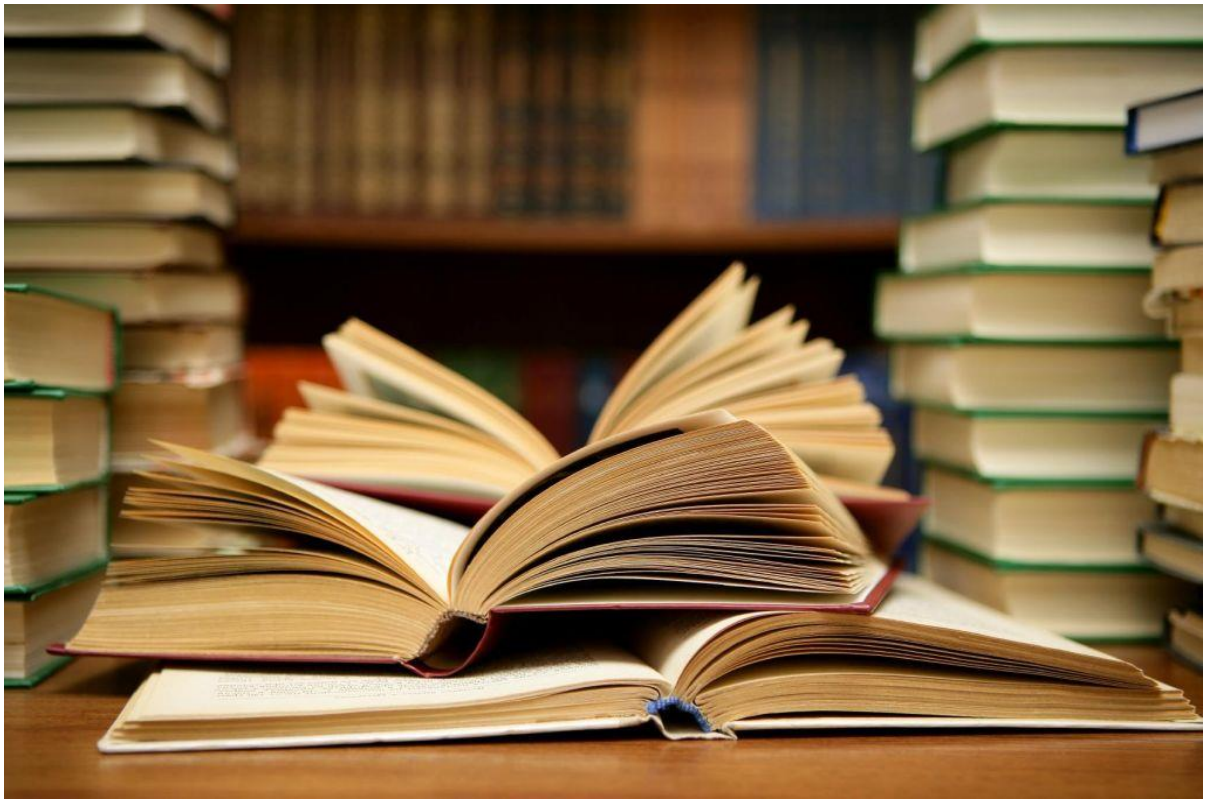
universidade de aveiro

BIBLESTIA

Sistema de Gestão de bibliotecas



catar

Índice

Introdução

Para a realização do projeto final de Base de Dados, decidimos criar uma base de dados que representa um sistema de gestão de bibliotecas. Utilizamos SQL Server e C# para a criação desta, com o intuito de aplicar os conhecimentos adquiridos ao longo do semestre da disciplina Base de Dados.

Este sistema tem em conta as ações dos leitores em cada biblioteca, e ainda a participação em atividades e a requisição de material.

Todas as transações descritas neste trabalho poderão ser visualizadas usando a interface gráfica *Biblestia,* que foi implementada utilizando a ferramenta Visual Studio.

Os materiais de consulta usados para a criação deste projeto foram os slides disponibilizados pelos professores, os guiões práticos e a documentação SQL e Visual Studio encontrada na Internet.

Para a realização deste projeto os elementos deste grupo trabalharam os dois pelo GitHub .

https://github.com/tiagosora/BD-Project-Biblestia

Ficheiro Enviados

O ficheiro .zip enviado contém os ficheiros necessários para executar o sistema. Todas as estruturas criadas foram separadas entre os seus respetivos ficheiros.

Pasta "Biblestia": Pasta com os ficheiros da interface criados para a manipulação da base de dados.

Pasta "SQL Server Queries":

- **UDFs.sql** -> Documento SQL com todas as User Defined Functions criadas.
- **SPs.sql ->** Documento SQL com todas as Stored Procedures criadas.
- **Views.sql** -> Documento SQL com todas as Views criadas.
- **Indexes.sql** -> Documento SQL com todos os Índices criados.
- **Triggers.sql** -> Documento SQL com todos os Triggers criados.
- **CreateSchema.sql** -> Documento SQL para a criação do esquema que suporta o sistema.
- **TableCreations.sql** -> Documento SQL com a criação das tabelas necessárias ao sistema e com as suas primary/foreign keys.
- **Insertions.sql** -> Documento SQL que adiciona os dados nas tabelas.
- **SystemReset.sql** -> Documento SQL que pode ser executado repetitivamente e reinicia o sistema completo.

Análise de requisitos

No nosso trabalho temos as seguintes entidades:

Biblioteca: Uma biblioteca é um edifício caracterizado por um nome, uma morada, um número de telefone e um endereço de e-mail. Cada biblioteca tem ainda funcionários e leitores, proporcionando aos seus leitores a possibilidade de participar em atividades e de requisitar materiais disponíveis. Esta também é a entidade nuclear onde todas as outras entidades dependem totalmente desta.

Funcionário: Um funcionário é definido por um nome, um NIF, uma data de nascimento, uma morada, um endereço e-mail, um número de segurança social, um ID de trabalhador e um cargo na biblioteca. Para além disso, o funcionário pode ser responsável por atividades da biblioteca e pode processar requisições de material. Um funcionário pode ter trabalhado em outras bibliotecas, mas apenas trabalha numa por vez.

Cargo: Um funcionário pode, não simultaneamente, ter vários cargos na mesma biblioteca. Assim sendo, o cargo de um funcionário é caracterizado por uma data de início e fim do exercer dessa função.

Leitor: Um leitor é definido por um nome, um NIF, um ID na biblioteca, uma morada, uma data de nascimento e um endereço e-mail.

Atividade: Uma atividade tem um nome, um tema, uma data de realização e uma duração. Uma atividade pode ser aderida por múltiplos leitores e deve sempre ter um funcionário delegado para ser responsável.

Material: Os materiais disponíveis numa biblioteca são identificados por um ID, por um estado de requisição (disponível ou requisitado) e pela secção da biblioteca em que possa estar exposto. A biblioteca disponibiliza livros, jornais, revistas, filmes e jogos.

Livro: Os livros são definidos por um título, um autor, um ano de reprodução, uma editora e um género literário.

Jornal: Um jornal é caracterizado por um nome, data de publicação e uma editora.

Revista: As revistas são definidas por um nome, uma data de publicação, uma editora, um género.

CD: Um CD é definido por um nome, uma categoria (áudio, vídeo, misto), um ano de produção e a marca que o produziu.

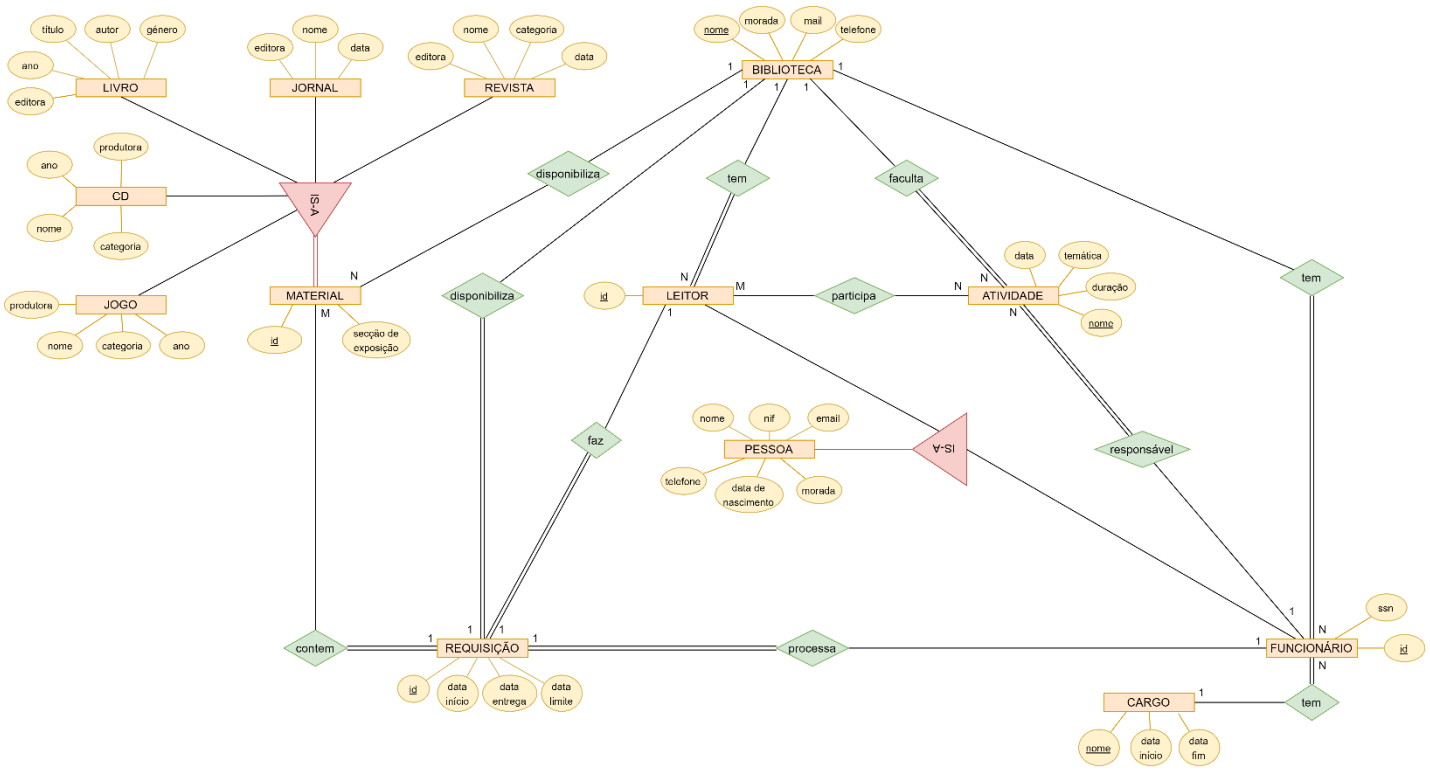
Jogo: Cada filme é caracterizado por um nome, um ano de produção, uma categoria e a marca que o produziu.

Requisição: Uma requisição é caracterizada por um ID na biblioteca, o funcionário que a processou, um material, uma data de requisição, uma data de entrega, uma data limite para a entrega, e o leitor que requisitou o material.

DER

Tendo em conta as entidades definidas na análise de requisitos, construímos o Diagrama de Entidades Relacional correspondente.

Esta versão é a versão final do DER, uma vez que este sofreu algumas alterações desde a primeira versão deste. Também foi aqui inserida uma entidade Pessoa.



Esquema Relacional

O esquema seguinte é a tradução de Diagrama de Entidades Relacional que foi apresentado anteriormente.

Para tratar das relações N para M criaram-se duas tabelas RequisiçãoMaterial e AtividadeLeitor.

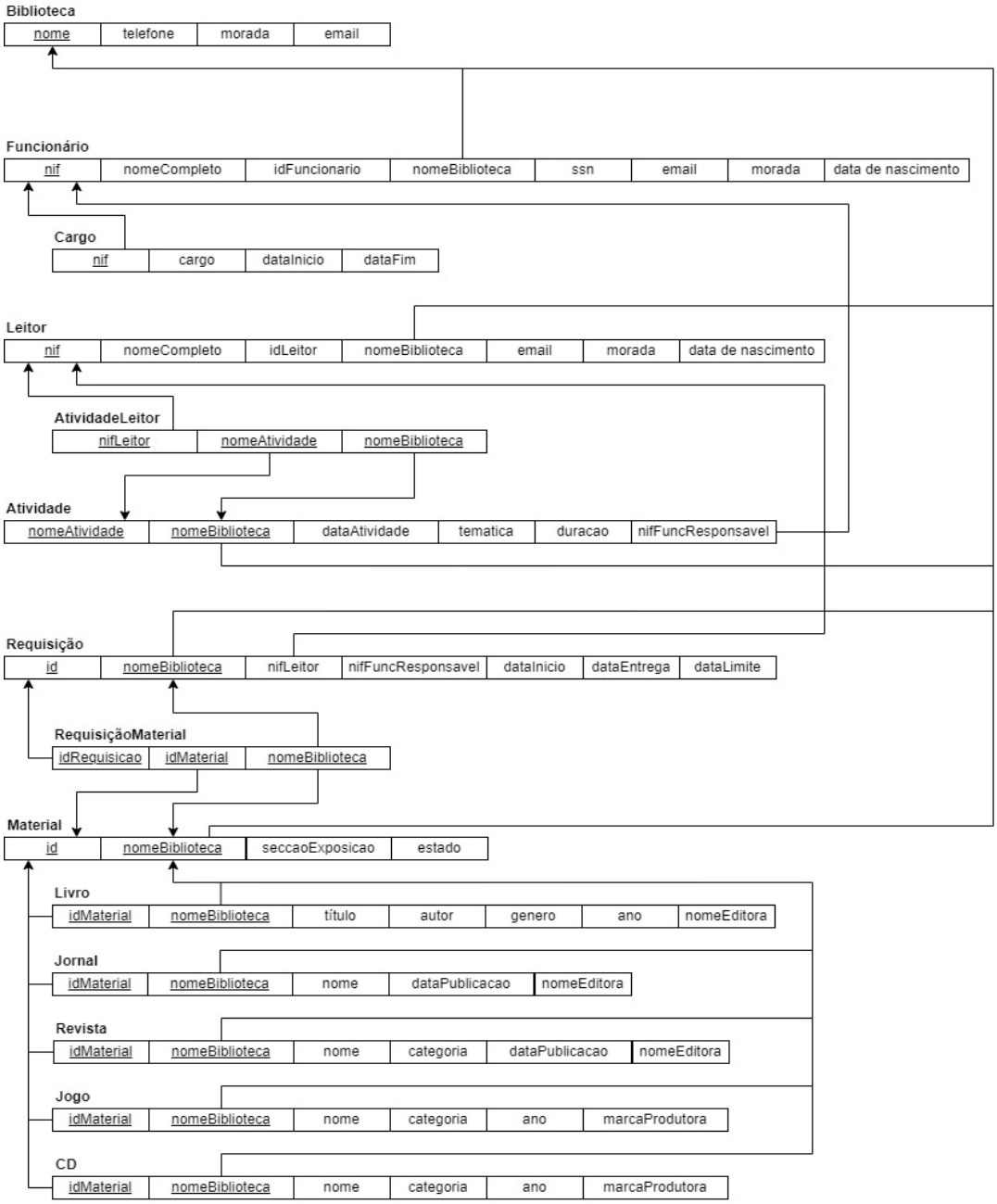
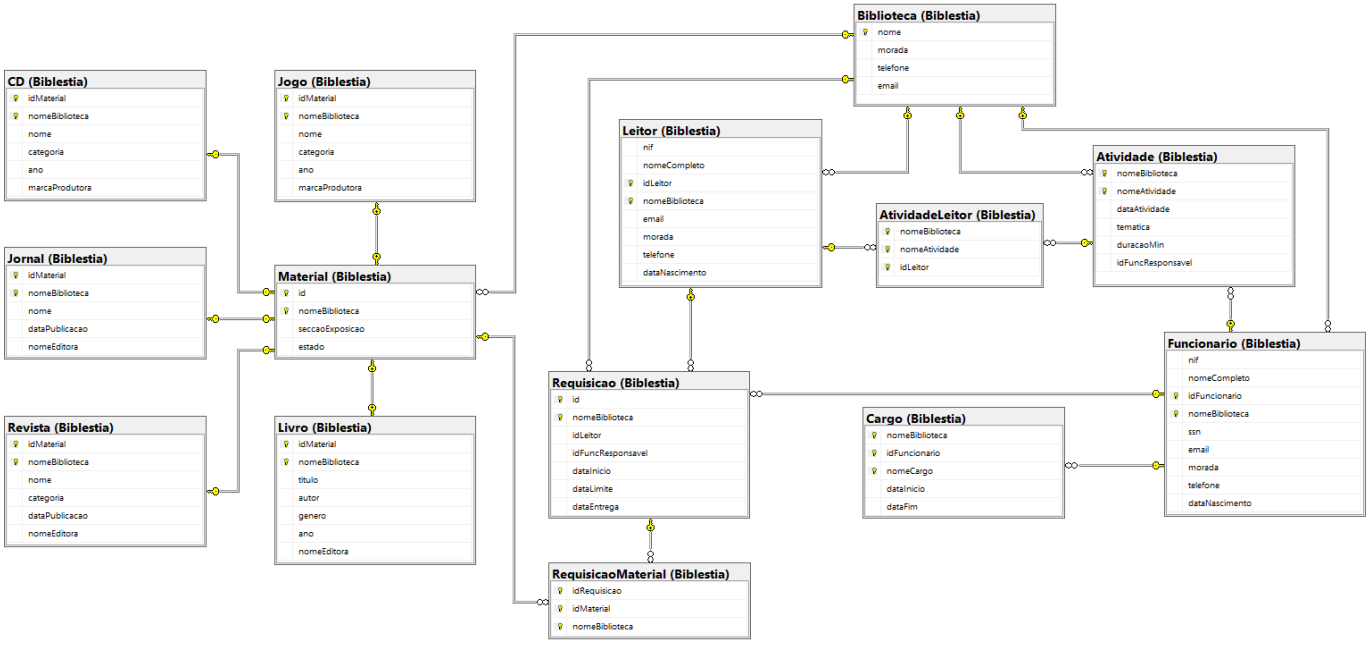


Diagrama da BD

Nas múltiplas entidades definidas, temos as seguintes dependências.



Normalização

Após a análise da Base de Dados definidas as relações e as entidades necessárias para o funcionamento da mesma. Durante todo o processo foram respeitadas as regras impostas com o intuito de obter a Terceira Forma Normal (3FN), respeitando as formas normais anteriores.

Interfaces

Como já referimos anteriormente, recorremos aos Windows Forms e à linguagem C# para criar a interface gráfica da *Biblestia*. Neste projeto criamos os seguintes Forms:

Biblestia: Interface de acesso e de seleção da biblioteca em interesse.

Main: Interface de trabalho da biblioteca. Aqui podemos consultar vários tipos de dados e fazer vários tipos de funções:

- **Leitores**: aqui podemos adicionar, editar e eliminar um leitor, assim como ver a lista de leitores da biblioteca, as requisições do leitor e as atividades deste mesmo.
- **Funcionários**: aqui podemos adicionar, editar e eliminar um funcionário, assim como, editar a lista de cargos deste mesmo. Ainda podemos ver o número total de funcionários, os funcionários atuais e a sua percentagem.
- **Requisições**: aqui podemos adicionar, editar e eliminar uma requisição, assim como ver as requisições do leitor e ver dados avançados desta requisição. Ainda conseguimos ver o histórico de requisições e as requisições atuais.
- **Atividades**: aqui podemos adicionar, editar e eliminar atividades, assim como ver a lista de participantes da atividade e aceder ao leitor mais ativo.
- **Materiais**: aqui podemos adicionar, editar e eliminar materiais, assim como ver as requisições de um certo material e as estatísticas deste tipo de Material. Ainda podemos ver em que estado cada material de encontra (Requisitado e Disponível).
- **Sair**: sair do forms Main.

Stored Procedures

adicionarFuncionario -> adiciona um funcionário a uma lista de funcionários

adicionarLeitor -> adiciona um leitor a uma lista de leitores

adicionarAtividade -> adiciona uma atividade a uma lista de atividades

adicionarCargo -> adiciona um cargo de um funcionário

editarFuncionario -> edita um funcionário de uma biblioteca

editarLeitor -> edita um leitor de uma biblioteca

editarCargo -> edita um cargo de um funcionário de uma biblioteca

editarAtividade -> edita uma atividade de uma biblioteca

removerFuncionario -> remove um funcionário da lista de funcionários de uma biblioteca

removerLeitor -> remove um leitor da lista de funcionários de uma biblioteca

removerAtividade -> remove uma atividade de uma biblioteca

removerCargo -> remove um cargo de um funcionário

saberTipo -> sabe o tipo de um dado material de uma biblioteca

removerParticipacao -> remove um leitor de uma participação em uma atividade

adicionarParticipacao -> adicionar um leitor de uma participação em uma atividade

User Defined Functions

Em relação à entidade Funcionário temos as seguintes UDF's:

- obterFuncionários
- obterFuncionariosAtuais
- obterCargosFuncionario

Em relação à entidade Leitor temos as seguintes UDF's:

- obterLeitores
- obterAtividadesLeitor
- obterDadosLeitor
- nAtividadesLeitor
- obterScoreLeitor

Em relação à entidade Atividades temas as seguintes UDF's:

- obterAtividadesLeitor
- obterAtividades
- obterAtvidadesResponsavel
- nAtividadesLeitor
- nLeitoresParticipantes
- leitoresAtividades
- nAtividades
- obterDadosLeitor
- nLeitoresAtividade

Em relação à entidade Materias temos as seguintes UDF's:

- obterMateriais
- obterDadosMateriais
- obterRequisicoesMaterial

Em relação à entidade Requisição temos as seguintes UDF's:

- obterRequisicoes
- obterRequisicoesMaterial
- obterLivrosRequisicao
- obterJornaisRequisicao
- obterCDsRequisicao
- obterJogosRequisicao
- obterRevistasRequisicao

- obterRequisicoesLeitor
- nRequisicoes
- obterRequisitor
- nReq (obter número de requisições)
- nLivrosReq
- nJornalReq
- nJogoReq
- nCDsReq
- nRevistaReq

Em relação à entidade Livros temos as seguintes UDF's:

- nLivrosReq
- obterLivros
- obterDadosLivro
- obterNumeroTotalLivros
- obterContGeneros

Em relação à entidade Jornais temos as seguintes UDF's:

- obterJornais
- nJornalReq
- obterDadosJornal
- obterNumeroTotalJornais
- obterContEditora

Em relação à entidade Jogos temos as seguintes UDF's:

- obterJogos
- obterDadosJogo
- obterNumeroTotalJogos
- obterContCategoriaJogos
- nJornalReq

Em relação à entidade CD temos as seguintes UDF's:

- obterCDs
- obterDadosCD
- obterNumeroTotalCDs
- obterContTipoCds
- nCDsReq

Em relação à entidade Revista temos as seguintes UDF's:

- obterRevistas

- obterDadosRevista
- obterNumeroTotalRevistas
- obterContCategoria
- nRevistaReq

Em relação à entidade Livro temos as seguintes UDF's:

- obterDadosLivro
- obterContGeneros
- nLivrosReq
- obterLivros
- obterNumeroTotalLivros

Triggers

Aqui estão os *Triggers* definidos para a eliminação. Este estão organizados por ordem de dependência das entidades da tabela (menos dependente para o mais dependente):

- **Delete_Biblioteca** -> trigger para eliminar uma biblioteca
- **Delete_Funcionario** -> trigger para eliminar um funcionário
- **Delete_Leitor** -> trigger para eliminar um leitor
- **Delete_Atividade** -> trigger para eliminar uma atividade
- **Delete_Requisicao** -> trigger para eliminar uma requisição
- **Delete_Material** -> trigger para eliminar um material
- **checkDatesFuncionario** -> trigger para verificar se as datas são coerentes

Views

```
drop view Biblestia.RequisicaoDados;
go
create view Biblestia.RequisicaoDados
as
    select Requisicao.*, RequisicaoMaterial.idMaterial, Funcionario.nomeCompleto as nomeCompletoFuncResponsavel, Leitor.nomeCompleto as nomeCompletoLeitor
    from    ((Biblestia.Requisicao join Biblestia.RequisicaoMaterial on id=idRequisicao)
            join Biblestia.Funcionario on idFuncionario=idFuncResponsavel)
            join Biblestia.Leitor on Requisicao.IdLeitor=Leitor.idLeitor;
go

drop view Biblestia.LeitoresTaxaParticipacao;
go
create view Biblestia.LeitoresTaxaParticipacao
as
    select  contAti, contReq, contAti*contReq as score, Leitor.*
    from    ((Biblestia.LeitoresNumeroAtividades join Biblestia.LeitoresNumeroRequisicoes
                on (LeitoresNumeroAtividades.idLeitor=LeitoresNumeroRequisicoes.idLeitor
                and LeitoresNumeroAtividades.nomeBiblioteca=LeitoresNumeroRequisicoes.nomeBiblioteca))
            join Biblestia.Leitor on LeitoresNumeroAtividades.idLeitor=Leitor.idLeitor)
go

drop view Biblestia.LeitoresNumeroAtividades;
go
create view Biblestia.LeitoresNumeroAtividades
as
    select nomeBiblioteca, idLeitor, count(*) as contAti from Biblestia.AtividadeLeitor
    group by nomeBiblioteca, idLeitor
go

drop view Biblestia.LeitoresNumeroRequisicoes;
go
create view Biblestia.LeitoresNumeroRequisicoes
as
    select nomeBiblioteca, idLeitor, count(*) as contReq from Biblestia.Requisicao
    group by nomeBiblioteca, idLeitor
go
```

Index

indexDataAtividade:

```
use p6g7;
go

drop index indexDataAtividade on Biblestia.Atividade;
go
create nonclustered index indexDataAtividade
on Biblestia.Atividade(nomeAtividade asc)
go
```

Conclusão

Com a realização deste projeto, pudemos ver em prática o funcionamento e a utilidade de um Sistema de Gestão de Base de Dados, bem como a segurança dos dados e as vantagens de ter a informação centralizada num só local.

Também o facto deste trabalho pedir uma interface intuitiva e clara, permitiu com que a visualização e a disposição dos dados e a interação com estes mesmos, fosse de maneira mais fluída.

Concluindo, os requisitos propostos no início do projeto forma realizados com sucesso, a especialização dos conteúdos relacionados com a matéria lecionada na cadeira de Base de Dados.