

**PROJETO DE BANCO DE DADOS**

**BIBLIOTECA COMPARTILHADA**

**ANA CLARA DA CONCEIÇÃO MACÊDO DA SILVA**

**GUSTAVO SILVA ALENCAR SIQUEIRA**

**JOÃO ANTONIO DOS SANTOS ILÁRIO**

**RODRIGO VASCONCELOS FIGUEIREDO**

**PALMAS - TO**

**10 de fevereiro de 2017**

**BIBLIOTECA COMPARTILHADA**

Projeto apresentado como requisito parcial para obtenção de aprovação na disciplina de Banco de Dados II, dos cursos de Sistemas da Informação e Ciências da Computação, no Centro Universitário Luterano de Palmas.

Prof. Me. Fernando Luiz de Oliveira

**PALMAS - TO**

**10 de fevereiro de 2017**

**SUMÁRIO**

**1.** [**APRESENTAÇÃO**](#_30j0zll) **4**

**2.** [**LEVANTAMENTO DE REQUISITOS**](#_1fob9te) **5**

**3.** [**PROJETO CONCEITUAL**](#_3znysh7) **6**

**4.** [**PROJETO LÓGICO**](#_2et92p0) **7**

**5.** [**PROJETO FÍSICO**](#_tyjcwt) **8**

[**6. DICIONÁRIO DE DADOS**](#_3dy6vkm) **10**

**7.** [**SQL EMBUTIDA - STORED PROCEDURE**](#_1t3h5sf) **13**

[**8. SQL EMBUTIDA - VIEWS**](#_2s8eyo1) **17**

[**9. SQL EMBUTIDA - TRIGGER**](#_17dp8vu) **18**

**10**[**. SQL EMBUTIDA -**](#_17dp8vu) **FUNÇÃO 19**

# APRESENTAÇÃO

O sistema de Biblioteca Compartilhada pretende atender aos seus leitores, otimizando processos e conferindo confiabilidade às informações e controles. Suas principais funcionalidades são: pesquisa de itens do acervo pelos próprios leitores da Biblioteca Compartilhada, podendo informar parâmetros de pesquisa; Automação do processo de empréstimo e devolução dos itens do acervo; Registro automatizado de empréstimos e Avaliação de itens realizada pelos próprios usuários da biblioteca compartilhada.

Espera-se que o sistema venha trazer diversos benefícios aos seus usuários, como: Incentivo à leitura, maiores cuidados na manutenção dos itens de outro usuário (do acervo da biblioteca compartilhada); Ampliação de itens, permitindo a inclusão de livros.

O sistema abrange o aplicativo para a manutenção dos dados (livros e usuários leitores), funções de empréstimo de livros, devolução e reserva bem como, a disponibilização de consultas ao acervo de livros para o usuário. Além disso, funções básicas para avaliação de um empréstimo.

# LEVANTAMENTO DE REQUISITOS

R1: Gerenciar Leitor:

UC 1.1: Cadastrar Leitor

UC 1.2: Alterar Dados do Leitor

UC 1.3: Pesquisar Leitor

UC 1.5: Desabilitar Leitor

UC 1.6: Excluir Leitor

R2: Gerenciar Livro:

UC 2.1: Cadastrar Livro

UC 2.2: Alterar Dados do Livro

UC 2.3: Recomendar Livro

UC 2.4: Pesquisar Livro

UC 2.5: Denunciar Livro

UC 2.6: Desabilitar Livro

UC 2.7: Excluir Livro

R3: Gerenciar Empréstimo:

UC 3.1: Solicitar Empréstimo

UC 3.2: Disponibilizar Empréstimo

UC 3.2: Alterar Empréstimo

UC 3.3: Realizar Empréstimo (Busca ou Envio)

UC 3.3: Cancelar Empréstimo

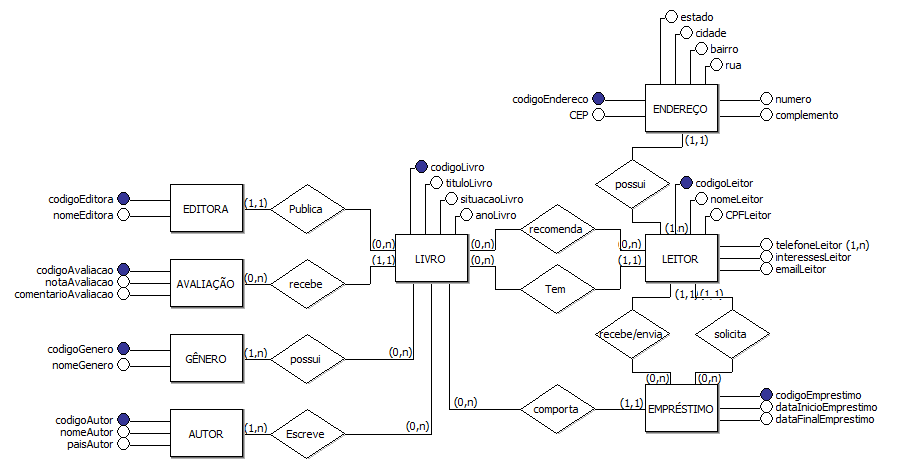
R4: Gerenciar Avaliação:

UC 4.1: Cadastrar Avaliação

UC 4.2: Alterar Avaliação

UC 4.3: Excluir Avaliação

# PROJETO CONCEITUAL



# PROJETO LÓGICO

Editora (**codigoEditora**, nomeEditora)

Genero (**codigoGenero**, nomeGenero)

Autor (**codigoAutor**, nomeAutor, paisAutor)

Endereco (**codigoEndereco**, CEP, estado, cidade, bairro, rua, numero, complemento)

Telefone (**codigoTelefone**, TelCelular, TelResidencial, TelComercial)

Leitor (**codigoLeitor**, nomeLeitor, CPFLeitor, emailLeitor, interesseLeitor, **codEndereco,** **codTelefone**)

Leitor.codEndereco = Endereco.codigoEndereco

Leitor.codTelefone = Telefone.codigoTelefone

Livro (**codigoLivro,** tituloLivro, situacaoLivro, anoLivro, **codEditora, codLeitor, codEmprestimo**)

Livro.codEditora = Editora.codigoEditora

Livro.codLeitor = Leitor.codigoLeitor

Livro.codEmprestimo = Emprestimo.codigoEmprestimo

Avaliacao (**codigoAvaliacao**, notaAvaliacao, comentarioAvaliacao, **codLivro**)

Avalicao.codLivro = Livro.codigoLivro

LivroPossuiGenero (**codGenero, codLivro**)

LivroPossuiGenero.codGenero = Genero.codigoGenero

LivroPossuiGenero.codLivro = Livro.codigoLivro

AutorEscreveLivro (**codAutor, codLivro**)

AutorEscreveLivro.codAutor = Autor.codigoAutor

AutorEscreveLivro.codLivro = Livro.codigoLivro

LeitorRecomendaLivro (**codLeitor, codLivro**)

LeitorRecomendaLivro.codLeitor= Leitor.codigoLeitor

LeitorRecomendaLivro.codLivro = Livro.codigoLivro

Emprestimo (**codigoEmprestimo**, dataInicioEmprestimo, dataFinalEmprestimo, **codLeitorRecebe, codLeitorSolicita**)

Emprestimo.codLeitorRecebe = Leitor.codigoLeitor

Emprestivo.codLeitorSolicita = Leitor.codigoLeitor

# PROJETO FÍSICO

CREATE DATABASE PROJETO\_BDII;

USE PROJETO\_BDII;

CREATE TABLE tb\_Editora(

codigoEditora INT IDENTITY(0,1),

nomeEditora VARCHAR(100) NOT NULL,

PRIMARY KEY(codigoEditora)

);

CREATE TABLE tb\_Genero(

codigoGenero INT IDENTITY(0,1),

nomeGenero VARCHAR(100) NOT NULL,

PRIMARY KEY(codigoGenero)

);

CREATE TABLE tb\_Autor(

codigoAutor INT IDENTITY(0,1),

nomeAutor VARCHAR(100) NOT NULL,

paisAutor VARCHAR(100) NOT NULL,

PRIMARY KEY(codigoAutor)

);

CREATE TABLE tb\_Endereco(

codigoEndereco INT IDENTITY(0,1),

CEP INT NOT NULL,

estado VARCHAR(100) NOT NULL,

cidade VARCHAR(100) NOT NULL,

bairro VARCHAR(100) NOT NULL,

rua VARCHAR(100) NOT NULL,

numero INT NOT NULL,

complemento VARCHAR(150) NOT NULL

PRIMARY KEY(codigoEndereco),

);

CREATE TABLE tb\_Telefone(

codigoTelefone INT IDENTITY(0,1),

TelCelular INT NOT NULL,

TelResidencial INT,

TelComercial INT

PRIMARY KEY(codigoTelefone),

);

CREATE TABLE tb\_Leitor(

codigoLeitor INT IDENTITY(0,1),

nomeLeitor VARCHAR(100) NOT NULL,

CPFLeitor INT NOT NULL,

emailLeitor VARCHAR(50) NOT NULL,

interesseLeitor VARCHAR(300) NOT NULL,

codInteresse INT NOT NULL,

codEndereco INT NOT NULL,

codTelefone INT NOT NULL,

PRIMARY KEY(codigoLeitor),

FOREIGN KEY(codEndereco) REFERENCES tb\_Endereco(codigoEndereco),

FOREIGN KEY(codTelefone) REFERENCES tb\_Telefone(codigoTelefone)

);

CREATE TABLE tb\_Livro(

codigoLivro INT IDENTITY(0,1),

tituloLivro VARCHAR(100) NOT NULL,

situacaoLivro VARCHAR(50) NOT NULL,

anoLivro INT NOT NULL,

codEditora INT NOT NULL,

codLeitor INT NOT NULL,

PRIMARY KEY(codigoLivro),

FOREIGN KEY(codEditora) REFERENCES tb\_Editora(codigoEditora),

FOREIGN KEY(codLeitor) REFERENCES tb\_Leitor(codigoLeitor),

FOREIGN KEY(codEmprestimo) REFERENCES tb\_Emprestimo(codigoEmprestimo)

);  
CREATE TABLE tb\_Avaliacao(

codigoAvaliacao INT IDENTITY(0,1),

notaAvaliacao INT NOT NULL,

comentarioAvaliacao VARCHAR(250),

codLivro INT NOT NULL,

PRIMARY KEY(codigoAvaliacao),

FOREIGN KEY(codLivro) REFERENCES tb\_Livro(codigoLivro)

);  
CREATE TABLE tb\_LivroPossuiGenero(

codGenero INT NOT NULL,

codLivro INT NOT NULL,

FOREIGN KEY(codGenero) REFERENCES tb\_Genero(codigoGenero),

FOREIGN KEY(codLivro) REFERENCES tb\_Livro(codigoLivro)

);

CREATE TABLE tb\_AutorEscreveLivro(

codAutor INT NOT NULL,

codLivro INT NOT NULL,

FOREIGN KEY(codAutor) REFERENCES tb\_Autor(codigoAutor),

FOREIGN KEY(codLivro) REFERENCES tb\_Livro(codigoLivro)

);

CREATE TABLE tb\_LeitorRecomendaLivro(

codLeitor INT NOT NULL,

codLivro INT NOT NULL,

FOREIGN KEY(codLeitor) REFERENCES Leitor(codigoLeitor),

FOREIGN KEY(codLivro) REFERENCES Livro(codigoLivro)

);

CREATE TABLE tb\_Emprestimo(

codigoEmprestimo INT IDENTITY(0,1),

dataInicioEmprestimo DATE NOT NULL,

dataFinalEmprestimo DATE NOT NULL,

codLeitorRecebe INT NOT NULL,

codLeitorSolicita INT NOT NULL,

PRIMARY KEY(codigoEmprestimo),

FOREIGN KEY(codLeitorRecebe) REFERENCES tb\_Leitor(codigoLeitor),

FOREIGN KEY(codLeitorSolicita) REFERENCES tb\_Leitor(codigoLeitor)

);

CREATE TABLE tb\_EmprestimoComportaLivro(

codEmprestimo INT NOT NULL,

codLivro INT NOT NULL,

FOREIGN KEY(codEmprestimo) REFERENCES Emprestimo(codigoEmprestimo),

FOREIGN KEY(codLivro) REFERENCES Livro(codigoLivro)

);

# 6. DICIONÁRIO DE DADOS

**Editora**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Campo** | **Tipo** | **Tamanho** | **Propriedades** | **Descrição** |
| codigoEditora | Int | - | Auto-increment | Primary Key | ID do registro de Editora |
| nomeEditora | Varchar | 100 | Not null | Nome da Editora |

**Gênero**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Campo** | **Tipo** | **Tamanho** | **Propriedades** | **Descrição** |
| codigoGenero | Int | - | Auto-increment | Primary Key | ID do registro de Genero |
| nomeGenero | Varchar | 100 | Not null | Nome do Genero |
| codigoGenero | Int | - | Auto-increment | Primary Key | ID do registro de Genero |

**Autor**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Campo** | **Tipo** | **Tamanho** | **Propriedades** | **Descrição** |
| codigoAutor | Int | - | Auto-increment | Primary Key | ID do registro de Autor |
| nomeAutor | Varchar | 100 | Not null | Nome do Autor |
| paisAutor | Varchar | 100 | Not null | País do Autor |

**Endereço**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Campo** | **Tipo** | **Tamanho** | **Propriedades** | **Descrição** |
| codigoEndereco | Int | - | Auto-increment | Primary Key | ID do registro de Endereço |
| CEP | Int | - | Not null | CEP |
| estado | Varchar | 100 | Not null | Nome do Estado |
| cidade | Varchar | 100 | Not null | Nome da Cidade |
| bairro | Varchar | 100 | Not null | Nome do Bairro |
| rua | Varchar | 100 | Not null | Nome da Rua |
| numero | Int | - | Not null | Número da Casa/Apartamento |
| complemento | Varchar | 150 | Not null | Complemento do Endereço |

**Telefone**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Campo** | **Tipo** | **Tamanho** | **Propriedades** | **Descrição** |
| codigoTelefone | Int | - | Auto-increment | Primary Key | ID do registro de Telefone |
| TelCelular | Int | - | Not null | Número de telefone Celular |
| TelResidencial | Int | - | - | Número de telefone Residencial |
| TelComercial | Int | - | - | Número de telefone Comercial |

**Leitor**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Campo** | **Tipo** | **Tamanho** | **Propriedades** | **Descrição** |
| codigoLeitor | Int | - | Auto-increment | Primary Key | ID do registro de Leitor |
| nomeLeitor | Varchar | 100 | Not null | Nome do Leitor |
| CPFLeitor | Int | - | Not null | Número do CPF do Leitor |
| emailLeitor | Varchar | 50 | Not null | E-mail do Leitor |
| interesseLeitor | Varchar | 300 | Not null | Interesse do Leitor |
| codEndereco | Int | - | Not null | Foreign Key | Código do Endereço |
| codTelefone | Int | - | Not null | Foreign Key | Código do Telefone |

**Livro**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Campo** | **Tipo** | **Tamanho** | **Propriedades** | **Descrição** |
| codigoLivro | Int | - | Auto-increment | Primary Key | ID do registro de Livro |
| tituloLivro | Varchar | 100 | Not null | Título do Livro |
| situacaoLivro | Varchar | 50 | Not null | Situação do Livro |
| anoLivro | Int | - | Not null | Ano do Livro |
| codEditora | Int | - | Not null | Foreign Key | Código da Editora |
| codLeitor | Int | - | Not null | Foreign Key | Código do Leitor |
| codEmprestimo | Int | - | Not null | Foreign Key | Código do Empréstimo |

**Avaliação**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Campo** | **Tipo** | **Tamanho** | **Propriedades** | **Descrição** |
| codigoAvaliacao | Int | - | Auto-increment | Primary Key | ID do registro de avaliação |
| notaAvaliacao | Int | - | Not null | Nota da Avalição |
| comentarioAvaliacao | Varchar | 250 | Not null | Comentário da Avaliação |
| codLivro | Int | - | Not null | Foreign Key | Código do Livro |

**Livro Possui Gênero**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Campo** | **Tipo** | **Tamanho** | **Propriedades** | **Descrição** |
| codGenero | Int | - | Not null | Foreign Key | Código do Gênero |
| codLivro | Int | - | Not null | Foreign Key | Código do Livro |

**Autor Escreve Livro**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Campo** | **Tipo** | **Tamanho** | **Propriedades** | **Descrição** |
| codAutor | Int | - | Not null | Foreign Key | Código do Autor |
| codLivro | Int | - | Not null | Foreign Key | Código do Livro |

**Leitor Recomenda Livro**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Campo** | **Tipo** | **Tamanho** | **Propriedades** | **Descrição** |
| codLeitor | Int | - | Not null | Foreign Key | Código do Leitor |
| codLivro | Int | - | Not null | Foreign Key | Código do Livro |

**Empréstimo**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Campo** | **Tipo** | **Tamanho** | **Propriedades** | **Descrição** |
| codigoEmprestimo | Int | - | Auto-increment | Primary Key | ID do registro de Empréstimo |
| dataInicioEmprestimo | Date | - | Not null | Data do início do empréstimo |
| dataFinalEmprestimo | Date | - | Not null | Data do fim do empréstimo |
| codLeitorRecebe | Int | - | Not null | Foreign Key | Código do Leitor |
| codLeitorSolicita | Int | - | Not null | Foreign Key | Código do Leitor |

**7. SQL EMBUTIDA - STORED PROCEDURE**

**Gerenciar Leitor - Cadastrar, atualizar e desativar**

ALTER PROCEDURE pr\_gerenciaLeitor

@codLeitor INT=NULL, @nomeLeitor VARCHAR(100)=NULL, @CPF INT=NULL,   
@emailLeitor VARCHAR(50)=NULL, @interesseLeitor VARCHAR(100)=NULL,   
@situacaoLeitor INT=NULL, @codEndereco INT=NULL, @codTelefone INT=NULL,   
@telCelular INT=NULL, @telResidencial INT=NULL, @telComercial INT=NULL,   
@CEP INT=NULL, @estado VARCHAR(100)=NULL, @cidade VARCHAR(100)=NULL,   
@bairro VARCHAR(100)=NULL, @rua VARCHAR(100)=NULL, @numero INT=NULL,   
@complemento VARCHAR(150)=NULL, @opcao INT=NULL, @msg VARCHAR(50) OUTPUT

AS

BEGIN

DECLARE

@codTel INT,

@codEnd INT

IF (@opcao = 1) BEGIN

INSERT INTO dbo.tb\_endereco VALUES (@CEP, @estado, @cidade, @bairro, @rua, @numero, @complemento)

SET @codEnd=(SELECT SCOPE\_IDENTITY())

INSERT INTO dbo.tb\_telefone VALUES (@telCelular ,@telResidencial, @telComercial)

SET @codTel=(SELECT SCOPE\_IDENTITY())

INSERT INTO tb\_leitor VALUES (@nomeLeitor, @CPF, @emailLeitor, @interesseLeitor,@situacaoLeitor, @codEnd, @codTel)

SET @msg='Leitor cadastrado com sucesso'

END

ELSE BEGIN

IF (@opcao = 2) BEGIN

UPDATE tb\_leitor SET nomeLeitor=@nomeLeitor, CPFLeitor=@CPF, emailLeitor=@emailLeitor, interesseLeitor=@interesseLeitor, situacaoLeitor=@situacaoLeitor, codEndereco=@codEndereco, codTelefone=@codTelefone

WHERE codLeitor=@codLeitor

UPDATE tb\_endereco SET CEP=@CEP, estado=@estado, cidade=@cidade, bairro=@bairro, rua=@rua, numero=@numero, complemento=@complemento

WHERE codEndereco=@codEndereco

UPDATE tb\_telefone SET TelCelular=@telCelular, TelResidencial=@telResidencial, TelComercial=@telComercial WHERE codTelefone=@codTelefone

SET @msg='Leitor atualizado com sucesso'

END

END

END

**Gerenciar Leitor - Pesquisar**

CREATE PROCEDURE pr\_pesquisaLeitor

@nomeLeitor VARCHAR(100)=NULL

AS

BEGIN

SELECT nomeLeitor FROM tb\_leitor WHERE CONTAINS(nomeLeitor, @nomeLeitor)

END

**Gerenciar Livro - Cadastrar, alterar e desativar**

CREATE PROCEDURE pr\_gerenciaLivro

@codLivro INT=NULL,

@tituloLivro VARCHAR(100)=NULL,

@situacaoLivro VARCHAR(50)=NULL,

@anoLivro INT=NULL,

@codEditora INT=NULL,

@codLeitor INT=NULL,

@opcao INT=NULL,

@msg VARCHAR(50) OUTPUT

AS

BEGIN

IF (@opcao = 1) BEGIN

INSERT INTO tb\_livro VALUES (@tituloLivro, @situacaoLivro, @anoLivro, @codEditora, @codLeitor)

SET @msg='Livro cadastrado com sucesso'

END

ELSE BEGIN

IF (@opcao = 2) BEGIN

UPDATE tb\_livro

SET tituloLivro=@tituloLivro, situacaoLivro=@situacaoLivro, anoLivro=@anoLivro, codEditora=@codEditora, codLeitor=@codLeitor WHERE codLivro=@codLivro

SET @msg='Livro atualizado com sucesso'

END

END

END

**Gerenciar Livro - Adicionar gênero**

CREATE PROCEDURE pr\_addGernero

@codGenero INT=NULL,

@codLivro INT=NULL,

@opcao INT=NULL,

AS

BEGIN

IF (@opcao = 1) BEGIN

INSERT INTO tb\_livropossuigenero VALUES (@codGenero, @codLivro)

END

ELSE BEGIN

IF (@opcao = 3) BEGIN

DELETE FROM tb\_livropossuigenero WHERE codGenero=@codGenero AND codLivro=@codLivro

END

END

**Gerenciar Livro - Recomendar Livro**

CREATE PROCEDURE pr\_addRecomendacao

@codLeitor INT=NULL,

@codLivro INT=NULL,

@opcao INT=NULL,

AS

BEGIN

IF (@opcao = 1) BEGIN

INSERT INTO tb\_leitorrecomendalivro VALUES (@codLeitor, @codLivro)

END

ELSE BEGIN

IF (@opcao = 3) BEGIN

DELETE FROM tb\_leitorrecomendalivro WHERE codLeitor=@codLeitor AND codLivro=@codLivro

END

END

**Gerenciar Livro - Pesquisar livro**

# 

CREATE PROCEDURE pr\_pesquisaLivro

@nomeLivro VARCHAR(100)=NULL

AS

BEGIN

SELECT l.tituloLivro, l.codEditora, l.anoLivro, l.situacaoLivro FROM tb\_livro l

WHERE CONTAINS(l.tituloLivro, @nomeLivro)

END

**Gerenciar Empréstimo** - **Cadastrar e atualizar**

CREATE PROCEDURE pr\_gerenciaEmprestimo

@codEmprestimo INT=NULL,

@dataInicioEmprestimo DATE=NULL,

@dataFinalEmprestimo DATE=NULL,

@codLeitorSolicita INT=NULL,

@opçao INT=NULL,

AS

BEGIN

IF (@opcao = 1) BEGIN

INSERT INTO tb\_emprestimo VALUES (@dataInicioEmprestimo, @dataFinalEmprestimo, @codLeitorSolicita)

END

ELSE BEGIN

IF (@opcao = 2) BEGIN

UPDATE tb\_emprestimo SET dataInicioEmprestimo=@dataInicioEmprestimo dataFinalEmprestimo=@dataFinalEmprestimo WHERE codEmprestimo=@codEmprestimo

END

END

END

**Gerenciar Empréstimo** - **Adicionar e remover livro ao empréstimo**

CREATE PROCEDURE pr\_adicionaLivroEmprestimo  
@codEmprestimo INT=NULL,  
@codLivro INT=NULL,  
@msg VARCHAR(50) OUTPUT  
AS  
BEGIN  
DECLARE @status INT

SET @status=(SELECT situacaoLivro FROM tb\_livro WHERE codLivro=@codLivro)

IF(@status=1) BEGIN

INSERT INTO tb\_emprestimocomportalivro VALUES (@codEmprestimo, @codLivro)

UPDATE tb\_livro SET situacaoLivro=’Ocupado’ WHERE codLivro=@codLivro

SET @msg='Livro inserido no empréstimo'

END

ELSE BEGIN

DELETE FROM tb\_emprestimocomportalivro WHERE codLivro=@codLivro

UPDATE dbo.tb\_livro SET situacaoLivro=’Disponível’ WHERE codLivro=@codLivro

SET @msg='Livro removido do empréstimo'

END

END

**Gerenciar Avaliação - Cadastrar, alterar e excluir**

CREATE PROCEDURE pr\_gerenciaAvaliacao

@codAvaliacao INT=NULL,

@notaAvaliacao INT=NULL,

@comentarioAvaliacao VARCHAR(250)=NULL,

@codLivro INT=NULL,

@opcao INT=NULL,

@msg VARCHAR(50) OUTPUT

AS

BEGIN

IF (@opcao = 1) BEGIN

INSERT INTO tb\_avaliacao VALUES (@notaAvaliacao, @comentarioAvaliacao, @codLivro)

SET @msg='Avaliação inserida'

END

ELSE BEGIN

IF (@opcao = 2) BEGIN

UPDATE tb\_avaliacao

SET notaAvaliacao=@notaAvaliacao, comentarioAvaliacao=@comentarioAvaliacao, codLivro=@codLivro WHERE codAvaliacao=@codAvaliacao  
 SET @msg='Avaliação Atualizada'

END

ELSE BEGIN

IF(@opcao = 3) BEGIN

DELETE FROM tb\_avaliacao WHERE codAvaliacao=@codAvaliacao

SET @msg='Avaliação removida'

END

END

END

# 8. SQL EMBUTIDA - VIEWS

**Livro - View livros disponíveis**

CREATE VIEW vw\_livro\_disponivel

(Livro, Editora, Ano)

AS

SELECT tituloLivro, nomeEditora, anoLivro FROM tb\_livro l INNER JOIN tb\_editora e ON l.codEditora=e.codEditora

WHERE l.situacaoLivro='Disponível'

**Livro - View livros mais lidos**

CREATE VIEW vw\_livros\_mais\_lidos

AS

SELECT COUNT(\*) AS Total, l.tituloLivro FROM dbo.tb\_livro l INNER JOIN dbo.tb\_emprestimocomportalivro e ON l.codLivro=e.codLivro

GROUP BY e.codLivro, l.tituloLivro

ORDER BY COUNT(\*) DESC

**Avaliação - Nota média de avaliação por livro**

CREATE VIEW vw\_livros\_media\_avaliacao

AS

SELECT AVG(a.notaAvaliacao) AS Media, l.tituloLivro FROM dbo.tb\_livro l INNER JOIN dbo.tb\_avaliacao a ON l.codLivro=a.codLivro

GROUP BY l.tituloLivro

# 9. SQL EMBUTIDA - TRIGGER

**Livro- Trigger Atualiza Situação do Livro**

CREATE TRIGGER tg\_AtualizaSituacaoLivro

ON tb\_emprestimo

FOR INSERT

AS

DECLARE

@codEmprestimo int

IF (select count(\*) from INSERTED)>0

BEGIN

SET @codEmprestimo = (SELECT codEmprestimo FROM INSERTED);

IF (SELECT COUNT(\*) FROM Livro WHERE codEmprestimo=@codEmprestimo ) > 0

UPDATE tb\_livro SET situacaoLivro='Ocupado';

ELSE BEGIN

UPDATE tb\_livro SET situacaoLivro='Disponível';

END

END

# 10. SQL EMBUTIDA - FUNÇÃO

**Função filtro por editora**

CREATE FUNCTION fc\_livros\_da\_editora(@editora VARCHAR(100))

RETURNS TABLE

AS

RETURN(

SELECT l.tituloLivro, a.nomeAutor, e.nomeEditora

FROM tb\_livro l

INNER JOIN tb\_editora e ON l.codEditora=e.codEditora

INNER JOIN dbo.tb\_autor a ON l.codAutor = a.codAutor

WHERE e.nomeEditora =@editora

)

**Função verifica situação do livro**

CREATE FUNCTION fc\_status\_livro(@codLivro INT)

RETURNS VARCHAR(50)

AS

BEGIN

DECLARE @status VARCHAR(50)

SELECT @status=(situacaoLivro)

FROM tb\_livro

WHERE codLivro=@codLivro

RETURN @status

END