**TESTE SIGCORP**

Nome: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Data: \_\_/\_\_/\_\_\_\_

**Tabelas**

|  |  |  |  |
| --- | --- | --- | --- |
| Pessoa | | | |
| Id | Guid | **PK** | **Not null** |
| Nome | Varchar(300) |  | **Not null** |
| Documento | Varchar(22) | **Unique** | **Not null** |
| Email | Varchar(300) | **Unique** | **Not null** |
| Telefone | Varchar(15) |  | **Null** |

|  |  |  |  |
| --- | --- | --- | --- |
| Conta | | | |
| Id | Guid | **PK** | **Not null** |
| Login | Varchar(300) | **Unique** | **Not null** |
| Senha | Varchar(MAX) |  | **Not null** |
| IsBloqueado | Bit |  | **Null** |
| AlterarSenha | bit |  | **Null** |
| PessoaId | GUID |  | **Not null** |

|  |  |  |  |
| --- | --- | --- | --- |
| Permissao | | | |
| Id | Guid | **PK** | **Not null** |
| Nome | Varchar(300) | **Unique** | **Not null** |

|  |  |  |  |
| --- | --- | --- | --- |
| PermissaoConta | | | |
| Id | Guid | **PK** | **Not null** |
| PermissaoId | Guid | **FK** | **Not null** |
| ContaId | Guid | **FK** | **Not null** |
| Read | Bit |  | **Not null** |
| Write | Bit |  | **Not null** |
| Execute | Bit |  | **Not null** |

|  |  |  |  |
| --- | --- | --- | --- |
| Venda | | | |
| Id | Guid | **PK** | **Not null** |
| Valor | Decimal(37,2) |  | **Not null** |

|  |  |  |  |
| --- | --- | --- | --- |
| Competencia | | | |
| Id | Guid | **PK** | **Not null** |
| Ano | Int |  | **Not null** |
| Mês | Int |  | **Not Null** |

|  |  |  |  |
| --- | --- | --- | --- |
| VendaProdutoCompetencia | | | |
| Id | Guid | **PK** | **Not null** |
| VendaId | Guid | **FK** | **Not null** |
| ProdutoId | Guid | **FK** | **Not null** |
| CompetenciaId | Guid | **FK** |  |
| DataVenda | DateTime2 |  | **Not null** |
| QuantidadeVendida | Int |  | **Not nulll** |

|  |  |  |  |
| --- | --- | --- | --- |
| Produto | | | |
| Id | Guid | **PK** | **Not null** |
| Nome | Varchar(300) |  | **Not null** |
| Descricao | Varchar(Max) |  | **Null** |
| Quantidade | Int |  | **Not null** |

|  |  |  |  |
| --- | --- | --- | --- |
| Boleto | | | |
| Id | Guid | **PK** | **Not null** |
| Numero | Bigint | **Unique** | **Not null** |
| DataVencimento | DateTime |  | **Not Null** |
| Valor | Decimal(35,2) |  | **Not null** |
| DataEmissao | DateTime |  | **Not Null** |
| Sequencial | Varchar(10) |  | **Not null** |
| CedenteId | Guid | **FK** | **Not null** |
| ClienteId (Pessoa) | Guid | **FK** | **Not null** |
| CompetenciaId | Guid | **FK** | **Not null** |

|  |  |  |  |
| --- | --- | --- | --- |
| Cedente | | | |
| Id | Guid | **PK** | **Not null** |
| Carteira | Varchar(30) |  | **Not null** |
| Contrato | Varchar(20) |  | **Not null** |
| NomeConta | Varchar(200) |  | **Not null** |
| Agencia | Varchar(20) |  | **Not Null** |
| DigitoVerificador | Char(1) |  | **Not null** |
| Instituicao | Varchar(200) |  | **Null** |
|  |  |  |  |

**Perguntas**

1. Escreva uma consulta em LINQ com base nas informações das tabelas acima. Buscar todas os produtos que possam ou não terem sido vendidos.

using (AppDbContext db = new AppDbContext())

{

var query = from t1 in db.Produto

join t2 in db.VendaProdutoCompetencias on t1.Id equals t2.ProdutoId into t3

from t2 in t3.DefaultIfEmpty()

elect new { Id = t1.Id, Nome = t1.Nome, DtVenda = t2.DataVenda };

foreach (var item in query)

{

Console.WriteLine("Pesq\_Venda()" + "" + item.Id + ";" + item.Nome + ";" + item.DtVenda + " Ok!");

}

}

select t1.id,t1.nome,t2.DataVenda

from Produto t1

left join VendaProdutoCompetencias t2 on t1.Id=t2.ProdutoId

1. Monte uma consulta em LINQ com base nas informações das tabelas acima: buscar uma conta onde o Login seja: “login@sigcorp.com.br” e Senha: “S#1$g55g@g542”

AppDbContext db = new AppDbContext();

var query = from t1 in db.Conta

join t2 in db.Pessoa on t1.PessoaId equals t2.Id

where t2.Email.Contains("login@sigcorp.com.br") & t1.Senha.Contains("S#1$g55g@g542")

select new {t1.Id, t1.Login,t1.Senha,t1.IsBloqueado,t1.AlterarSenha,t2.Nome,t2.Documento,t2.Email

,t2.Telefone};

foreach (var item in query)

{

Console.WriteLine("{0}\t{1}", item.Id, item.Login, item.Senha, item.IsBloqueado

, item.AlterarSenha, item.Nome, item.Documento, item.Email, item.Telefone);

}

select t1.\*,t2.\* from conta t1

inner join Pessoa t2 on t1.PessoaId = t2.Id

where t2.Email like 'login@sigcorp.com.br' and t1.Senha like 'S#1$g55g@g542'

1. Monte uma classe C# para cadastrar, atualizar, remover, buscar via identificador e buscar todos, genérica.

namespace EfCore.Controler

{

public class Pessoa\_Crud

{

public Guid Id\_ { get; set; }

public string Nome\_ { get; set; }

public string Documento\_ { get; set; }

public string Email\_ { get; set; }

public string Telefone\_ { get; set; }

public void Buscar()

{

try

{

using (AppDbContext db = new AppDbContext())

{

var query = from t1 in db.Pessoa

where t1.Id.Equals(Id\_)

select new { t1.Id, t1.Nome, t1.Documento, t1.Email, t1.Telefone };

foreach (var item in query)

{

Console.WriteLine("Buscar():" + item.Id + ";" + item.Nome + ";" + item.Documento + ";" + item.Email + ";" + item.Telefone + ";" + " OK!");

}

}

}

catch (Exception e)

{

if (e.Source != null)

Console.WriteLine("Err: {0}", e.Source);

}

}

public bool Buscar\_Aux()

{

try

{

using (AppDbContext db = new AppDbContext())

{

var Pessoas = db.Set<Model.Pessoa>();

var registro = db.Pessoa.Find(Id\_);

if (registro.Id != null)

{

return true;

}

else return false;

}

}

catch (Exception e)

{

if (e.Source != null)

Console.WriteLine("Err: {0}", e.Source);

return false;

}

}

public void Cadastrar()

{

try

{

using (AppDbContext db = new AppDbContext())

{

var Pessoas = db.Set<Model.Pessoa>();

Pessoas.Add(new Model.Pessoa { Id = Id\_, Nome = Nome\_, Documento = Documento\_, Email = Email\_, Telefone = Telefone\_ });

db.SaveChanges();

Console.WriteLine("Cadastrar(): {0}", "Ok!");

}

}

catch (Exception e)

{

if (e.Source != null)

Console.WriteLine("Err: {0}", e.Source);

throw;

}

}

public void Atualizar()

{

try

{

using (AppDbContext db = new AppDbContext())

{

var Pessoas = db.Set<Model.Pessoa>();

Pessoas.Update(new Model.Pessoa { Id = Id\_, Nome = Nome\_, Documento = Documento\_, Email = Email\_, Telefone = Telefone\_ });

db.SaveChanges();

Console.WriteLine("Atualizar(): {0}", "Ok!");

}

}

catch (Exception e)

{

if (e.Source != null)

Console.WriteLine("Err: {0}", e.Source);

}

}

public void Remover()

{

try

{

using (AppDbContext db = new AppDbContext())

{

var Pessoas = db.Set<Model.Pessoa>();

Pessoas.Remove(new Model.Pessoa { Id = Id\_});

db.SaveChanges();

Console.WriteLine("Deletar(): {0}", "Ok!");

}

}

catch (Exception e)

{

if (e.Source != null)

Console.WriteLine("Err: {0}", e.Source);

}

}

}

}

1. Monte uma classe C# para buscar todas as pessoas que não possuam uma conta.

namespace EfCore.Controler

{

public class Pesquisa

{

public void Pesq\_Conta\_sem\_Filtro()

{

using (AppDbContext db = new AppDbContext())

{

var query = from t1 in db.Pessoa

join t2 in db.Conta on t1.Id equals t2.PessoaId

select new { Id = t1.Id, Nome = t1.Nome, Login = t2.Login };

foreach (var item in query)

{

Console.WriteLine("Pesq\_Conta\_sem\_Filtro(): " + ";" + item.Id + ";" + item.Nome + ";" + item.Login + "Ok!");

}

}

}

}

}

1. Monte uma classe C# (POCO) para cada Tabela acima. (Orm EF Core )

namespace EfCore.Model

{

public class Boleto

{

[Key]

[Required]

public Guid Id { get; set; }

[Required]

//[UniqueKey(groupId: "1", order: 0)]

[Column(TypeName = "bigint")]

public int Numero { get; set; }

[Required]

public DateTime DataVencimento { get; set; }

[Required]

[Column(TypeName = "decimal(35, 2)")]

public decimal Valor { get; set; }

[Required]

public DateTime DataEmissao { get; set; }

[Required]

[Column(TypeName = "varchar(10)")]

public int Sequencial { get; set; }

/\*

\*Relacionamentos

\*/

[Required]

public Guid CedenteId { get; set; }

public Cedente Cedente { get; set; }

[Required]

public Guid ClientId { get; set; }

public Pessoa Pessoa { get; set; }

[Required]

public Guid CompetenciaId { get; set; }

public Competencia Competencia { get; set; }

}

}

namespace EfCore.Model

{

public class Cedente

{

[Key,Required]

public Guid Id { get; set; }

[Required,Column(TypeName = "varchar(30)")]

public string Carteira { get; set; }

[Required,Column(TypeName = "varchar(20)")]

public string Contrato { get; set; }

[Required,Column(TypeName = "varchar(200)")]

public string NomeConta { get; set; }

[Required,Column(TypeName = "varchar(20)")]

public string Agencia { get; set; }

[Required,Column(TypeName = "char(1)")]

public char DigitoVerificador { get; set; }

[Column(TypeName = "varchar(200)")]

public string Instituicao { get; set; }

/\*

\* Relacionamentos

\*/

public ICollection<Boleto> Boleto { get; set; }

}

}

namespace EfCore.Model

{

public class Competencia

{

[Key]

[Required]

public Guid CompetenciaId { get; set; }

[Required]

public int Ano { get; set; }

[Required]

public int Mes { get; set; }

/\*

\* Relacionamentos

\*/

public ICollection<VendaProdutoCompetencia> VendaProdutoCompetencias { get; set; }

public ICollection<Boleto> Boleto { get; set; }

}

}

namespace EfCore.Model

{

public class Conta

{

[Key,Required]

public Guid Id { get; set; }

[Required,DatabaseGeneratedAttribute(DatabaseGeneratedOption.Identity),Column(TypeName = "varchar(300)")]

public string Login { get; set; }

[Required,Column(TypeName = "varchar(Max)")]

public string Senha { get; set; }

public Boolean IsBloqueado { get; set; }

public Boolean AlterarSenha { get; set; }

[Required]

public Guid PessoaId { get; set; }

public Pessoa Pessoas { get; set; }

public ICollection<PermissaoConta> PermissaoConta { get; set; }

}

}

namespace EfCore.Model

{

public class Permissao

{

[Key]

[Required]

public Guid Id { get; set; }

[Required, DatabaseGeneratedAttribute(DatabaseGeneratedOption.Identity), Column(TypeName = "varchar(300)")]

public string Nome { get; set; }

public ICollection<PermissaoConta> PermissaoConta { get; set; }

}

}

namespace EfCore.Model

{

public class Pessoa

{

[Key]

[Required]

public Guid Id { get; set; }

[Required]

[Column(TypeName = "varchar(300)")]

public string Nome { get; set; }

[Required,DatabaseGeneratedAttribute(DatabaseGeneratedOption.Identity), Column(TypeName = "varchar(22)")]

public string Documento { get; set; }

[Required,DatabaseGeneratedAttribute(DatabaseGeneratedOption.Identity), Column(TypeName = "varchar(300)")]

public string Email { get; set; }

[Column(TypeName = "varchar(15)")]

public string Telefone { get; set; }

/\*

\* Relacionamento

\*/

public ICollection<Conta> Contas { get; set; }

public ICollection<Boleto> Boleto { get; set; }

}

}

namespace EfCore.Model

{

public class Produto

{

[Key]

public Guid Id { get; set; }

[Required]

[Column(TypeName = "varchar(300)")]

public string Nome { get; set; }

[Column(TypeName = "varchar(Max)")]

public string Descricao { get; set; }

[Required]

public int Quantidade { get; set; }

public ICollection<VendaProdutoCompetencia> VendaProdutoCompetencias { get; set; }

}

}

namespace EfCore.Model

{

public class Venda

{

[Key]

[Required]

public Guid Id { get; set; }

[Required]

[Column(TypeName = "decimal(37, 2)")]

public decimal Valor { get; set; }

public ICollection<VendaProdutoCompetencia> VendaProdutoCompetencias { get; set; }

}

}

namespace EfCore.Model

{

public class VendaProdutoCompetencia

{

[Key]

[Required]

public Guid Id { get; set; }

[Required]

public DateTime DataVenda { get; set; }

[Required]

public int QuantidadeVendida { get; set; }

/\*

\* Relacionamento 1 para muitos

\* \*/

[Required]

public Guid VendaId { get; set; }

public Venda Vendas { get; set; }

[Required]

public Guid ProdutoId { get; set; }

public Produto Produtos { get; set; }

[Required]

public Guid CompetenciaId { get; set; }

public Competencia Competencias { get; set; }

}

}

namespace EfCore.Model

{

public class PermissaoConta

{

[Key]

[Required]

public Guid Id { get; set; }

[Required]

public Guid ContaId { get; set; }

[Required]

public Boolean Read { get; set; }

[Required]

public Boolean Write { get; set; }

[Required]

public Boolean Execute { get; set; }

/\*

\* relacionamento

\*/

[Required]

public Guid PermissaoId { get; set; }

public Permissao Permissao { get; set; }

}

}

1. Monte uma classe C# para desserializar um arquivo txt:
   1. De 1 a 20 Nome
   2. 21 a 35 CPF/CNPJ
   3. 36 a 56 Valor (com duas casas decimais)
   4. 57 a 58 Indicador ativo ou inativo
   5. 59 a 60 Indicador de Pessoa Física ou Jurídica
   6. 61 a 100 Campo Livre
2. Criar uma tela para exibir um relatório que exiba todos os produtos, o relatório deve exportar para excel e pdf, sendo desenvolvido utilizando ReportView ou em uma tela HTML utilizando JQuery, mas caso o sistema seja feito em Angular o a exibição do relatório fica à disposição do candidato. O relatório deve montar um gráfico com os dados obtidos além de conter, pelo menos, os filtros:

- Produto mais vendido

- Produto menos vendido

- Quantidade de produto

- Nome