new List<string>();**  **callbackError.Add("Problemas ao identificar o usuário.");**  **TempData["CallbackError"] = callbackError;**  **return View("Error");**  **}**  **var userFactors = await \_userManager.GetValidTwoFactorProvidersAsync(userId);**  **var factorOptions = userFactors.Select(purpose => new SelectListItem { Text = purpose, Value = purpose }).ToList();**  **return View(new SendCodeViewModel { Providers = factorOptions, ReturnUrl = returnUrl, RememberMe = rememberMe });**  **}** |

Adiciona o using System.Linq;

**using System.Linq;**

Criar a ViewModel **SendCodeViewModel** na pasta ViewModels/ AccountViewModels na camada Infra.CrossCutting.Identity definir como **public**

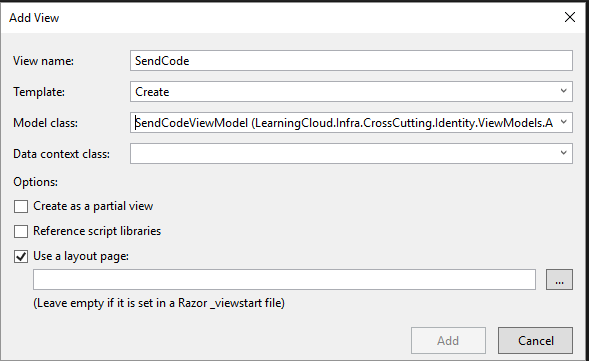
**Adiciona a referencia para System.Web.Mvc;** na camadaInfra.CrossCutting.Identity

|  |
| --- |
| **using System.Web.Mvc;**  **using System.Collections.Generic;**  **using System.ComponentModel.DataAnnotations;**  **namespace NewLearningCloud.Infra.CrossCutting.Identity.ViewModels.AccountViewModels**  **{**  **public class SendCodeViewModel**  **{**  **[Required(ErrorMessage = "Informe Provedor de dois fatores de Autenticação")]**  **[Display(Name = "Provedor de dois fatores de Autenticação")]**  **public string SelectedProvider { get; set; }**  **public ICollection<SelectListItem> Providers { get; set; }**  **[ScaffoldColumn(false)]**  **public string ReturnUrl { get; set; }**  **[ScaffoldColumn(false)]**  **public bool RememberMe { get; set; }**  **}**  **}** |

ActionResult SendCode Post

|  |
| --- |
| **// POST: /Account/SendCode**  **[HttpPost]**  **[AllowAnonymous]**  **[Route("Codigo")]**  **[ValidateAntiForgeryToken]**  **public async Task<ActionResult> SendCode(SendCodeViewModel model)**  **{**  **if (!ModelState.IsValid)**  **{**  **List<string> callbackError = new List<string>();**  **callbackError.Add("Problemas ao enviar o código de verificação.");**  **TempData["CallbackError"] = callbackError;**  **return View();**  **}**  **// Generate the token and send it**  **if (!await \_signInManager.SendTwoFactorCodeAsync(model.SelectedProvider))**  **{**  **return View("Error");**  **}**  **return RedirectToAction("VerifyCode", new { Provider = model.SelectedProvider, ReturnUrl = model.ReturnUrl, RememberMe = model.RememberMe });**  **}** |

Criar a View para SendCode (Create com **SendCodeViewModel (LearningCloud.Infra.CrossCutting.Identity.ViewModels.AccountViewModels)**)



Alterar os seguintes códigos em destaque

|  |
| --- |
| **@model NewLearningCloud.Infra.CrossCutting.Identity.ViewModels.AccountViewModels.SendCodeViewModel**  **@{**  **ViewBag.Title = "Enviar código de verificação";**  **}**  **<h2>@ViewBag.Title</h2>**  **@using (Html.BeginForm("SendCode", "Account", new { ReturnUrl = Model.ReturnUrl }, FormMethod.Post, new { @class = "form-horizontal", role = "form" }))**  **{**  **@Html.AntiForgeryToken()**  **@Html.Hidden("rememberMe", @Model.RememberMe)**  **<hr />**  **<div class="form-horizontal">**  **<div class="form-group">**  **@Html.LabelFor(model => model.SelectedProvider, htmlAttributes: new { @class = "control-label col-md-4 required" })**  **<div class="col-md-8">**  **@Html.DropDownListFor(model => model.SelectedProvider, Model.Providers, String.Empty, new { @class = "form-control" })**  **@Html.ValidationMessageFor(model => model.SelectedProvider, "", new { @class = "text-danger" })**  **</div>**  **</div>**  **<div class="form-group">**  **<div class="col-md-offset-4 col-md-8">**  **<input type="submit" value="Enviar" class="btn btn-default" />**  **</div>**  **</div>**  **</div>**  **}**  **@section Scripts {**  **@Scripts.Render("~/bundles/jqueryval")**  **}** |

Criara ActionResult para VerifyCode Get

|  |
| --- |
| **// GET: /Account/VerifyCode**  **[AllowAnonymous]**  **[Route("Codigo/Verificar")]**  **public async Task<ActionResult> VerifyCode(string provider, string returnUrl, bool rememberMe)**  **{**  **// Require that the user has already logged in via username/password or external login**  **if (!await \_signInManager.HasBeenVerifiedAsync())**  **{**  **List<string> callbackError = new List<string>();**  **callbackError.Add("Problemas ao verificar o código.");**  **TempData["CallbackError"] = callbackError;**  **return View("Error");**  **}**  **var user = await \_userManager.FindByIdAsync(await \_signInManager.GetVerifiedUserIdAsync());**  **if (user != null)**  **{**  **ViewBag.Status = "DEMO: Caso não chegue o " + provider + ". O código é: ";**  **ViewBag.CodigoAcesso = await \_userManager.GenerateTwoFactorTokenAsync(user.Id, provider);**  **}**  **return View(new VerifyCodeViewModel { Provider = provider, ReturnUrl = returnUrl, RememberMe = rememberMe });**  **}** |

Criara a ViewModel **VerifyCodeViewModel** na pasta ViewModels/AccountViewModels na camada de Infra.CrossCutting.Identity, definir como **public**

|  |
| --- |
| **using System.ComponentModel.DataAnnotations;**  **namespace LearningCloud.Infra.CrossCutting.Identity.ViewModels.AccountViewModels**  **{**  **public class VerifyCodeViewModel**  **{**  **[Required(ErrorMessage = "Problemas ao recuperar o provedor. Atualize a página e tente novamente.")]**  **public string Provider { get; set; }**  **[Required(ErrorMessage = "Preencha o campo Código.")]**  **[Display(Name = "Código")]**  **public string Code { get; set; }**  **public string ReturnUrl { get; set; }**  **[Display(Name = "Lembrar desse navegador?")]**  **public bool RememberBrowser { get; set; }**  **[ScaffoldColumn(false)]**  **public bool RememberMe { get; set; }**  **}**  **}** |

ActionResult VerifyCode Post

|  |
| --- |
| **// POST: /Account/VerifyCode**  **[HttpPost]**  **[AllowAnonymous]**  **[Route("Codigo/Verificar")]**  **[ValidateAntiForgeryToken]**  **public async Task<ActionResult> VerifyCode(VerifyCodeViewModel model)**  **{**  **if (!ModelState.IsValid)**  **{**  **return View(model);**  **}**  **var result = await \_signInManager.TwoFactorSignInAsync(model.Provider, model.Code, isPersistent: model.RememberMe, rememberBrowser: model.RememberBrowser);**  **switch (result)**  **{**  **case SignInStatus.Success:**  **return RedirectToLocal(model.ReturnUrl);**  **case SignInStatus.LockedOut:**  **return View("Lockout");**  **case SignInStatus.Failure:**  **default:**  **ModelState.AddModelError("", "Código Inválido.");**  **return View(model);**  **}**  **}** |

Criar a View para VerifyCode (Empty com )

Verificar implementação da ApplicationRoleManager identity Isolation

incluir o codigo a baixo

|  |
| --- |
| **@model LearningCloud.Infra.CrossCutting.Identity.ViewModels.AccountViewModels.VerifyCodeViewModel**  **@{**  **ViewBag.Title = "Validação de código de verificação";**  **}**  **<h2>@ViewBag.Title</h2>**  **@using (Html.BeginForm("VerifyCode", "Account", new { ReturnUrl = Model.ReturnUrl }, FormMethod.Post, new { @class = "form-horizontal", role = "form" }))**  **{**  **@Html.AntiForgeryToken()**  **@Html.Hidden("provider", @Model.Provider)**  **@Html.Hidden("rememberMe", @Model.RememberMe)**  **<hr />**  **@Html.ValidationSummary("", new { @class = "text-danger" })**  **<div class="form-group">**  **@Html.LabelFor(m => m.Code, new { @class = "col-md-4 control-label" })**  **<div class="col-md-8">**  **@Html.TextBoxFor(m => m.Code, new { @class = "form-control" })**  **</div>**  **</div>**  **<div class="form-group">**  **<div class="col-md-offset-4 col-md-8">**  **<div class="checkbox">**  **@Html.CheckBoxFor(m => m.RememberBrowser)**  **@Html.LabelFor(m => m.RememberBrowser)**  **</div>**  **</div>**  **</div>**  **<div class="form-group">**  **<div class="col-md-offset-4 col-md-8">**  **<input type="submit" class="btn btn-default" value="Enviar" />**  **<h5>@ViewBag.Status &nbsp;<strong>@ViewBag.CodigoAcesso</strong></h5>**  **</div>**  **</div>**  **}**  **@section Scripts {**  **@Scripts.Render("~/bundles/jqueryval")**  **}** |

Vamos criar um novo controller na pasta Controllers no projeto MVC

* LearningCloud.MVC
  + Pasta Controllers (Clica com direito)
  + Add
  + Controller…
  + MVC 5 Controller - Empty
  + ManageController

|  |
| --- |
| **using System.Web.Mvc;**  **namespace LearningCloud.MVC.Controllers**  **{**  **public class ManageController : Controller**  **{**  **// GET: Manage**  **public ActionResult Index()**  **{**  **return View();**  **}**  **}**  **}** |

Acrescentar o [Authorize] e [RoutePrefix("Gerenciar")] na classe

|  |
| --- |
| **namespace LearningCloud.MVC.Controllers**  **{**  [Authorize]  [RoutePrefix("Gerenciar")]  **public class ManageController : Controller**  **{**  **// GET: Manage**  **public ActionResult Index()**  **{**  **return View();**  **}**  **}**  **}** |

Criar dois campos privados e um construtor para injetar o ApplicationSignInManager e o ApplicationUserManager

|  |
| --- |
| **using NewLearningCloud.Infra.CrossCutting.Identity.Configuration;**  **namespace NewLearningCloud.MVC.Controllers**  **{**  **[Authorize]**  **[RoutePrefix("Gerenciar")]**  **public class ManageController : Controller**  **{**  **private ApplicationSignInManager \_signInManager;**  **private ApplicationUserManager \_userManager;**  **public ManageController(ApplicationUserManager userManager, ApplicationSignInManager signInManager)**  **{**  **\_userManager = userManager;**  **\_signInManager = signInManager;**  **}** |

Vamos criar uma sobrescrita para o método Dispose para destruir as instâncias que injetamos no controller

|  |
| --- |
| **protected override void Dispose(bool disposing)**  **{**  **if (disposing)**  **{**  **if (\_userManager != null)**  **{**  **\_userManager.Dispose();**  **\_userManager = null;**  **}**  **if (\_signInManager != null)**  **{**  **\_signInManager.Dispose();**  **\_signInManager = null;**  **}**  **}**  **base.Dispose(disposing);**  **}** |

Criar uma #region Helpers no final da classe ManageController

|  |
| --- |
| **#region Helpers**  **// Used for XSRF protection when adding external logins**  **private const string XsrfKey = "XsrfId";**  **private IAuthenticationManager AuthenticationManager**  **{**  **get**  **{**  **return HttpContext.GetOwinContext().Authentication;**  **}**  **}**  **private void AddErrors(IdentityResult result)**  **{**  **foreach (var error in result.Errors)**  **{**  **ModelState.AddModelError("", error);**  **}**  **}**  **private bool HasPassword()**  **{**  **var user = \_userManager.FindById(User.Identity.GetUserId());**  **if (user != null)**  **{**  **return user.PasswordHash != null;**  **}**  **return false;**  **}**  **public enum ManageMessageId**  **{**  **AddPhoneSuccess,**  **ChangePasswordSuccess,**  **SetTwoFactorSuccess,**  **SetPasswordSuccess,**  **RemoveLoginSuccess,**  **LinkLoginSuccess,**  **RemovePhoneSuccess,**  **Error**  **}**  **#endregion**  }  } |

VAmos alterar a ActionResult Index ficando

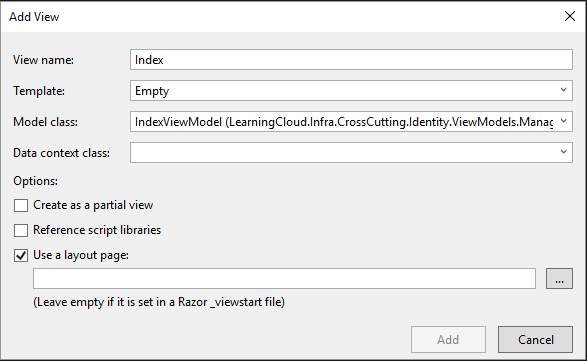
|  |
| --- |
| **// GET: Manage**  **public async Task<ActionResult> Index(ManageMessageId? message)**  **{**  **ViewBag.Status = 0;**  **ViewBag.StatusMessage =**  **message == ManageMessageId.ChangePasswordSuccess ? "A senha foi alterada com sucesso."**  **: message == ManageMessageId.SetPasswordSuccess ? "A senha foi enviada com sucesso."**  **: message == ManageMessageId.SetTwoFactorSuccess ? "A segunda validação foi enviada com sucesso."**  **: message == ManageMessageId.Error ? "Ocorreu uma exceção ao processar sua solicitação."**  **: message == ManageMessageId.AddPhoneSuccess ? "O Telefone foi adicionado com sucesso."**  **: message == ManageMessageId.RemovePhoneSuccess ? "O Telefone foi removido com sucesso."**  **: "";**  **if (message == ManageMessageId.Error)**  **{**  **ViewBag.Status = -1;**  **}**  **var userId = User.Identity.GetUserId();**  **var model = new IndexViewModel**  **{**  **HasPassword = HasPassword(),**  **PhoneNumber = await \_userManager.GetPhoneNumberAsync(userId),**  **TwoFactor = await \_userManager.GetTwoFactorEnabledAsync(userId),**  **Logins = await \_userManager.GetLoginsAsync(userId),**  **BrowserRemembered = await AuthenticationManager.TwoFactorBrowserRememberedAsync(userId)**  **};**  **return View(model);**  **}** |

Vamos criar a ViewModel IndexViewModel na pasta ViewModels/ManageViewModels do projeto Infra.CrossCutting.Identity

IndexViewModel

|  |
| --- |
| **using System.Collections.Generic;**  **using Microsoft.AspNet.Identity;**  **namespace NewLearningCloud.Infra.CrossCutting.Identity.ViewModels.ManageViewModels**  **{**  **public class IndexViewModel**  **{**  **public bool HasPassword { get; set; }**  **public IList<UserLoginInfo> Logins { get; set; }**  **public string PhoneNumber { get; set; }**  **public bool TwoFactor { get; set; }**  **public bool BrowserRemembered { get; set; }**  **}**  **}** |

Vamos criar a view Index para ManageController



Adicionar o seguinte código

|  |
| --- |
| **@model LearningCloud.Infra.CrossCutting.Identity.ViewModels.ManageViewModels.IndexViewModel**  **@using NewLearningCloud.Infra.CrossCutting.Identity.Configuration;**  **@{**  **ViewBag.Title = "Gerenciar Conta";**  **int \_status = ViewBag.Status;**  **}**  **<h2>@ViewBag.Title</h2>**  **@if (!string.IsNullOrEmpty(ViewBag.StatusMessage))**  **{**  **if (\_status < 0)**  **{**  **<div class="alert alert-danger alert-dismissible fade in" role="alert">**  **<button type="button" class="close" data-dismiss="alert" aria-label="Fechar">**  **<span aria-hidden="true">×</span>**  **</button>**  **<h5><strong>@ViewBag.StatusMessage</strong></h5>**  **</div>**  **}**  **else if (\_status > 0)**  **{**  **<div class="alert alert-warning alert-dismissible fade in" role="alert">**  **<button type="button" class="close" data-dismiss="alert" aria-label="Fechar">**  **<span aria-hidden="true">×</span>**  **</button>**  **<h5><strong>@ViewBag.StatusMessage</strong></h5>**  **</div>**  **}**  **else**  **{**  **<div class="alert alert-success alert-dismissible fade in" role="alert">**  **<button type="button" class="close" data-dismiss="alert" aria-label="Fechar">**  **<span aria-hidden="true">×</span>**  **</button>**  **<h5><strong>@ViewBag.StatusMessage</strong></h5>**  **</div>**  **}**  **}**  **<table class="table table-hover">**  **<tbody>**  **<tr>**  **<th scope="row">Senha:</th>**  **@if (Model.HasPassword)**  **{**  **<td><span class="badge badge-info" style="padding:6px 5px 0 5px;"><text class="text-center" style="font-size:13pt;">\* \* \* \* \* \* \* \* \* \* \* \*</text></span></td>**  **<td>@Html.ActionLink("Trocar Senha", "ChangePassword", null, new { @class = "btn btn-default" })</td>**  **}**  **else**  **{**  **<td>Não foi definida uma senha para conta local.</td>**  **<td>@Html.ActionLink("Definir senha para conta local", "SetPassword", null, new { @class = "btn btn-default" })</td>**  **}**  **</tr>**  **<tr>**  **<th scope="row">Logins Externos:</th>**  **<td>Logins Sociais: <span style="padding:5px 15px;" class="badge badge-pill @(Model.Logins.Count > 0 ? "badge-info": "badge-warning")">@Model.Logins.Count</span></td>**  **<td>**  **@Html.ActionLink("Gerenciar", "ManageLogins", null, new { @class = "btn btn-default" })**  **</td>**  **</tr>**  **<tr>**  **<th scope="row">Número de celular:</th>**  **@if (Model.PhoneNumber != null)**  **{**  **<td><span style="padding:5px 15px;" class="badge badge-warning">@Model.PhoneNumber</span></td>**  **<td>**  **@Html.ActionLink("Trocar", "AddPhoneNumber", null, new { @class = "btn btn-default" })**  **@Html.ActionLink("Remover", "RemovePhoneNumber", null, new { @class = "btn btn-danger" })**  **</td>**  **}**  **else**  **{**  **<td>Não Informado.**  **<span style="padding:5px 15px;" class="badge badge-warning"></span>**  **</td>**  **<td>@Html.ActionLink("Adicionar", "AddPhoneNumber", null, new { @class = "btn btn-default" })</td>**  **}**  **</tr>**  **<tr>**  **<th scope="row">E-mail:</th>**  **@if (!Model.EmailConfirmed)**  **{**  **<td>**  **E-mail: <span style="padding:5px 15px;" class="badge badge-warning">@Model.Email</span> Não confirmado.**  **</td>**  **<td>@Html.ActionLink("Confirmar", "DisplayEmail", "Account", new { Id = @User.GetInfoUser("UserAccessId") }, new { @class = "btn btn-default" })</td>**  **}**  **else**  **{**  **<td><span style="padding:5px 15px;" class="badge badge-info">@Model.Email</span></td>**  **<td>@Html.ActionLink("Alterar", "EditEmail", "Account", new { Id = @User.GetInfoUser("UserAccessId") }, new { @class = "btn btn-default disabled" })</td>**  **}**  **</tr>**  **@if (Model.EmailConfirmed || Model.PhoneNumber != null)**  **{**  **<tr>**  **<th scope="row">Autenticação de dois fatores:</th>**  **@if (Model.TwoFactor)**  **{**  **<td>Two Factor está habilitado.<span style="padding:5px 15px;" class="badge badge-success"></span></td>**  **using (Html.BeginForm("DisableTwoFactorAuthentication", "Manage", FormMethod.Post, new { @class = "form-horizontal", role = "form" }))**  **{**  **@Html.AntiForgeryToken()**  **<td>**  **<input type="submit" value="Desabilitar" class="btn btn-warning" />**  **</td>**  **}**  **}**  **else**  **{**  **<td>Two Factor NÃO está habilitado.<span style="padding:5px 15px;" class="badge badge-warning"></span></td>**  **using (Html.BeginForm("EnableTwoFactorAuthentication", "Manage", FormMethod.Post, new { @class = "form-horizontal", role = "form" }))**  **{**  **@Html.AntiForgeryToken()**  **<td>**  **<input type="submit" value="Habilitar" class="btn btn-default" />**  **</td>**  **}**  **}**  **</tr>**  **<tr>**  **<th scope="row"></th>**  **@if (Model.BrowserRemembered)**  **{**  **<td>O browser será lembrado para Two Factor.<span style="padding:5px 15px;" class="badge badge-success"></span></td>**  **using (Html.BeginForm("ForgetBrowser", "Manage", FormMethod.Post, new { @class = "form-horizontal", role = "form" }))**  **{**  **@Html.AntiForgeryToken()**  **<td>**  **<input type="submit" value="Esquecer Browser" class="btn btn-default" />**  **</td>**  **}**  **}**  **else**  **{**  **<td>O browser NÃO será lembrado para Two Factor.<span style="padding:5px 15px;" class="badge badge-warning"></span></td>**  **using (Html.BeginForm("RememberBrowser", "Manage", FormMethod.Post, new { @class = "form-horizontal", role = "form" }))**  **{**  **@Html.AntiForgeryToken()**  **<td>**  **<input type="submit" value="Lembrar Browser" class="btn btn-default" />**  **</td>**  **}**  **}**  **</tr>**  **}**  **else**  **{**  **<tr>**  **<th scope="row">Autenticação de dois fatores:</th>**  **<td colspan="2">**  **<i style="color: #ffd800;text-shadow: 1px 1px 5px #292929; font-size: 1.2em;" class="fa fa-exclamation-triangle">&nbsp;&nbsp;</i>**  **<span class="text-warning">Para habilitar a autenticação de de dois fatores adicione um <strong>número de celular</strong> ou <strong>confirme seu e-mail</strong>.</span>**  **</td>**  **</tr>**  **}**  **</tbody>**  **</table>** |

Alterar a view \_LoginPartial na pasta Views/ Shared para adicionar o link de acesso ao gerenciamento de conta

de:

|  |
| --- |
| **<li>**  **<a href="#">**  **<i class="fa fa-user-md">&nbsp;&nbsp;</i>Minha Conta**  **</a>**  **</li>** |

Para:

|  |
| --- |
| **<li>**  **<a href="@Url.Action("Index", "Manage", new { Area = "" })">**  **<i class="fa fa-user-md">&nbsp;&nbsp;</i>Minha Conta**  **</a>**  **</li>** |

Vamos criar as ActionsResult para ChangePassword e a viewmodel ChangePasswordViewModel na pasta ViewModels/ManageViewModels no projeto Infra.CrossCutting.Identity

ChangePassword GET

|  |
| --- |
| **// GET: /Manage/ChangePassword**  **public ActionResult ChangePassword()**  **{**  **return View();**  **}** |

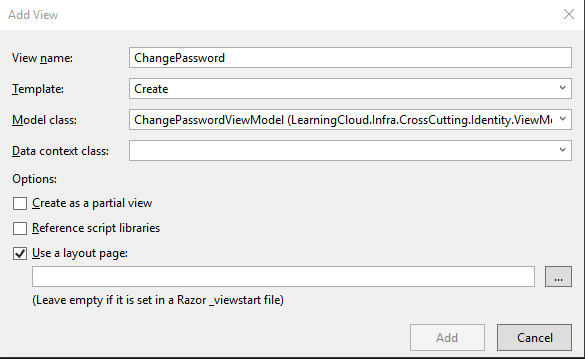
ChangePassword POST

|  |
| --- |
| **// POST: /Manage/ChangePassword**  **[HttpPost]**  **[ValidateAntiForgeryToken]**  **public async Task<ActionResult> ChangePassword(ChangePasswordViewModel model)**  **{**  **if (!ModelState.IsValid)**  **{**  **return View(model);**  **}**  **var result = await \_userManager.ChangePasswordAsync(User.Identity.GetUserId(), model.UsuarioAcesso\_OldPassword, model.UsuarioAcesso\_NewPassword);**  **if (result.Succeeded)**  **{**  **var user = await \_userManager.FindByIdAsync(User.Identity.GetUserId());**  **if (user != null)**  **{**  **await \_signInManager.SignInAsync(user, isPersistent: false, rememberBrowser: false);**  **}**  **return RedirectToAction("Index", new { Message = ManageMessageId.ChangePasswordSuccess });**  **}**  **AddErrors(result);**  **return View(model);**  **}** |

ChangePasswordViewModel

|  |
| --- |
| **using System.ComponentModel.DataAnnotations;**  **namespace NewLearningCloud.Infra.CrossCutting.Identity.ViewModels.ManageViewModels**  **{**  **public class ChangePasswordViewModel**  **{**  **[Required(ErrorMessage = "Preencha o campo Senha atual.")]**  **[DataType(DataType.Password)]**  **[Display(Name = "Senha atual")]**  **public string UsuarioAcesso\_OldPassword { get; set; }**  **[Required(ErrorMessage = "Preencha o campo Nova senha.")]**  **[StringLength(100, ErrorMessage = "A {0} deve conter pelo menos {2} caracteres.", MinimumLength = 6)]**  **[DataType(DataType.Password)]**  **[Display(Name = "Nova senha")]**  **public string UsuarioAcesso\_NewPassword { get; set; }**  **[Required(ErrorMessage = "Preencha o campo Confirmação da Senha.")]**  **[DataType(DataType.Password)]**  **[Display(Name = "Confirmação da Senha")]**  **[Compare("UsuarioAcesso\_NewPassword", ErrorMessage = "A senha e a confirmação da senha estão diferentes.")]**  **public string UsuarioAcesso\_ConfirmPassword { get; set; }**  **}**  **}** |

Vamos adicionar a Viwe ChangePassword



Alterar os codigos em destaque:

|  |
| --- |
| @model LearningCloud.Infra.CrossCutting.Identity.ViewModels.ManageViewModels.ChangePasswordViewModel  @{  ViewBag.Title = **"Trocar senha";**  }  **<h2>@ViewBag.Title</h2>**  **@using (Html.BeginForm())**  **{**  **@Html.AntiForgeryToken()**    **<div class="form-horizontal">**  **<h4>ChangePasswordViewModel</h4>**  **<hr />**  **@Html.ValidationSummary(true, "", new { @class = "text-danger" })**  **<div class="form-group">**  **@Html.LabelFor(model => model.UsuarioAcesso\_OldPassword, htmlAttributes: new { @class = "control-label col-md-2" })**  **<div class="col-md-10">**  **@Html.EditorFor(model => model.UsuarioAcesso\_OldPassword, new { htmlAttributes = new { @class = "form-control" } })**  **@Html.ValidationMessageFor(model => model.UsuarioAcesso\_OldPassword, "", new { @class = "text-danger" })**  **</div>**  **</div>**  **<div class="form-group">**  **@Html.LabelFor(model => model.UsuarioAcesso\_NewPassword, htmlAttributes: new { @class = "control-label col-md-2" })**  **<div class="col-md-10">**  **@Html.EditorFor(model => model.UsuarioAcesso\_NewPassword, new { htmlAttributes = new { @class = "form-control" } })**  **@Html.ValidationMessageFor(model => model.UsuarioAcesso\_NewPassword, "", new { @class = "text-danger" })**  **</div>**  **</div>**  **<div class="form-group">**  **@Html.LabelFor(model => model.UsuarioAcesso\_ConfirmPassword, htmlAttributes: new { @class = "control-label col-md-2" })**  **<div class="col-md-10">**  **@Html.EditorFor(model => model.UsuarioAcesso\_ConfirmPassword, new { htmlAttributes = new { @class = "form-control" } })**  **@Html.ValidationMessageFor(model => model.UsuarioAcesso\_ConfirmPassword, "", new { @class = "text-danger" })**  **</div>**  **</div>**  **<div class="form-group">**  **<div class="col-md-offset-2 col-md-10">**  **<input type="submit" value="Trocar senha" class="btn btn-default" />**  **</div>**  **</div>**  **</div>**  **}**  **<div>**  **@Html.ActionLink("Back to List", "Index")**  **</div>** |

Vamos alterar a View \_LoginPartial para criar um atalho para a alteração de senha

de

|  |
| --- |
| **<a href="#">**  **<i class="fa fa-cogs">&nbsp;&nbsp;</i>Alterar Senha**  **</a>** |

Para:

|  |
| --- |
| **<a href="@Url.Action("ChangePassword", "Manage", new { Area = "" })">**  **<i class="fa fa-cogs">&nbsp;&nbsp;</i>Alterar Senha**  **</a>** |

Vamos incluir as ActionResult SetPassword no ManageController e tambem a viewmodel SetPasswordViewModel na pasta ViewModels/ManageViewModels do projeto Infra.CrossCutting.Identity

**GET** SetPassword

|  |
| --- |
| **// GET: /Manage/SetPassword**  **[Route("Conta/CriarSenha")]**  **public ActionResult SetPassword()**  **{**  **return View();**  **}** |

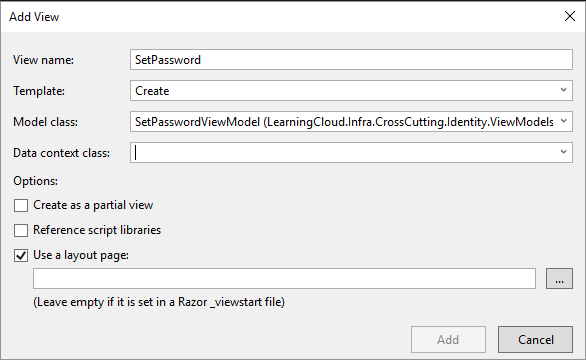
POST SetPassword

|  |
| --- |
| **// POST: /Manage/SetPassword**  **[HttpPost]**  **[Route("Conta/CriarSenha")]**  **[ValidateAntiForgeryToken]**  **public async Task<ActionResult> SetPassword(SetPasswordViewModel model)**  **{**  **if (ModelState.IsValid)**  **{**  **var result = await \_userManager.AddPasswordAsync(User.Identity.GetUserId(), model.UsuarioAcesso\_NewPassword );**  **if (result.Succeeded)**  **{**  **var user = await \_userManager.FindByIdAsync(User.Identity.GetUserId());**  **if (user != null)**  **{**  **await \_signInManager.SignInAsync(user, isPersistent: false, rememberBrowser: false);**  **}**  **return RedirectToAction("Index", new { Message = ManageMessageId.SetPasswordSuccess });**  **}**  **AddErrors(result);**  **}**  **// If we got this far, something failed, redisplay form**  **return View(model);**  **}** |

SetPasswordViewModel

|  |
| --- |
| **using System.ComponentModel.DataAnnotations;**  **namespace NewLearningCloud.Infra.CrossCutting.Identity.ViewModels.ManageViewModels**  **{**  **public class SetPasswordViewModel**  **{**  **[Required(ErrorMessage = "Preencha o campo Nova Senha.")]**  **[StringLength(100, ErrorMessage = "A {0} deve conter pelo menos {2} caracteres.", MinimumLength = 6)]**  **[DataType(DataType.Password)]**  **[Display(Name = "Nova Senha")]**  **public string UsuarioAcesso\_NewPassword { get; set; }**    **[Required(ErrorMessage = "Preencha o campo Confirmação da Senha.")]**  **[DataType(DataType.Password)]**  **[Display(Name = "Confirmação da Senha")]**  **[Compare("UsuarioAcesso\_NewPassword", ErrorMessage = "A senha e a confirmação da senha estão diferentes.")]**  **public string UsuarioAcesso\_ConfirmPassword { get; set; }**  **}**  **}** |

Vamos adicionar a View para SetPassword



Alterar as seguintes linhas em destaque

|  |
| --- |
| @model LearningCloud.Infra.CrossCutting.Identity.ViewModels.ManageViewModels.SetPasswordViewModel  @{  ViewBag.Title = "**Criar Senha Local**";  }  <h2>**@ViewBag.Title**</h2>  **<p class="text-info">**  **Você não tem uma senha local para este site.**  **Adicione uma senha para a conta local para que você possa entrar sem um login externo.**  **</p>**  **<hr />**  @using (Html.BeginForm())  {  @Html.AntiForgeryToken()    <div class="form-horizontal">  <h4>SetPasswordViewModel</h4>  <hr />  @Html.ValidationSummary(true, "", new { @class = "text-danger" })  <div class="form-group">  @Html.LabelFor(model => model.UsuarioAcesso\_NewPassword, htmlAttributes: new { @class = "control-label col-md-2" })  <div class="col-md-10">  @Html.EditorFor(model => model.UsuarioAcesso\_NewPassword, new { htmlAttributes = new { @class = "form-control" } })  @Html.ValidationMessageFor(model => model.UsuarioAcesso\_NewPassword, "", new { @class = "text-danger" })  </div>  </div>  <div class="form-group">  @Html.LabelFor(model => model.UsuarioAcesso\_ConfirmPassword, htmlAttributes: new { @class = "control-label col-md-2" })  <div class="col-md-10">  @Html.EditorFor(model => model.UsuarioAcesso\_ConfirmPassword, new { htmlAttributes = new { @class = "form-control" } })  @Html.ValidationMessageFor(model => model.UsuarioAcesso\_ConfirmPassword, "", new { @class = "text-danger" })  </div>  </div>  <div class="form-group">  <div class="col-md-offset-2 col-md-10">  <input type="submit" value="**Salvar**" class="btn btn-default" />  </div>  </div>  </div>  }  <div>  @Html.ActionLink("Back to List", "Index")  </div> |

ActionResult ManageLogins no ManageController

Adiciona o using System.Linq;

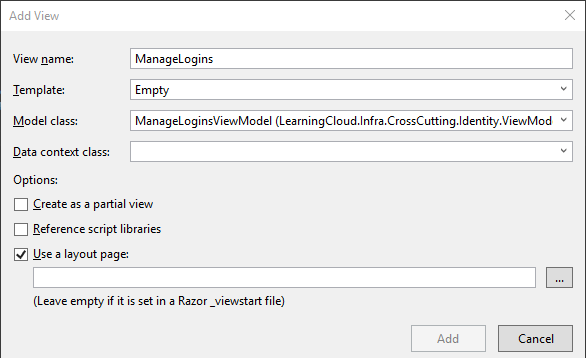
|  |
| --- |
| **// GET: /Manage/ManageLogins**  **[Route("Conta/GerenciarLogins")]**  **public async Task<ActionResult> ManageLogins(ManageMessageId? message)**  **{**  **ViewBag.Status = 0;**  **var user = await \_userManager.FindByIdAsync(User.Identity.GetUserId());**  **if (user == null)**  **{**  **List<string> callbackError = new List<string>();**    **callbackError.Add("Problemas ao identificar o usuário.");**  **TempData["CallbackError"] = callbackError;**  **return View("Error");**  **}**  **if (message == ManageMessageId.Error)**  **{**  **ViewBag.Status = -1;**  **}**  **ViewBag.StatusMessage =**  **message == ManageMessageId.RemoveLoginSuccess ? "O login externo foi removido com sucesso"**  **: message == ManageMessageId.Error ? "Ocorreu uma exceção ao processar sua solicitação."**  **: message == ManageMessageId.LinkLoginSuccess? "O login externo foi atribuído com sucesso."**  **: "";**    **var userLogins = await \_userManager.GetLoginsAsync(User.Identity.GetUserId());**  **var otherLogins = AuthenticationManager.GetExternalAuthenticationTypes().Where(auth => userLogins.All(ul => auth.AuthenticationType != ul.LoginProvider)).ToList();**  **ViewBag.ShowRemoveButton = user.PasswordHash != null || userLogins.Count > 1;**  **return View(new ManageLoginsViewModel**  **{**  **CurrentLogins = userLogins,**  **OtherLogins = otherLogins,**  **Callback = (List<string>)TempData["CallbackError"]**  **});**  **}** |

Criar a Viewmodel ManageLoginsViewModel na pasta ViewModels/ManageViewModels do projeto Infra.CrossCutting.Identity

ManageLoginsViewModel

|  |
| --- |
| **using System.Collections.Generic;**  **using Microsoft.AspNet.Identity;**  **using Microsoft.Owin.Security;**  **namespace NewLearningCloud.Infra.CrossCutting.Identity.ViewModels.ManageViewModels**  **{**  **public class ManageLoginsViewModel**  **{**  **public IList<UserLoginInfo> CurrentLogins { get; set; }**  **public IList<AuthenticationDescription> OtherLogins { get; set; }**  **public List<string> Callback { get; set; }**  **}**  **}** |

View ManageLogins (Emty com ManageLoginsViewModel (LearningCloud.Infra.CrossCutting.Identity.ViewModels.ManageViewModels))



|  |
| --- |
| **@model NewLearningCloud.Infra.CrossCutting.Identity.ViewModels.ManageViewModels.ManageLoginsViewModel**  **@using Microsoft.Owin.Security**  **@{**  **ViewBag.Title = "Gerencie Logins Externos";**  **int \_status = ViewBag.Status;**  **}**  **<h2>@ViewBag.Title</h2>**  **@if (!string.IsNullOrEmpty(ViewBag.StatusMessage))**  **{**  **if (\_status < 0)**  **{**  **<div class="alert alert-danger alert-dismissible fade in" role="alert">**  **<button type="button" class="close" data-dismiss="alert" aria-label="Fechar">**  **<span aria-hidden="true">×</span>**  **</button>**  **<h5><strong>@ViewBag.StatusMessage</strong></h5>**  **<ul>**  **@if (Model.Callback != null)**  **{**  **foreach (var error in Model.Callback)**  **{**  **<li>@error</li>**  **}**  **}**  **</ul>**  **</div>**  **}**  **else if (\_status > 0)**  **{**  **<div class="alert alert-warning alert-dismissible fade in" role="alert">**  **<button type="button" class="close" data-dismiss="alert" aria-label="Fechar">**  **<span aria-hidden="true">×</span>**  **</button>**  **<h5><strong>@ViewBag.StatusMessage</strong></h5>**  **</div>**  **}**  **else**  **{**  **<div class="alert alert-success alert-dismissible fade in" role="alert">**  **<button type="button" class="close" data-dismiss="alert" aria-label="Fechar">**  **<span aria-hidden="true">×</span>**  **</button>**  **<h5><strong>@ViewBag.StatusMessage</strong></h5>**  **</div>**  **}**  **}**  **<p class="text-success">@ViewBag.StatusMessage</p>**  **@{**  **var loginProviders = Context.GetOwinContext().Authentication.GetExternalAuthenticationTypes();**  **if (loginProviders.Count() == 0)**  **{**  **<div>**  **<p>**  **O serviço de logins com contas externas está desetivado. Entre em contato com o administrador do sistema.**  **</p>**  **</div>**  **}**  **else**  **{**  **if (Model.CurrentLogins.Count > 0)**  **{**  **<h4>Registered Logins</h4>**  **<table class="table">**  **<tbody>**  **@foreach (var account in Model.CurrentLogins)**  **{**  **<tr>**  **<td>@account.LoginProvider</td>**  **<td>**  **@if (ViewBag.ShowRemoveButton)**  **{**  **using (Html.BeginForm("RemoveLogin", "Manage"))**  **{**  **@Html.AntiForgeryToken()**  **<div>**  **@Html.Hidden("loginProvider", account.LoginProvider)**  **@Html.Hidden("providerKey", account.ProviderKey)**  **<input type="submit" class="btn btn-default" value="Remover" title="Remover o login @account.LoginProvider de sua conta LearningCloud" />**  **</div>**  **}**  **}**  **else**  **{**  **@: &nbsp;!**  **}**  **</td>**  **</tr>**  **}**  **</tbody>**  **</table>**  **}**  **if (Model.OtherLogins.Count > 0)**  **{**  **using (Html.BeginForm("LinkLogin", "Manage"))**  **{**  **@Html.AntiForgeryToken()**  **<div id="socialLoginList">**  **<p>**  **@foreach (AuthenticationDescription p in Model.OtherLogins)**  **{**  **<button type="submit" class="btn btn-default" id="@p.AuthenticationType" name="provider" value="@p.AuthenticationType" title="Entrar usando sua conta @p.Caption">@p.AuthenticationType</button>**  **}**  **</p>**  **</div>**  **}**  **}**  **}**  **}** |

ActionResult RemoveLogin

|  |
| --- |
| **// POST: /Manage/RemoveLogin**  **[HttpPost]**  **[Route("Conta/GerenciarLogins/Excluir")]**  **[ValidateAntiForgeryToken]**  **public async Task<ActionResult> RemoveLogin(string loginProvider, string providerKey)**  **{**  **ManageMessageId? message;**  **var result = await \_userManager.RemoveLoginAsync(User.Identity.GetUserId(), new UserLoginInfo(loginProvider, providerKey));**  **if (result.Succeeded)**  **{**  **var user = await \_userManager.FindByIdAsync(User.Identity.GetUserId());**  **if (user != null)**  **{**  **await \_signInManager.SignInAsync(user, isPersistent: false, rememberBrowser: false);**  **}**  **message = ManageMessageId.RemoveLoginSuccess;**  **}**  **else**  **{**  **message = ManageMessageId.Error;**  **}**  **return RedirectToAction("ManageLogins", new { Message = message });**  **}** |

ActionResult LinkLogin LinkLoginCallback

ActionResult LinkLogin

|  |
| --- |
| // POST: /Manage/LinkLogin  [HttpPost]  [Route("Conta/GerenciarLogins/Atribuir")]  [ValidateAntiForgeryToken]  public ActionResult LinkLogin(string provider)  {  // Request a redirect to the external login provider to link a login for the current user  return new AccountController.ChallengeResult(provider, Url.Action("LinkLoginCallback", "Manage"), User.Identity.GetUserId());  } |

ActionResult LinkLoginCallback

|  |
| --- |
| // GET: /Manage/LinkLoginCallback  [Route("Conta/GerenciarLogins/Atribuir/Retorno")]  public async Task<ActionResult> LinkLoginCallback()  {  var loginInfo = await AuthenticationManager.GetExternalLoginInfoAsync(XsrfKey, User.Identity.GetUserId());  if (loginInfo == null)  {  return RedirectToAction("ManageLogins", new { Message = ManageMessageId.Error });  }  var result = await \_userManager.AddLoginAsync(User.Identity.GetUserId(), loginInfo.Login);  List<string> callbackError = new List<string>();  foreach (string error in result.Errors)  {  callbackError.Add(error);  }  TempData["CallbackError"] = callbackError;  return result.Succeeded ? RedirectToAction("ManageLogins", new { Message = ManageMessageId.LinkLoginSuccess }) : RedirectToAction("ManageLogins", new { Message = ManageMessageId.Error });  } |

Vamos traduzir as mensagens de erro retornadas pelo identity

Instalar o pacote de tradução para o identity

* Package Manager Console
  + seleciona o *Default project* (LearningCloud.Infra.CrossCutting.Identity)
  + Install-Package Microsoft.AspNet.Identity.Core.pt-br

No Web config da raiz do projeto alterar o seguinte código

|  |
| --- |
| <system.web>  **<globalization uiCulture="pt-BR" culture="pt-BR" />**  <compilation debug="true" targetFramework="4.5.2" /> |

Vamos criar algumas páginas de erro personalizadas, ainda no Web config da raiz do projeto alterar o seguinte código

|  |
| --- |
| **<system.web>**  **<globalization uiCulture="pt-BR" culture="pt-BR" />**  **<compilation debug="true" targetFramework="4.5.2" />**  **<customErrors mode="On" defaultRedirect="~/CustomError/Oops">**  **<error statusCode="404" redirect="~/CustomError/NotFound" />**  **<error statusCode="403" redirect="~/CustomError/AccessDenied" />**  **</customErrors>** |

Vamos criar um controller na pasta de controler da raiz do projeto chamado **CustomError**

**CustomErrorController**

|  |
| --- |
| **using System.Web.Mvc;**  **namespace NewLearningCloud.MVC.Controllers**  **{**  **public class CustomErrorController : Controller**  **{**  **public ActionResult Oops()**  **{**  **return View();**  **}**  **public ActionResult NotFound()**  **{**  **return View();**  **}**  **public ActionResult AccessDenied()**  **{**  **return View();**  **}**  **}**  **}** |

Vamos criar as views **Oops, NotFound e AccessDenied**

View Oops

|  |
| --- |
| **@{**  **ViewBag.Title = "Algo de errado não está certo.";**  **}**  **<div class="row">**  **<div class="col-md-2 col-md-offset-1" style="padding:50px 0 50px 50px; border:1px none #ff0000">**  **<img src="~/Content/Images/img\_icons/icons8-Tornado-100.png" width="100" height="100" />**  **</div>**  **<div class="col-md-9" style="border:1px none #4cff00">**  **<hr />**  **<h1>Oops!</h1>**  **<h3>Algo de errado não está certo.</h3>**  **<h4>Ocorreu uma exceção ao processar sua solicitação.</h4>**  **<div>&nbsp;</div>**  **</div>**  **</div>** |

View NotFound

|  |
| --- |
| **@{**  **ViewBag.Title = "Página não encontrada";**  **}**  **<div class="row">**  **<div class="col-md-2 col-md-offset-1" style="padding:50px 0 50px 50px; border:1px none #ff0000">**  **<img src="~/Content/Images/img\_icons/icons8-Windy Weather-100.png" width="100" height="100" />**  **</div>**  **<div class="col-md-9" style="border:1px none #4cff00">**  **<hr />**  **<h1>Oops!</h1>**  **<h3>A página que você procura não foi encontrada.</h3>**  **<div>&nbsp;</div>**  **<h4>Possíveis Motivos:</h4>**  **<ul>**  **<li>O conteúdo não está mais no ar;</li>**  **<li>A página mudou de lugar;</li>**  **<li>Você digitou o endereço errado.</li>**  **</ul>**  **</div>**  **</div>** |

View AccessDenied

|  |
| --- |
| **@{**  **ViewBag.Title = "Acesso Negado";**  **}**  **<div class="row">**  **<div class="col-md-2 col-md-offset-1" style="padding:50px 0 50px 50px; border:1px none #ff0000">**  **<img src="~/Content/Images/img\_icons/icons8-Partly Cloudy Rain-100.png" width="100" height="100" />**  **</div>**  **<div class="col-md-9" style="border:1px none #4cff00">**  **<hr />**  **<h1>Acesso Negado!</h1>**  **<h3>Acreditamos que você não deveria estar nesta página.</h3>**  **<div>&nbsp;</div>**  **</div>**  **</div>** |

Vamos adicionar as ActionResult AddPhoneNumber e VerifyPhoneNumber no ManageController e as viewmodels AddPhoneNumberViewModel e VerifyPhoneNumberViewModel na pasta ViewModels/ManageViewModels do projeto Infra.CrossCutting.Identity

ActionResult AddPhoneNumber GET

|  |
| --- |
| **// GET: /Manage/AddPhoneNumber**  **[Route("Conta/Celular")]**  **public ActionResult AddPhoneNumber()**  **{**  **return View();**  **}** |

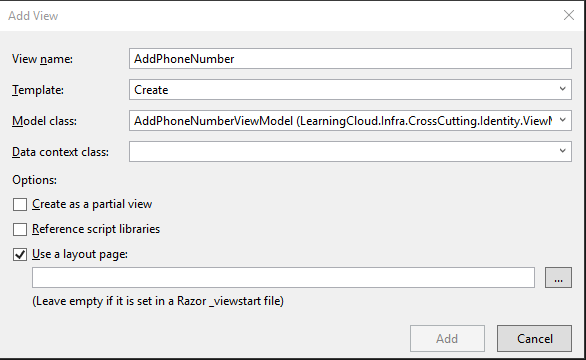
ActionResult AddPhoneNumber POST

|  |
| --- |
| **// POST: /Manage/AddPhoneNumber**  **[HttpPost]**  **[Route("Conta/Celular")]**  **[ValidateAntiForgeryToken]**  **public async Task<ActionResult> AddPhoneNumber(AddPhoneNumberViewModel model)**  **{**  **if (!ModelState.IsValid)**  **{**  **return View(model);**  **}**  **// Generate the token and send it**  **var code = await \_userManager.GenerateChangePhoneNumberTokenAsync(User.Identity.GetUserId(), model.UsuarioAcesso\_PhoneNumber);**  **if (\_userManager.SmsService != null)**  **{**  **var message = new IdentityMessage**  **{**  **Destination = model.Number,**  **Body = "Seu código de segurança é: " + code**  **};**  **await \_userManager.SmsService.SendAsync(message);**  **}**  **return RedirectToAction("VerifyPhoneNumber", new { PhoneNumber = model.UsuarioAcesso\_PhoneNumber });**  **}** |

ViewModel AddPhoneNumberViewModel

|  |
| --- |
| **using System.ComponentModel.DataAnnotations;**  **namespace NewLearningCloud.Infra.CrossCutting.Identity.ViewModels.ManageViewModels**  **{**  **public class AddPhoneNumberViewModel**  **{**  **[Required(ErrorMessage = "Preencha o campo Celular.")]**  **[Phone]**  **[Display(Name = "Celular")]**  **public string UsuarioAcesso\_PhoneNumber{ get; set; }**  **}**  **}** |

Adicionar a View AddPhoneNumber (Create com AddPhoneNumberViewModel (LearningCloud.Infra.CrossCutting.Identity.ViewModels.ManageViewModels))



Alterar os seguintes códigos em destaque:

|  |
| --- |
| **@model NewLearningCloud.Infra.CrossCutting.Identity.ViewModels.ManageViewModels.AddPhoneNumberViewModel**  **@{**  **ViewBag.Title = "Adicionar Celular";**  **}**  **<h2>@ViewBag.Title</h2>**  **@using (Html.BeginForm())**  **{**  **@Html.AntiForgeryToken()**    **<div class="form-horizontal">**  **<h4>AddPhoneNumberViewModel</h4>**  **<hr />**  **@Html.ValidationSummary(true, "", new { @class = "text-danger" })**  **<div class="form-group">**  **@Html.LabelFor(model => model.UsuarioAcesso\_PhoneNumber, htmlAttributes: new { @class = "control-label col-md-2" })**  **<div class="col-md-10">**  **@Html.EditorFor(model => model.UsuarioAcesso\_PhoneNumber, new { htmlAttributes = new { @class = "form-control" } })**  **@Html.ValidationMessageFor(model => model.UsuarioAcesso\_PhoneNumber, "", new { @class = "text-danger" })**  **</div>**  **</div>**  **<div class="form-group">**  **<div class="col-md-offset-2 col-md-10">**  **<input type="submit" value="Enviar código de verificação" class="btn btn-default" />**  **</div>**  **</div>**  **</div>**  **}**  **<div>**  **@Html.ActionLink("Back to List", "Index")**  **</div>** |

Vamos adicionar a ActionResult VerifyPhoneNumber GET

|  |
| --- |
| **// GET: /Manage/VerifyPhoneNumber**  **[Route("Conta/Celular/Verificar/{phoneNumber}")]**  **public async Task<ActionResult> VerifyPhoneNumber(string phoneNumber)**  **{**  **// This code allows you exercise the flow without actually sending codes**  **// For production use please register a SMS provider in IdentityConfig and generate a code here.**  **var code = await \_userManager.GenerateChangePhoneNumberTokenAsync(User.Identity.GetUserId(), phoneNumber);**  **ViewBag.Status = "DEMO: Caso o código não chegue via SMS o código é: ";**  **ViewBag.CodigoAcesso = code;**  **return phoneNumber == null ? View("Error") : View(new VerifyPhoneNumberViewModel { UsuarioAcesso\_PhoneNumber = phoneNumber });**  **}** |

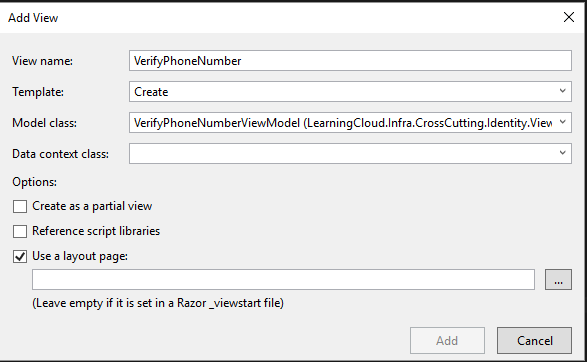
ActionResult VerifyPhoneNumber POST

|  |
| --- |
| **// POST: /Manage/VerifyPhoneNumber**  **[HttpPost]**  **[Route("Conta/Celular/Verificar/{phoneNumber?}")]**  **[ValidateAntiForgeryToken]**  **public async Task<ActionResult> VerifyPhoneNumber(VerifyPhoneNumberViewModel model)**  **{**  **if (!ModelState.IsValid)**  **{**  **return View(model);**  **}**  **var result = await \_userManager.ChangePhoneNumberAsync(User.Identity.GetUserId(), model.UsuarioAcesso\_PhoneNumber, model.Code);**  **if (result.Succeeded)**  **{**  **var user = await \_userManager.FindByIdAsync(User.Identity.GetUserId());**  **if (user != null)**  **{**  **await \_signInManager.SignInAsync(user, isPersistent: false, rememberBrowser: false);**  **}**  **return RedirectToAction("Index", new { Message = ManageMessageId.AddPhoneSuccess });**  **}**  **ViewBag.Status = "DEMO: Caso o código não chegue via SMS o código é: ";**  **ViewBag.CodigoAcesso = model.Code;**  **// No caso de falha, reexibir a view.**  **ModelState.AddModelError("", "Problemas ao adicionar celular.");**  **foreach (var item in result.Errors)**  **{**  **ModelState.AddModelError("", item);**  **}**  **return View(model);**  **}** |

VerifyPhoneNumberViewModel

|  |
| --- |
| **using System.ComponentModel.DataAnnotations;**  **namespace NewLearningCloud.Infra.CrossCutting.Identity.ViewModels.ManageViewModels**  **{**  **public class VerifyPhoneNumberViewModel**  **{**  **[Required(ErrorMessage = "Preencha o campo Código.")]**  **[Display(Name = "Código")]**  **public string Code { get; set; }**  **[ScaffoldColumn(false)]**  **[Required(ErrorMessage = "Problemas ao recuperar o número do celular. Clique em Voltar e informe novamente seu número.")]**  **public string UsuarioAcesso\_PhoneNumber { get; set; }**  **}**  **}** |

Vamos adicionar a View para (Create com VerifyPhoneNumberViewModel (LearningCloud.Infra.CrossCutting.Identity.ViewModels.ManageViewModels))



Alterar os seguintes trechos em destaque

|  |
| --- |
| **@model LearningCloud.Infra.CrossCutting.Identity.ViewModels.ManageViewModels.VerifyPhoneNumberViewModel**  **@{**  **ViewBag.Title = "Verificar número de celular";**  **}**  **<h2>@ViewBag.Title</h2>**  **@using (Html.BeginForm("VerifyPhoneNumber", "Manage", FormMethod.Post, new { @class = "form-horizontal", role = "form" }))**  **{**  **@Html.AntiForgeryToken()**    **<div class="form-horizontal">**  **@Html.Hidden("phoneNumber", @Model.UsuarioAcesso\_PhoneNumber)**  **@Html.ValidationMessageFor(model => model.UsuarioAcesso\_PhoneNumber, "", new { @class = "text-danger" })**  **<h4>Digite seu código de segurança.</h4>**  **<hr />**  **@Html.ValidationSummary(true, "", new { @class = "text-danger" })**  **<div class="form-group">**  **@Html.LabelFor(model => model.Code, htmlAttributes: new { @class = "control-label col-md-4" })**  **<div class="col-md-8">**  **@Html.EditorFor(model => model.Code, new { htmlAttributes = new { @class = "form-control" } })**  **@Html.ValidationMessageFor(model => model.Code, "", new { @class = "text-danger" })**  **</div>**  **</div>**  **<div class="form-group">**  **<div class="col-md-offset-4 col-md-8">**  **<input type="submit" value="Enviar" class="btn btn-default" />**  **<h5>@ViewBag.Status &nbsp;<strong>@ViewBag.CodigoAcesso</strong></h5>**  **</div>**  **</div>**  **</div>**  **}**  **<div>**  **@Html.ActionLink("Back to List", "Index")**  **</div>** |

Vamos adicionar as ActionResult **RemovePhoneNumber**, **RememberBrowser**, **ForgetBrowser**, **EnableTwoFactorAuthentication** e **DisableTwoFactorAuthentication**

ActionResult **RemovePhoneNumber**

|  |
| --- |
| **// GET: /Manage/RemovePhoneNumber**  **[Route("Conta/Celular/Remover")]**  **public async Task<ActionResult> RemovePhoneNumber()**  **{**  **var result = await \_userManager.SetPhoneNumberAsync(User.Identity.GetUserId(), null);**  **if (!result.Succeeded)**  **{**  **return RedirectToAction("Index", new { Message = ManageMessageId.Error });**  **}**  **var user = await \_userManager.FindByIdAsync(User.Identity.GetUserId());**  **if (user != null)**  **{**  **await \_signInManager.SignInAsync(user, isPersistent: false, rememberBrowser: false);**  **}**  **return RedirectToAction("Index", new { Message = ManageMessageId.RemovePhoneSuccess });**  **}** |

ActionResult **RememberBrowser**

|  |
| --- |
| **// POST: /Manage/RememberBrowser**  **[HttpPost]**  **//[Route("Conta/LembrarNavegador")]**  **[ValidateAntiForgeryToken]**  **public ActionResult RememberBrowser()**  **{**  **var rememberBrowserIdentity = AuthenticationManager.CreateTwoFactorRememberBrowserIdentity(User.Identity.GetUserId());**  **AuthenticationManager.SignIn(new AuthenticationProperties { IsPersistent = true }, rememberBrowserIdentity);**  **return RedirectToAction("Index", "Manage");**  **}** |

ActionResult **ForgetBrowser**

|  |
| --- |
| **// POST: /Manage/ForgetBrowser**  **[HttpPost]**  **//[Route("Conta/EsquecerNavegador")]**  **[ValidateAntiForgeryToken]**  **public ActionResult ForgetBrowser()**  **{**  **AuthenticationManager.SignOut(DefaultAuthenticationTypes.TwoFactorRememberBrowserCookie);**  **return RedirectToAction("Index", "Manage");**  **}** |

ActionResult **EnableTwoFactorAuthentication**

|  |
| --- |
| **// POST: /Manage/EnableTwoFactorAuthentication**  **[HttpPost]**  **[Route("Conta/HabilitarDoisFatores")]**  **[ValidateAntiForgeryToken]**  **public async Task<ActionResult> EnableTwoFactorAuthentication()**  **{**  **await \_userManager.SetTwoFactorEnabledAsync(User.Identity.GetUserId(), true);**  **var user = await \_userManager.FindByIdAsync(User.Identity.GetUserId());**  **if (user != null)**  **{**  **await \_signInManager.SignInAsync(user, isPersistent: false, rememberBrowser: false);**  **}**  **return RedirectToAction("Index", "Manage");**  **}** |

ActionResult **DisableTwoFactorAuthentication**

|  |
| --- |
| **// POST: /Manage/DisableTwoFactorAuthentication**  **[HttpPost]**  **[Route("Conta/DesabilitarDoisFatores")]**  **[ValidateAntiForgeryToken]**  **public async Task<ActionResult> DisableTwoFactorAuthentication()**  **{**  **await \_userManager.SetTwoFactorEnabledAsync(User.Identity.GetUserId(), false);**  **var user = await \_userManager.FindByIdAsync(User.Identity.GetUserId());**  **if (user != null)**  **{**  **await \_signInManager.SignInAsync(user, isPersistent: false, rememberBrowser: false);**  **}**  **return RedirectToAction("Index", "Manage");**  **}** |

RolesAdminController na área admin

|  |
| --- |
| using System.Web.Mvc;  using NewLearningCloud.Infra.CrossCutting.Identity.Configuration;  namespace NewLearningCloud.MVC.Areas.Admin.Controllers  {  public class RolesAdminController : Controller  {  private readonly ApplicationRoleManager \_roleManager;  private ApplicationUserManager \_userManager;  public RolesAdminController(ApplicationUserManager userManager, ApplicationRoleManager roleManager)  {  \_userManager = userManager;  \_roleManager = roleManager;  }  // GET: Admin/RolesAdmin  public ActionResult Index()  {  return View();  }  }  } |

Infra.CrossCutting.Identity.Configuration

criar ApplicationRoleManager public

|  |
| --- |
| using Microsoft.Owin;  using Microsoft.AspNet.Identity;  using Microsoft.AspNet.Identity.EntityFramework;  using Microsoft.AspNet.Identity.Owin;  using NewLearningCloud.Infra.CrossCutting.Identity.ContextIdentity;  namespace NewLearningCloud.Infra.CrossCutting.Identity.Configuration  {  public class ApplicationRoleManager : RoleManager<IdentityRole>  {  public ApplicationRoleManager(IRoleStore<IdentityRole, string> roleStore)  : base(roleStore)  {  }  public static ApplicationRoleManager Create(IdentityFactoryOptions<ApplicationRoleManager> options, IOwinContext context)  {  return new ApplicationRoleManager(new RoleStore<IdentityRole>(context.Get<ApplicationDbContext>()));  }  }  } |

BootStrapper.cs

container.Register<ApplicationRoleManager>(Lifestyle.Scoped);

Alterar controller RolesAdminController

Index PAnel

<ul class="nav navbar-nav">

<li><a href="@Url.Action("Index", "Aula", new { Area = "Admin" })"><i class="fa fa-file-video-o">&nbsp;&nbsp;</i>Aulas</a></li>

<li><a href="@Url.Action("Index", "RolesAdmin", new { Area = "Admin" })"><i class="fa fa-file-video-o">&nbsp;&nbsp;</i>Papeis no Sistema</a></li>

</ul>

Criar View para index da RolesAdminController

Index Empty (without model)

|  |
| --- |
| @model IEnumerable<Microsoft.AspNet.Identity.EntityFramework.IdentityRole>  @{  ViewBag.Title = "Index";  }  <h2>Index</h2>  <p>  @Html.ActionLink("Criar Nova", "Create")  </p>  <table class="table">  <tr>  <th>  @Html.DisplayNameFor(model => model.Name)  </th>  <th>  </th>  </tr>  @foreach (var item in Model)  {  <tr>  <td>  @Html.DisplayFor(modelItem => item.Name)  </td>  <td>  @Html.ActionLink("Editar", "Edit", new { id = item.Id }) |  @Html.ActionLink("Detalhes", "Details", new { id = item.Id }) |  @Html.ActionLink("Deletar", "Delete", new { id = item.Id })  </td>  </tr>  }  </table> |

-----------------------------------------

Alterar view index

@Html.ActionLink("Trocar Senha", "ChangePassword", null, new { @class = "btn btn-default btn-block" })

<a href="@Url.Action("ChangePassword")" class="btn btn-default btn-block"><i class="fa fa-key">&nbsp;&nbsp;</i>Trocar Senha</a>

@Html.ActionLink("Definir senha para conta local", "SetPassword", null, new { @class = "btn btn-default btn-block" })

<a href="@Url.Action("SetPassword")" class="btn btn-default btn-block"><i class="fa fa-key">&nbsp;&nbsp;</i>Definir Senha para Conta Local</a>

@Html.ActionLink("Gerenciar", "ManageLogins", null, new { @class = "btn btn-default btn-block" })

<a href="@Url.Action("ManageLogins")" class="btn btn-default btn-block"><i class="fa fa-cogs">&nbsp;&nbsp;</i>Gerenciar</a>

@Html.ActionLink("Trocar", "AddPhoneNumber", null, new { @class = "btn btn-default btn-block" })

<a href="@Url.Action("AddPhoneNumber")" class="btn btn-default btn-block"><i class="fa fa-exchange">&nbsp;&nbsp;</i>Trocar</a>

@Html.ActionLink("Remover", "RemovePhoneNumber", null, new { @class = "btn btn-danger btn-block" })

<a href="@Url.Action("RemovePhoneNumber")" class = "btn btn-danger btn-block"><i class="fa fa-trash-o">&nbsp;&nbsp;</i>Remover</a>

@Html.ActionLink("Adicionar", "AddPhoneNumber", null, new { @class = "btn btn-default btn-block" })

<a href="@Url.Action("AddPhoneNumber")" class="btn btn-default btn-block"><i class="fa fa-mobile">&nbsp;&nbsp;</i>Adicionar</a>

@Html.ActionLink("Confirmar", "DisplayEmail", "Account", new { Id = @User.GetInfoUser("UserAccessId") }, new { @class = "btn btn-default btn-block" })

<a href="@Url.Action("DisplayEmail","Account", new { Id = @User.GetInfoUser("UserAccessId") })" class="btn btn-success btn-block"><i class="fa fa-envelope-o">&nbsp;&nbsp;</i>Confirmar</a>

@Html.ActionLink("Alterar", "EditEmail", "Account", new { Id = @User.GetInfoUser("UserAccessId") }, new { @class = "btn btn-default btn-block disabled" })

<a href="@Url.Action("EditEmail","Account", new { Id = @User.GetInfoUser("UserAccessId") })" class="btn btn-default btn-block disabled"><i class="fa fa-envelope-open-o">&nbsp;&nbsp;</i>Alterar</a>

<input type="submit" value="Desabilitar" class="btn btn-warning btn-block" />

<button type="submit" class="btn btn-warning btn-block"><i class="fa fa-toggle-off">&nbsp;&nbsp;</i>Desabilitar</button>

<input type="submit" value="Habilitar" class="btn btn-default btn-block" />

<button type="submit" class="btn btn-default btn-block"><i class="fa fa-toggle-on">&nbsp;&nbsp;</i>Habilitar</button>

<input type="submit" value="Lembrar Browser" class="btn btn-default btn-block" />

<button type="submit" class="btn btn-default btn-block"><i class="fa fa-toggle-on">&nbsp;&nbsp;</i>Lembrar Browser</button>

<input type="submit" value="Esquecer Browser" class="btn btn-default btn-block" />

<button type="submit" class="btn btn-default btn-block"><i class="fa fa-toggle-off">&nbsp;&nbsp;</i>Esquecer Browser</button>

-------------------------

Site.css

/\* Set width on the form input elements since they're 100% wide by default \*/

input,

select,

textarea {

/\***max-width: 280px;\*/**

}

.btn-labeled {padding-top:0; padding-bottom:0;}

.btn-label {position:relative; left:-12px; display:inline-block; padding:6px 12px; background:rgba(0,0,0,0.15); border-radius:3px 0 0 3px; border:1px none #4cff00; width:40px}

.btn-label-text{display: inline-block; text-align:left; min-width:210px; border:1px none #00f7ff;}

.btn-label-textmini {display: inline-block; text-align:left; min-width:70px; border:1px none #00f7ff;}

------------------------------

View Index

|  |
| --- |
| @model NewLearningCloud.Infra.CrossCutting.Identity.ViewModels.ManageViewModels.IndexViewModel  @using NewLearningCloud.Infra.CrossCutting.Identity.Configuration;  @{  ViewBag.Title = "Gerenciar Conta";  int \_status = ViewBag.Status;  }  <h2>@ViewBag.Title</h2>  @if (!string.IsNullOrEmpty(ViewBag.StatusMessage))  {  if (\_status < 0)  {  <div class="alert alert-danger alert-dismissible fade in" role="alert">  <button type="button" class="close" data-dismiss="alert" aria-label="Fechar">  <span aria-hidden="true">×</span>  </button>  <h5><strong>@ViewBag.StatusMessage</strong></h5>  </div>  }  else if (\_status > 0)  {  <div class="alert alert-warning alert-dismissible fade in" role="alert">  <button type="button" class="close" data-dismiss="alert" aria-label="Fechar">  <span aria-hidden="true">×</span>  </button>  <h5><strong>@ViewBag.StatusMessage</strong></h5>  </div>  }  else  {  <div class="alert alert-success alert-dismissible fade in" role="alert">  <button type="button" class="close" data-dismiss="alert" aria-label="Fechar">  <span aria-hidden="true">×</span>  </button>  <h5><strong>@ViewBag.StatusMessage</strong></h5>  </div>  }  }  <table class="table table-hover">  <tbody>  <tr>  <th scope="row">Usuário:</th>  <td><span style="padding:5px 15px;" class="badge badge-default">@User.Identity.Name</span></td>  <td><a href="@Url.Action("EditUserName","Account", new { Id = @User.GetInfoUser("UserAccessId") })" class="btn btn-default btn-labeled disabled"><span class="btn-label"><i class="fa fa fa-user">&nbsp;&nbsp;</i></span><span class="btn-label-text">Alterar Usúario</span></a></td>  </tr>  <tr>  <th scope="row">Senha:</th>  @if (Model.HasPassword)  {  <td><span class="badge badge-default" style="padding:6px 5px 0 5px;"><text class="text-center" style="font-size:13pt;">\* \* \* \* \* \* \* \* \* \* \* \*</text></span></td>  <td><a href="@Url.Action("ChangePassword")" class="btn btn-default btn-labeled"><span class="btn-label"><i class="fa fa-key">&nbsp;&nbsp;</i></span><span class="btn-label-text">Trocar Senha</span></a></td>  }  else  {  <td>Não foi definida uma senha para conta local.</td>  <td><a href="@Url.Action("SetPassword")" class="btn btn-default btn-labeled"><span class="btn-label"><i class="fa fa-key">&nbsp;&nbsp;</i></span><span class="btn-label-text">Definir Senha para Conta Local</span></a></td>  }  </tr>  <tr>  <th scope="row">Logins Externos:</th>  <td>Logins Sociais: <span style="padding:5px 15px;" class="badge badge-default @(Model.Logins.Count > 0 ? "badge-info": "badge-warning")">@Model.Logins.Count</span></td>  <td><a href="@Url.Action("ManageLogins")" class="btn btn-default btn-labeled"><span class="btn-label"><i class="fa fa-cogs">&nbsp;&nbsp;</i></span><span class="btn-label-text">Gerenciar Logins</span></a></td>  </tr>  <tr>  <th scope="row">Número de celular:</th>  @if (Model.PhoneNumber != null)  {  <td><span style="padding:5px 15px;" class="badge badge-default">@Model.PhoneNumber</span></td>  <td>  <div style="max-width:137px; float:left; padding:0 2px 0 0; border:1px none #ff0000;">  <a href="@Url.Action("AddPhoneNumber")" class="btn btn-default btn-labeled"><span class="btn-label"><i class="fa fa-exchange">&nbsp;&nbsp;</i></span><span class="btn-label-textmini">Trocar</span></a>  </div>  <div style="max-width:137px; float:left; padding:0 0 0 2px;">  <a href="@Url.Action("RemovePhoneNumber")" class="btn btn-danger btn-labeled"><span class="btn-label"><i class="fa fa-trash-o">&nbsp;&nbsp;</i></span><span class="btn-label-textmini">Remover</span></a>  </div>  </td>  }  else  {  <td>  Não Informado.  </td>  <td><a href="@Url.Action("AddPhoneNumber")" class="btn btn-default btn-labeled"><span class="btn-label"><i class="fa fa-mobile">&nbsp;&nbsp;</i></span><span class="btn-label-text">Adicionar Celular</span></a></td>  }  </tr>  <tr>  <th scope="row">E-mail:</th>  @if (!Model.EmailConfirmed)  {  <td>  E-mail: <span style="padding:5px 15px;" class="badge badge-default">@Model.Email</span> Não confirmado.  </td>  <td>  <a href="@Url.Action("DisplayEmail","Account", new { Id = @User.GetInfoUser("UserAccessId") })" class="btn btn-success btn-labeled"><span class="btn-label"><i class="fa fa-envelope-o">&nbsp;&nbsp;</i></span><span class="btn-label-text">Confirmar E-mail</span></a>  </td>  }  else  {  <td><span style="padding:5px 15px;" class="badge badge-default">@Model.Email</span></td>  <td>  <a href="@Url.Action("EditEmail","Account", new { Id = @User.GetInfoUser("UserAccessId") })" class="btn btn-default btn-labeled disabled"><span class="btn-label"><i class="fa fa-envelope-open-o">&nbsp;&nbsp;</i></span><span class="btn-label-text">Alterar E-mail</span></a>  </td>  }  </tr>  @if (Model.EmailConfirmed || Model.PhoneNumber != null)  {  <tr>  <th scope="row">Autenticação de dois fatores:</th>  @if (Model.TwoFactor)  {  <td>Two Factor está habilitado.</td>  using (Html.BeginForm("DisableTwoFactorAuthentication", "Manage", FormMethod.Post, new { @class = "form-horizontal", role = "form" }))  {  @Html.AntiForgeryToken()  <td>  <button type="submit" class="btn btn-default btn-labeled"><span class="btn-label"><i class="fa fa-toggle-off">&nbsp;&nbsp;</i></span><span class="btn-label-text">Desabilitar Two Factor</span></button>  </td>  }  }  else  {  <td>Two Factor NÃO está habilitado.</td>  using (Html.BeginForm("EnableTwoFactorAuthentication", "Manage", FormMethod.Post, new { @class = "form-horizontal", role = "form" }))  {  @Html.AntiForgeryToken()  <td>  <button type="submit" class="btn btn-default btn-labeled"><span class="btn-label"><i class="fa fa-toggle-on">&nbsp;&nbsp;</i></span><span class="btn-label-text">Habilitar Two Factor</span></button>  </td>  }  }  </tr>  <tr>  <th scope="row"></th>  @if (Model.BrowserRemembered)  {  <td>O browser será lembrado para Two Factor.</td>  using (Html.BeginForm("ForgetBrowser", "Manage", FormMethod.Post, new { @class = "form-horizontal", role = "form" }))  {  @Html.AntiForgeryToken()  <td>  <button type="submit" class="btn btn-default btn-labeled"><span class="btn-label"><i class="fa fa-toggle-off">&nbsp;&nbsp;</i></span><span class="btn-label-text">Esquecer Browser</span></button>  </td>  }  }  else  {  <td>O browser NÃO será lembrado para Two Factor.</td>  using (Html.BeginForm("RememberBrowser", "Manage", FormMethod.Post, new { @class = "form-horizontal", role = "form" }))  {  @Html.AntiForgeryToken()  <td>  <button type="submit" class="btn btn-default btn-labeled"><span class="btn-label"><i class="fa fa-toggle-on">&nbsp;&nbsp;</i></span><span class="btn-label-text">Lembrar Browser</span></button>  </td>  }  }  </tr>  }  else  {  <tr>  <th scope="row">Autenticação de dois fatores:</th>  <td colspan="2">  <i style="color: #ffd800;text-shadow: 1px 1px 5px #292929; font-size: 1.2em;" class="fa fa-exclamation-triangle">&nbsp;&nbsp;</i>  <span class="text-warning">Para habilitar a autenticação de de dois fatores adicione um <strong>número de celular</strong> ou <strong>confirme seu e-mail</strong>.</span>  </td>  </tr>  }  </tbody>  </table> |

\_ExternalLoginsListPartial.cshtml

de

|  |
| --- |
| @model NewLearningCloud.Infra.CrossCutting.Identity.ViewModels.AccountViewModels.ExternalLoginListViewModel  @using Microsoft.Owin.Security  <h4>Ou utilize outro serviço para fazer o login:</h4>  <hr />  @{  var loginProviders = Context.GetOwinContext().Authentication.GetExternalAuthenticationTypes();  if (loginProviders.Count() == 0)  {  <div>  <p>  There are no external authentication services configured. See <a href="http://go.microsoft.com/fwlink/?LinkId=403804">this article</a>  for details on setting up this ASP.NET application to support logging in via external services.  </p>  </div>  }  else  {  using (Html.BeginForm("ExternalLogin", "Account", new { ReturnUrl = Model.ReturnUrl }))  {  @Html.AntiForgeryToken()  <div id="socialLoginList">  <p>  @foreach (AuthenticationDescription p in loginProviders)  {  <button type="submit" class="btn btn-default" id="@p.AuthenticationType" name="provider" value="@p.AuthenticationType" title="Entrar utilizando sua conta do @p.Caption">@p.AuthenticationType</button>  }  </p>  </div>  }  }  } |

Para

|  |
| --- |
| @model NewLearningCloud.Infra.CrossCutting.Identity.ViewModels.AccountViewModels.ExternalLoginListViewModel  @using Microsoft.Owin.Security  @{  string classButton = "";  string classButtonIcon = "";  }  <h4>Ou utilize outro serviço para fazer o login:</h4>  <hr />  @{  var loginProviders = Context.GetOwinContext().Authentication.GetExternalAuthenticationTypes();  if (loginProviders.Count() == 0)  {  <div>  <p>  There are no external authentication services configured. See <a href="http://go.microsoft.com/fwlink/?LinkId=403804">this article</a>  for details on setting up this ASP.NET application to support logging in via external services.  </p>  </div>  }  else  {  using (Html.BeginForm("ExternalLogin", "Account", new { ReturnUrl = Model.ReturnUrl }))  {  @Html.AntiForgeryToken()  <div id="socialLoginList">  <p>  @foreach (AuthenticationDescription p in loginProviders)  {  switch (p.AuthenticationType)  {  case "Facebook":  classButton = "btn-primary";  classButtonIcon = "fa-facebook";  break;  case "Google":  classButton = "btn-danger";  classButtonIcon = "fa-google-plus";  break;  default:  classButton = "btn-default";  classButtonIcon = "fa-sign-in";  break;  }  <div class="form-group">  <button type="submit" class="btn btn-labeled @classButton" id="@p.AuthenticationType" name="provider" value="@p.AuthenticationType" title="Entrar utilizando sua conta @p.Caption">  <span class="btn-label">  <i class="fa @classButtonIcon">&nbsp;&nbsp;</i>  </span>  <span class="btn-label-text">Entrar com sua conta @p.AuthenticationType</span>  </button>  </div><!-- /.form-group -->  }  </p>  </div>  }  }  } |

---------------------------

verificar retorno de erros

RemovePhoneNumber

EnableTwoFactorAuthentication

DisableTwoFactorAuthentication

IndexViewModel

public string Email { get; set; }

public bool EmailConfirmed { get; set; }

Index(ManageMessageId? message)

var user = \_userManager.FindById(userId);

EmailConfirmed = user.EmailConfirmed

ApplicationUser

new Claim("UserId",UsuarioAcesso\_Usuario.Usuario\_Id),

new Claim("UserAccessId", Id),

Index(ManageMessageId? message)

Email = user.Email,

EmailConfirmed = user.EmailConfirmed

View Index

@if (!Model.EmailConfirmed)

{

<dt>E-mail:</dt>

<dd>Confirmar E-mail: @Model.Email [ @Html.ActionLink("Confirmar", "DisplayEmail", "Account", new { Id = @User.GetInfoUser("UserAccessId") }, null) ]</dd>

}

@if (Model.EmailConfirmed || Model.PhoneNumber != null)

{

Problemas ao recuperar o número do celular. Clique em Voltar e informe novamente seu número.

Identity fase 2

AccountController

Lembrar

* RoleViewModel
* 1 - Application
  + Interfaces
    - IRoleAppService
  + Services
    - RoleAppService
* 2 - Domain
  + Interfaces
    - Repositories => IRoleRepository
  + Services
    - IRoleService
* Domain / Services
  + RoleService
* Infra.Data
  + ClaimsRolesNewLearningCloudContext
  + RoleRepository
* Infra.CrossCutting.IoC
  + BootStrapper
    - container.Register<IRoleRepository, RoleRepository>(Lifestyle.Scoped);
    - container.Register<IRoleAppService, RoleAppService>(Lifestyle.Scoped);
    - container.Register<IRoleService, RoleService>(Lifestyle.Scoped);

Add-Migration Ignore –IgnoreChanges

update-database -TargetMigration Ignore

DomainToViewModelMappingProfile

public DomainToViewModelMappingProfile()

{

CreateMap<Aula, AulaViewModel>();

CreateMap<Role, RoleViewModel>();

}

ViewModelToDomainMappingProfile

public ViewModelToDomainMappingProfile()

{

CreateMap<AulaViewModel, Aula>();

CreateMap<RoleViewModel, Role>();

}

**@if (User.IsInRole("Admin"))**

**{**

Mudar CSS para colocar o Site.Varsity.cc colocando o box e content-header

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

.content-header {

position: relative;

padding: 15px 15px 0 15px;

}

.content-header > h1 {

margin: 0;

font-size: 24px;

}

.content-header > h1 > small {

font-size: 15px;

display: inline-block;

padding-left: 4px;

font-weight: 300;

}

.content-header > .breadcrumb {

float: right;

background: transparent;

margin-top: 0;

margin-bottom: 0;

font-size: 12px;

padding: 7px 5px;

position: absolute;

top: 15px;

right: 10px;

border-radius: 2px;

}

.content-header > .breadcrumb > li > a {

color: #444;

text-decoration: none;

display: inline-block;

}

.content-header > .breadcrumb > li > a > .fa,

.content-header > .breadcrumb > li > a > .glyphicon,

.content-header > .breadcrumb > li > a > .ion {

margin-right: 5px;

}

.content-header > .breadcrumb > li + li:before {

content: '>\00a0';

}

@media (max-width: 991px) {

.content-header > .breadcrumb {

position: relative;

margin-top: 5px;

top: 0;

right: 0;

float: none;

background: #d2d6de;

padding-left: 10px;

}

.content-header > .breadcrumb li:before {

color: #97a0b3;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*

\* Component: Box

\* --------------

\*/

.box {

position: relative;

border-radius: 3px;

background: #ffffff;

border-top: 3px solid #d2d6de;

/\*margin-bottom: 20px;\*/

margin:10px 50px 20px 50px;

width: 95%;

box-shadow: 0 1px 1px rgba(0, 0, 0, 0.1);

}

.box.box-primary {

border-top-color: #3c8dbc;

}

.box.box-info {

border-top-color: #00c0ef;

}

.box.box-danger {

border-top-color: #dd4b39;

}

.box.box-warning {

border-top-color: #f39c12;

}

.box.box-success {

border-top-color: #00a65a;

}

.box.box-default {

border-top-color: #d2d6de;

}

.box.collapsed-box .box-body,

.box.collapsed-box .box-footer {

display: none;

}

.box .nav-stacked > li {

border-bottom: 1px solid #f4f4f4;

margin: 0;

}

.box .nav-stacked > li:last-of-type {

border-bottom: none;

}

.box.height-control .box-body {

max-height: 300px;

overflow: auto;

}

.box .border-right {

border-right: 1px solid #f4f4f4;

}

.box .border-left {

border-left: 1px solid #f4f4f4;

}

.box.box-solid {

border-top: 0;

}

.box.box-solid > .box-header .btn.btn-default {

background: transparent;

}

.box.box-solid > .box-header .btn:hover,

.box.box-solid > .box-header a:hover {

background: rgba(0, 0, 0, 0.1);

}

.box.box-solid.box-default {

border: 1px solid #d2d6de;

}

.box.box-solid.box-default > .box-header {

color: #444444;

background: #d2d6de;

background-color: #d2d6de;

}

.box.box-solid.box-default > .box-header a,

.box.box-solid.box-default > .box-header .btn {

color: #444444;

}

.box.box-solid.box-primary {

border: 1px solid #3c8dbc;

}

.box.box-solid.box-primary > .box-header {

color: #ffffff;

background: #3c8dbc;

background-color: #3c8dbc;

}

.box.box-solid.box-primary > .box-header a,

.box.box-solid.box-primary > .box-header .btn {

color: #ffffff;

}

.box.box-solid.box-info {

border: 1px solid #00c0ef;

}

.box.box-solid.box-info > .box-header {

color: #ffffff;

background: #00c0ef;

background-color: #00c0ef;

}

.box.box-solid.box-info > .box-header a,

.box.box-solid.box-info > .box-header .btn {

color: #ffffff;

}

.box.box-solid.box-danger {

border: 1px solid #dd4b39;

}

.box.box-solid.box-danger > .box-header {

color: #ffffff;

background: #dd4b39;

background-color: #dd4b39;

}

.box.box-solid.box-danger > .box-header a,

.box.box-solid.box-danger > .box-header .btn {

color: #ffffff;

}

.box.box-solid.box-warning {

border: 1px solid #f39c12;

}

.box.box-solid.box-warning > .box-header {

color: #ffffff;

background: #f39c12;

background-color: #f39c12;

}

.box.box-solid.box-warning > .box-header a,

.box.box-solid.box-warning > .box-header .btn {

color: #ffffff;

}

.box.box-solid.box-success {

border: 1px solid #00a65a;

}

.box.box-solid.box-success > .box-header {

color: #ffffff;

background: #00a65a;

background-color: #00a65a;

}

.box.box-solid.box-success > .box-header a,

.box.box-solid.box-success > .box-header .btn {

color: #ffffff;

}

.box.box-solid > .box-header > .box-tools .btn {

border: 0;

box-shadow: none;

}

.box.box-solid[class\*='bg'] > .box-header {

color: #fff;

}

.box .box-group > .box {

margin-bottom: 5px;

}

.box .knob-label {

text-align: center;

color: #333;

font-weight: 100;

font-size: 12px;

margin-bottom: 0.3em;

}

.box > .overlay,

.overlay-wrapper > .overlay,

.box > .loading-img,

.overlay-wrapper > .loading-img {

position: absolute;

top: 0;

left: 0;

width: 100%;

height: 100%;

}

.box .overlay,

.overlay-wrapper .overlay {

z-index: 50;

background: rgba(255, 255, 255, 0.7);

border-radius: 3px;

}

.box .overlay > .fa,

.overlay-wrapper .overlay > .fa {

position: absolute;

top: 50%;

left: 50%;

margin-left: -15px;

margin-top: -15px;

color: #000;

font-size: 30px;

}

.box .overlay.dark,

.overlay-wrapper .overlay.dark {

background: rgba(0, 0, 0, 0.5);

}

.box-header:before,

.box-body:before,

.box-footer:before,

.box-header:after,

.box-body:after,

.box-footer:after {

content: " ";

display: table;

}

.box-header:after,

.box-body:after,

.box-footer:after {

clear: both;

}

.box-header {

color: #444;

display: block;

padding: 10px;

position: relative;

}

.box-header.with-border {

border-bottom: 1px solid #f4f4f4;

}

.collapsed-box .box-header.with-border {

border-bottom: none;

}

.box-header > .fa,

.box-header > .glyphicon,

.box-header > .ion,

.box-header .box-title {

display: inline-block;

font-size: 18px;

margin: 0;

line-height: 1;

}

.box-header > .fa,

.box-header > .glyphicon,

.box-header > .ion {

margin-right: 5px;

}

.box-header > .box-tools {

position: absolute;

right: 10px;

top: 5px;

}

.box-header > .box-tools [data-toggle="tooltip"] {

position: relative;

}

.box-header > .box-tools.pull-right .dropdown-menu {

right: 0;

left: auto;

}

.box-header > .box-tools .dropdown-menu > li > a {

color: #444!important;

}

.btn-box-tool {

padding: 5px;

font-size: 12px;

background: transparent;

color: #97a0b3;

}

.open .btn-box-tool,

.btn-box-tool:hover {

color: #606c84;

}

.btn-box-tool.btn:active {

box-shadow: none;

}

.box-body {

border-top-left-radius: 0;

border-top-right-radius: 0;

border-bottom-right-radius: 3px;

border-bottom-left-radius: 3px;

padding: 10px;

}

.no-header .box-body {

border-top-right-radius: 3px;

border-top-left-radius: 3px;

}

.box-body > .table {

margin-bottom: 0;

}

.box-body .fc {

margin-top: 5px;

}

.box-body .full-width-chart {

margin: -19px;

}

.box-body.no-padding .full-width-chart {

margin: -9px;

}

.box-body .box-pane {

border-top-left-radius: 0;

border-top-right-radius: 0;

border-bottom-right-radius: 0;

border-bottom-left-radius: 3px;

}

.box-body .box-pane-right {

border-top-left-radius: 0;

border-top-right-radius: 0;

border-bottom-right-radius: 3px;

border-bottom-left-radius: 0;

}

.box-footer {

border-top-left-radius: 0;

border-top-right-radius: 0;

border-bottom-right-radius: 3px;

border-bottom-left-radius: 3px;

border-top: 1px solid #f4f4f4;

padding: 10px;

background-color: #ffffff;

}

.chart-legend {

margin: 10px 0;

}

@media (max-width: 991px) {

.chart-legend > li {

float: left;

margin-right: 10px;

}

}

.box-comments {

background: #f7f7f7;

}

.box-comments .box-comment {

padding: 8px 0;

border-bottom: 1px solid #eee;

}

.box-comments .box-comment:before,

.box-comments .box-comment:after {

content: " ";

display: table;

}

.box-comments .box-comment:after {

clear: both;

}

.box-comments .box-comment:last-of-type {

border-bottom: 0;

}

.box-comments .box-comment:first-of-type {

padding-top: 0;

}

.box-comments .box-comment img {

float: left;

}

.box-comments .comment-text {

margin-left: 40px;

color: #555;

}

.box-comments .username {

color: #444;

display: block;

font-weight: 600;

}

.box-comments .text-muted {

font-weight: 400;

font-size: 12px;

}

**-------------------------------------------------**

alterar html de view Login

<div class="container">

<div class="row">

<div class="col-lg-12 col-md-12">

<section class="content-header">

<h1> @ViewBag.Title <small></small> </h1>

<ol class="breadcrumb">

<li><a href="#"><i class="fa fa-dashboard"></i> Level</a></li>

<li class="active">Here</li>

</ol>

</section>

<section class="content">

<div class="box box-default">

<div class="box-header with-border">

<h3 class="box-title">zzzzzzzzz</h3>

</div>

<div class="box-body">

</div><!-- /.box-body -->

</div><!-- /.box box-default -->

</section><!-- /.content -->

</div><!-- /.col-lg-12 col-md-12 -->

</div><!-- /.row -->

</div><!-- /.container →

**continuar com**

**------------------------ controllerAula (Edit [HttpPost])**

**if (returnaction != "Index")**

**{**

**TempData["status"] = 0;**

**TempData["message"] = "Aula alterada com sucesso!";**

**}**

**------------------------ View Details**

**@{**

**ViewBag.Title = "Detalhes";**

**}**

**@if (TempData["status"] != null && TempData["status"] != "")**

**{**

**if ((int)TempData["status"] == 0)**

**{**

**<div class="box\_alert">**

**<div id="alert" class="alert alert-success alert-dismissible" role="alert">**

**<button type="button" class="close" data-dismiss="alert" aria-hidden="true">×</button>**

**<p><i class="icon fa fa-check"></i>@Html.Raw(@TempData["message"])</p>**

**</div>**

**</div>**

**}**

**else if ((int)TempData["status"] < 0)**

**{**

**<div class="box\_alert">**

**<div id="alert" class="alert alert-danger alert-dismissible" role="alert">**

**<button type="button" class="close" data-dismiss="alert" aria-hidden="true">×</button>**

**<p><i class="icon fa fa-check"></i>@Html.Raw(@TempData["message"])</p>**

**</div>**

**</div>**

**}**

**else if ((int)TempData["status"] > 0)**

**{**

**<div class="box\_alert">**

**<div id="alert" class="alert alert-warning alert-dismissible" role="alert">**

**<button type="button" class="close" data-dismiss="alert" aria-hidden="true">×</button>**

**<p><i class="icon fa fa-check"></i>@Html.Raw(@TempData["message"])</p>**

**</div>**

**</div>**

**}**

**}**

**---------------------------------------**

**--------------------------**

**Vamos alterar o layout da página de entrada do nosso portal**