ADDETC – Área Departamental de Engenharia Eletrónica e Telecomunicações e de Computadores

LEIM -Licenciatura Engenharia informática e multimédia

Sistemas de base de dados

Trabalho prático 2

Turma:

LEIM-51D

Trabalho realizado por:

Pedro Ferreira Nº45112 email: a43747@alunos.isel.pt

Duarte Domingues Nº45140 email: a45140@alunos.isel.pt

Docente:

Porfírio Filipe

Data de entrega:

29/01/2021

Conteúdo

Introdução

O trabalho está dividido em duas partes:

- A primeira parte do trabalho prático da unidade curricular de Sistemas de Bases de Dados tem como objetivo consolidar e aplicar os conhecimentos sobre conceção de modelos conceptuais, lógicos e físicos para bases de dados relacionais. Utilizar a linguagem SQL para criar bases de dados. Tem como tema o conceito de uma comunidade de amigos querer implementar um sistema de informação para suportar a partilha de conteúdos multimédia do tipo, nomeadamente: filme, música, fotografia ou poema. Este trabalho numa fase inicial requer que seja desenhada uma estrutura de dados para armazenar informações de todo o seu fluxo de operações e funcionamento e de seguida a implementação numa base de dados.

- Na segunda parte do trabalho da unidade curricular de Sistemas de Bases de Dados tem como objetivo a consolidar aspetos de conceção e implementação de bases de dados relacionais. Utilizar a linguagem Structured Query Language (SQL) e Java Database Connectivity (JDBC), para criar, alterar e interrogar bases de dados utilizando tecnologias Java no contexto da World Wide Web (WWW):

Conceção

1ºParte

MODELO ENTIDADE ASSOCIAÇÃO

Para o desenvolvimento do modelo Entidade Associação foram recolhidas as restrições existenciais do enunciado deste trabalho prático.

As restrições recolhidas foram as seguintes:

- Sobre os users é necessário manter registo do seu nome, password, nacionalidade, data de nascimento e reputação.

- Se os users acedem a um recurso podem comentá-lo e avaliá-lo, caso o recurso seja carregado o user recebe pontos de reputação.

- A data do comentário tem de ser posterior á da publicação do recurso.

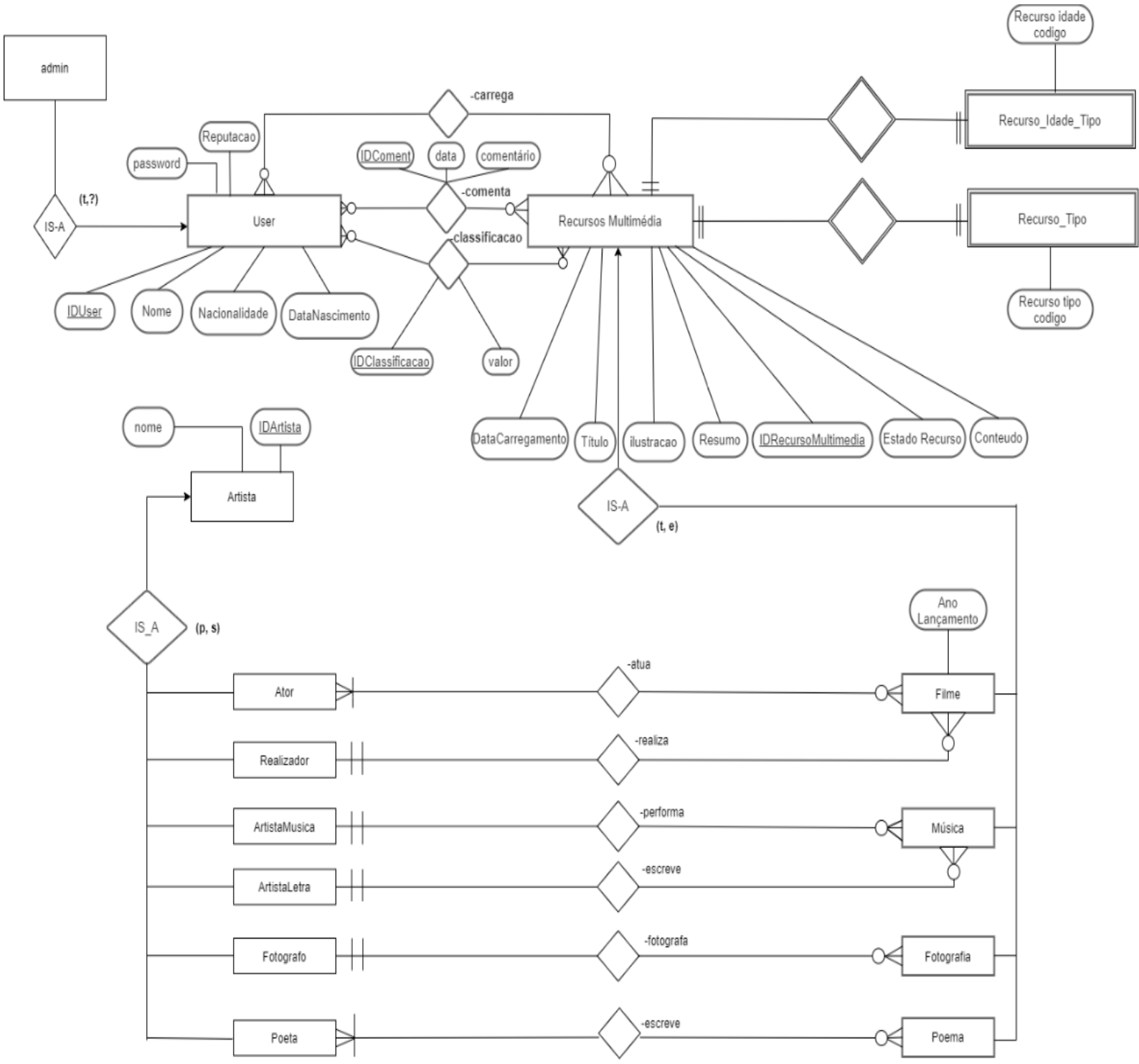
- Cada recurso possui um conteúdo, resumo, uma ilustração, data de carregamento, um título e um estado de recurso que será determinado pelo admin (bloqueado ou não).

- Um recurso multimédia se for do tipo filme tem registado o nome do realizador, atores principais e ano de lançamento. Se for do tipo música tem associado o autor da música e da letra. Caso seja do tipo fotografia tem o nome do fotografo. Já se for tipo poema apenas tem registado o nome dos autores.

- A data de carregamento de um recurso (filme) tem de ser posterior à data de lançamento do mesmo.

- Os atores e realizadores só podem fazer parte de recursos do tipo filme, os artistas de música e de letras só podem fazer parte de recursos do tipo música, os fotógrafos apenas podem ser associados a fotografias já os poetas só podem fazer parte de poemas.

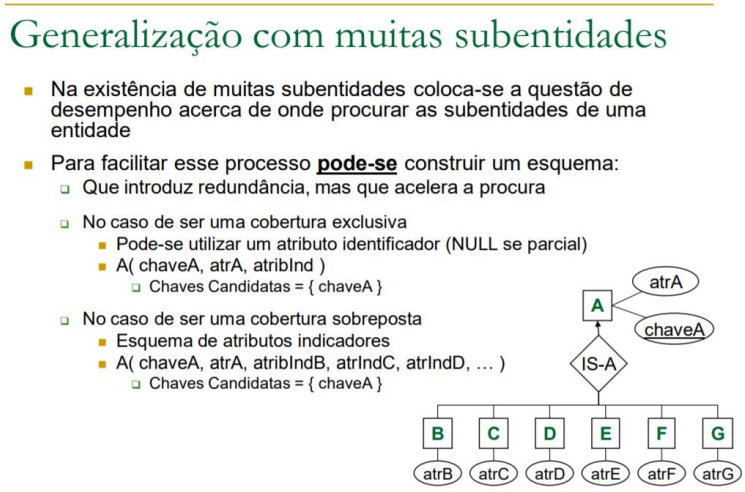
  Tendo estas informações foi então estipulado e delineado o seguinte modelo entidade associação.


admin
IS-A
(t,?)
User
Reputacao
password
IDUser
Nome
Nacionalidade
DataNascimento
-carrega
IDComent
data
comentário
-comenta
classificacao
IDClassificacao
valor
Recursos Multimédia
Recurso idade codigo
Recurso_Idade_Tipo
Recurso_Tipo
Recurso tipo codigo
DataCarregamento
Título
ilustracao
Resumo
IDRecursoMultimedia
Estado Recurso
Conteudo
IS-A
(t, e)
nome
IDArtista
Artista
IS_A
(p, s)
Ator
Realizador
ArtistaMusica
ArtistaLetra
Fotografo
Poeta
-atua
-realiza
-performa
-escreve
-fotografa
-escreve
Ano Lançamento
Filme
Música
Fotografia
Poema

MODELO RELACIONAL

O modelo relacional permite criar um modelo lógico consistente da informação a ser armazenada, realiza a conversão entre a modelação inicial do problema para uma base de dados funcional. Para tal, é necessário eliminar qualquer redundância de dados e outras irregularidades no modelo desenvolvido. Após se finalizar o modelo entidade associação, foi então definido o modelo relacional.

Recursos Multimédia, filme, música, fotografia e poema



Sendo a entidade Recurso Multimédia uma generalização de filme, música, fotografia e poema, efetuou-se a transformação tendo em conta a ligação IS-A exclusiva, sendo assim RecursoMultimedia ficará com os atributos em comuns de todas as subentidades e a chave primaria das subentidades será a mesma da entidade RecursoMultimedia:

RecursoMultimedia(idRecurso, ilustração, recursoTipo, ilustração, conteúdo, titulo, dataCarregamento, resumo, estadoRecurso, userCriador, idadeTipo)

Chaves candidatas={idRecurso}

Chaves estrangeiras={{userCriador→User}}

Filme (idRecurso ,anoLancamento, idRealizador)

Chaves estrangeiras={{idRealizador →Artista},{idRecurso →RecursoMultimedia}}

Música (idRecurso, idAutor,,idAutorLetra, idRecurso)

Chaves estrangeiras={{idArtistaMusica, idArtistaLetra →Artista}, {idRecurso → RecursoMultimedia}}

Fotografia (idRecurso, idFotografo, idRecurso)

Chaves estrangeiras={{idFotografo → Artista},{idRecurso →RecursoMultimedia}}

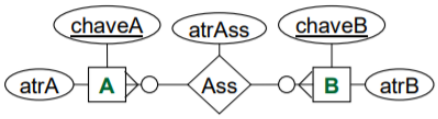
Poema (idRecurso)

Chaves estrangeiras={{idRecurso → RecursoMultimedia}}

Ator_Atua_Filme e Poeta_Atua_Poesia

Associação binária M:N (M-0 : 0-N)

- Facultativo nos dois lados



- A associação é modelada com um EsqRel próprio
  - A(chaveA, atrA)
    - Chaves Candidatas = { chaveA }
  - B(chaveB, atrB)
    - Chaves Candidatas = { chaveB }
  - A_B (chaveA, chaveB, atrAss)
    - Chaves Candidatas = { chaveA, chaveB }
    - Chaves Estrangeiras = { chaveA } e { chaveB }

Ator_Atua_Filme(idAtor, idFilme,)

Chaves candidatas={ idAtor , idFilme }

Chaves estrangeiras={{ idAtor →Artista},{ idFilme →RecursoMultimedia}}

Poeta_Atua_Poesia (idPoeta, idPoema)

Chaves candidatas={ idPoeta, idPoema }

Chaves estrangeiras={{idPoeta → Artista},{idPoema→ RecursoMultimedia }}

Artista, ator, realizador, artista letra, artista música, fotografo e poeta

Nesta relação temos de novo uma generalização, mas desta vez poderá ser sobreposta, então a generalização artista terá como atributos o que todas as subentidades têm em comum e as chaves primárias das subentidades será a da generalização.

Artista (idArtista, name)

Chaves candidatas={ idArtista }

Ator (idAtor)

Chaves estrangeiras={{ idAtor →Artista}}

Realizador (idRealizador)

Chaves estrangeiras={{ idRealizador →Artista}}

ArtistaLetra (idArtistaLetra)

Chaves estrangeiras={{ idArtistaLetra →Artista}}

ArtistaMusica (idArtistaMusica)

Chaves estrangeiras={{ idArtistaMusica⟶Artista}}

ArtistaLetra (idArtistaLetra)

Chaves estrangeiras={{ idArtistaLetra ⟶Artista}}

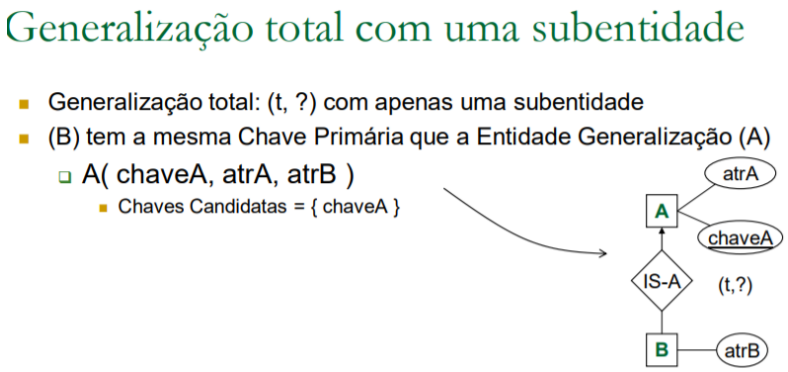
Fotografo (idFotografo)

Chaves estrangeiras={{idFotografo ⟶Artista}}

Poeta (idPoeta)

Chaves estrangeiras={{idPoeta ⟶Artista}}

User e Admin



Admin será um subentidade de User, logo a sua chave primaria é a de User:

User(idUser, name, nacionalidade, dataNascimento, reputação, password)

Chaves candidatas={idUser}

Admin(idUser)

Chaves estrangeiras={{idUser ⟶ User}}

Comentário e Classificação

Comentario(comentarioId, UserComentador, recursoId, comentário, dataComentario)

Chaves estrangeiras={{ UserComentador ⟶ User},{ recursoId ⟶ RecursoMultimedia}}

Classificacao(classificacaoId, userClassificador, recursoId, classificacaoVal)

Chaves estrangeiras={{ userClassificador ⟶ User},{ recursoId ⟶ RecursoMultimedia}}

Criação do modelo físico

Nesta etapa, realizámos a conceção do modelo físico, em sql, criando as diversas tabelas previamente planificadas nas etapas anteriores, definindo os diversos atributos, parâmetros, chaves e restrições.

Começou-se por criar a base de dados, com o comando "**create database**".

De seguida definiu-se a tabela para o utilizador, com os seus múltiplos atributos e a tabela para o administrador que é uma sub-identidade de utilizador, tendo uma *foreign key* para o ID do utilizador.

```
CREATE TABLE user(
idUser int NOT NULL auto_increment ,
name varchar(100) NOT NULL,
nacionalidade varchar(100) NOT NULL,
dataNascimento date NOT NULL,
reputacao int default(0),
password varchar(100) not null,
constraint pk_user primary key(idUser),
constraint reputacao CHECK(reputacao >= 0)
);
```

```
CREATE TABLE admin(
idUser int not null, #foregin key para user
constraint pk_admin primary key(idUser)

);
```

Após isto foi concebida a tabela para representar o recurso multimédia, tendo uma *foreign key* para o utilizador que a criou, para a tabela recurso tipo e idade tipo. Sendo que o recurso multimédia irá ter múltiplas sub-identidades.

```
CREATE TABLE recurso_multimedia(
idRecurso int NOT NULL auto_increment,
RecursoTipo char(3) not null ,
ilustracao blob not null,
conteudo blob not null,
titulo varchar(100) NOT NULL,
dataCarregamento date NOT NULL,
resumo varchar(5000),
estadoRecurso bool default(true),
userCriador int NOT NULL,
idadeTipo char(4) not null,
constraint pk_recurso primary key(idRecurso)
);
```

```
create table recursoTipo(
    recurso_tipo_codigo char(3) not null,
    primary key  (recurso_tipo_codigo)
);

create table recursoIdadeTipo(
    recurso_idade_codigo char(4) not null,
    primary key  (recurso_idade_codigo)
);
```

De seguida, foram feitas tabelas para o comentário e para a classificação, os comentários têm a referência do utilizador que realizou o comentário, e o recurso comentado. A classificação vai ter um valor inteiro entre 0 e 5, tendo a referência para o utilizador e recurso.

```
CREATE TABLE classificacao(
classificacaoId int not null auto_increment,
UserClassificador int not NULL,
RecursoId int not null,
classificacaoVal int not null ,
constraint pk_classificacao primary key(classificacaoId),
constraint classificacaoVal CHECK(classificacaoVal >= 0 and classificacaoVal <=5)
);
```

```
CREATE TABLE comentario(
comentarioId int not null auto_increment,
UserComentador int not NULL,
RecursoId int not null,
comentario varchar(5000) NOT NULL,
#classificacao int ,
dataComentario date,
constraint pk_comentario primary key(comentarioId)
);
```

Em relação ao caso da criação das tabelas e das suas diversas sub-identidades, criou-se uma tabela para representar um artista com o seu id e nome. Realizaram-se também diversas tabelas para os diferentes tipo de artista, como por exemplo, os realizadores e atores. Estas tabelas referenciam o artista e têm como chave principal o ID do artista, permitindo um artista ser de múltiplos tipos ao mesmo tempo, como por exemplo um artista ser poeta e ator.

```
CREATE TABLE artista(

idArtista int NOT NULL auto_increment,
name varchar(100) NOT NULL,
constraint pk_artista primary key(idArtista) );
);
```

```
create table ator(
    idAtor int not null,
    tag char (3) default "ATR",
    constraint fk_ator foreign key (idAtor) references artista(idArtista),
    constraint pk_ator primary key(idAtor)
```

```
create table realizador(
    idRealizador  int not null,
    tag char (3) default "RLZ",
    constraint fk_realizador foreign key (idRealizador) references artista(idArtista),
    constraint pk_realizador primary key(idRealizador)
);
```

```
create table artistaLetra(
    idArtistaLetra int not null,
    tag char (3) default "ATL",
    constraint fk_artistaLetra foreign key (idArtistaLetra) references artista(idArtista),
    constraint pk_artistaLetra primary key(idArtistaLetra)
);
```

Apos a criação dos diversos artistas, com a definição dos seus diversos tipos, foi necessário a criação de tabelas para os múltiplos géneros de conteúdo multimédia, como por exemplo, filmes e poemas. As tabelas dos conteúdos multimédia têm referências para os artistas que participam nelas, como por exemplo, na fotografia existe uma *foreign key* para referenciar um fotografo.

```
CREATE TABLE fotografia(
idFotografo int NOT NULL,
idRecurso int not null,
constraint pk_fotografia primary key(idRecurso)
);
```

```
CREATE TABLE musica(
idAutor int NOT NULL,
idAutorLetra int NOT NULL,
idRecurso int not NULL,
constraint pk_musica primary key(idRecurso));
```

Na realização da criação das tabelas do filme e do poema surgiu o seguinte problema, o filme pode ter diversos atores, sendo o número de atores total desconhecido, não sendo ideal criar atributos para cada ator. Em relação a este problema, a solução pensada foi criar uma tabela individual para guardar os diversos atores em relação ao filme em que atuam, e o mesmo para os poetas e os seus poemas.

```
CREATE TABLE filme(
anoLancamento date NOT NULL,
idRealizador int NOT NULL,
idRecurso int not NULL,
constraint pk_filme primary key(idRecurso)
);
```

```
CREATE TABLE poema(
idRecurso int not null,
constraint pk_poema primary key(idRecurso)
);
```

```
CREATE TABLE ator_atua_filme(
idAtor int not null,
idFilme int not null,
constraint pk_atua primary key (idAtor,idFilme),
constraint fk_atua_ator foreign key (idAtor) references ator(idAtor),
constraint fk_atua_filme foreign key (idFilme) references filme(idRecurso)
);
```

```
CREATE TABLE poeta_atua_poesia(
idPoeta int not null,
idPoema int not null,
constraint pk_atua primary key (idPoeta,idPoema),
constraint fk_atua_poeta foreign key (idPoeta) references poeta(IdPoeta),
constraint fk_atua_poema foreign key (idPoema) references poema(idRecurso)
);
```

Carregamento de dados de teste realizado por um script SQL DML

Finalmente, apôs a criação das múltiplas tabelas, inserimos diversos exemplos nas tabelas recorrendo ao comando "***insert***" e ao comando "***select***" para poder selecionar as tabelas para visualização, como visto no seguinte exemplo.

```
insert into recurso_multimedia (RecursoTipo,titulo,ilustracao,conteudo,dataCarregamento,resumo,userCriador,idadeTipo)
values ("flm","Velocidade Furiosa","img","cont",'2019-03-04',"Um oficial da polícia de Los Angeles Brian OConnor vê a
sua lealdade para com a polícia quando entra no mundo de corridas ilegais como infiltrado e se sente fascinado,
pondo assim em causa de que lado escolherá ficar.",1,"G");
```

De seguida foram realizadas algumas expressões para visualizar informação de forma especifica, como no seguinte exemplo em que se selecionam os nomes dos diversos poetas e do recurso multimédia onde participam.

```
select artista.name,recurso_multimedia.titulo
from sbd1.ator_atua_filme
    inner join sbd1.artista on ator_atua_filme.idAtor = artista.idArtista
    inner join sbd1.recurso_multimedia  ON ator_atua_filme.idFilme = recurso_multimedia.idRecurso ;
```

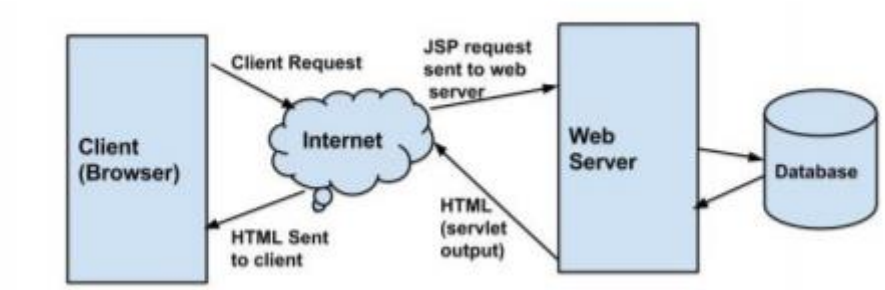
Codificação de um script SQL DDL (instruções DROP) que destrua o modelo físico

Por fim, foi feito um script com instruções *drop* para destruir o modelo físico, dando "*drop*" nas tabelas e na base de dados.

2º Parte

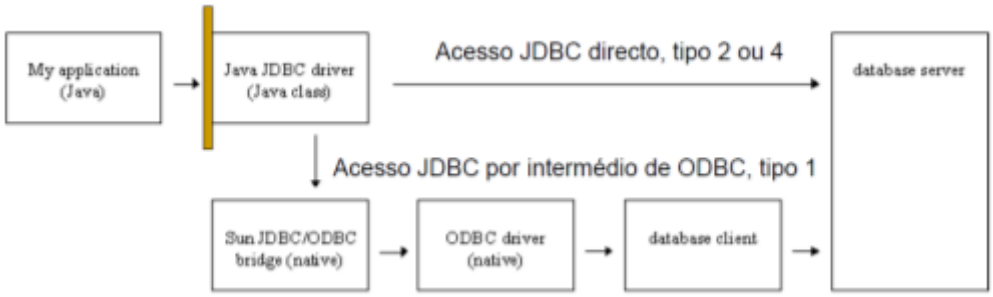
JSP

JavaServer Pages (JSP) é uma tecnologia que ajuda criar páginas web geradas dinamicamente baseadas em HTML, XML ou outros tipos de documentos.



JDBC

Java Database Connectivity é uma interface standard para permite o acesso, em código Java, a bases de dados relacionais, uma implementação do standard X/Open SQL CLI (Call Level Interface), compatível com o SQL.



Nesta parte do trabalho prático os tipos de utilizadores foram divididos em três diferentes tipos, tendo diferentes permissões e funcionalidades entre eles.

Tarefas do utilizados do tipo 1:

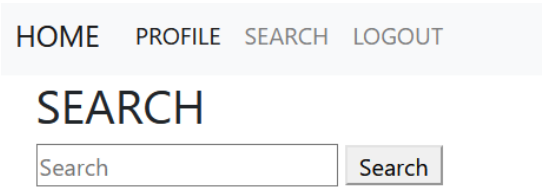
1– Procurar recursos, com filtro pelo escalão etário, indicando uma palavra relevante num controle auto complete. As faixas etárias foram divididas em cinco tipos diferentes baseado no

Motion picture content rating system.

As palavras relevantes que foram tidas em conta para a pesquisa foram: título dos artistas, título dos filmes, palavras do resumo dos recursos multimédia.

Foram realizadas *queries* para buscar esta informação à base de dados. Nos *queries mysql* utilizou-se o comando LIKE "% %" que permite fazer pesquisas consoante a presença de carateres específico, foi também usado inner join de forma a poder procurar os recursos consoante o id do artista para os diversos tipos de recurso multimédia como, por exemplo, poemas e filmes.

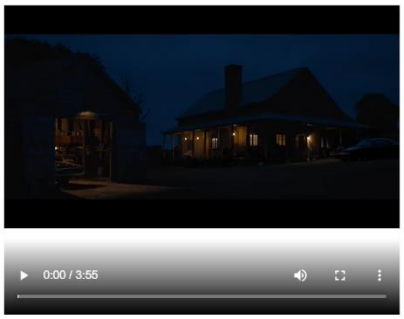
```
select idRecurso,titulo from sbd1.recurso_multimedia where (resumo)  LIKE "%g%" or titulo LIKE "%g%";


#GET IDS E NOMES DE POEMA
select recurso_multimedia.idRecurso,titulo from sbd1.poeta_atua_poesia
inner join sbd1.recurso_multimedia on recurso_multimedia.idRecurso=poeta_atua_poesia.idPoema
inner join  sbd1.artista on artista.idArtista=poeta_atua_poesia.idPoeta where (artista_name  LIKE "%a%");

#GET IDS E NOMES DE FILME
select recurso_multimedia.idRecurso,titulo from sbd1.filme
inner join sbd1.recurso_multimedia on recurso_multimedia.idRecurso=filme.idRecurso
inner join  sbd1.artista on artista.idArtista=filme.idRealizador where (artista_name  LIKE "%a%");
```



2 – Consultar/visualizar todos os atributos de um determinado recurso. Nesta tarefa de forma a não ter de realizar múltiplas *queries*, foi realizada uma *querie* para buscar múltiplos atributos dos diferentes tipos de recurso multimédia de uma vez. Também é mostrado o recurso multimédia específico, como por exemplo, um vídeo no caso dos filmes e uma fotografia no caso das páginas das fotografias. Para cada tipo de recurso foi criada uma página jsp.

```
#buscar atributos do recurso multimedia que é um filme com id equivalente a 1
SELECT * from sbd1.recurso_multimedia
inner join sbd1.filme  on filme.idRecurso = recurso_multimedia.idRecurso
inner join sbd1.artista on artista.idArtista = filme.idRealizador
 inner join sbd1.user on user.idUser = recurso_multimedia.userCriador
 where sbd1.recurso_multimedia.idRecurso =1;
```

Exemplo de uma página de recurso multimédia para um filme:



Released in: 2001-05-20
Rated: G
About:
Um oficial da polícia de Los Angeles Brian OConnor vê a sua lealdade para com a policia quando entra no mundo de corridas ilegais como infiltrado e se sente fascinado, pondo assim em causa de que lado escolherá ficar.
Director: Vin Diesel
Consult

3– Comentar e atribuir uma (única)classificação a um recurso. Nesta etapa o utilizador pode comentar múltiplos comentários e atribuir apenas uma única classificação. De forma a procurar pelas classificações consoante um id específico utilizou-se a seguinte *querie.*

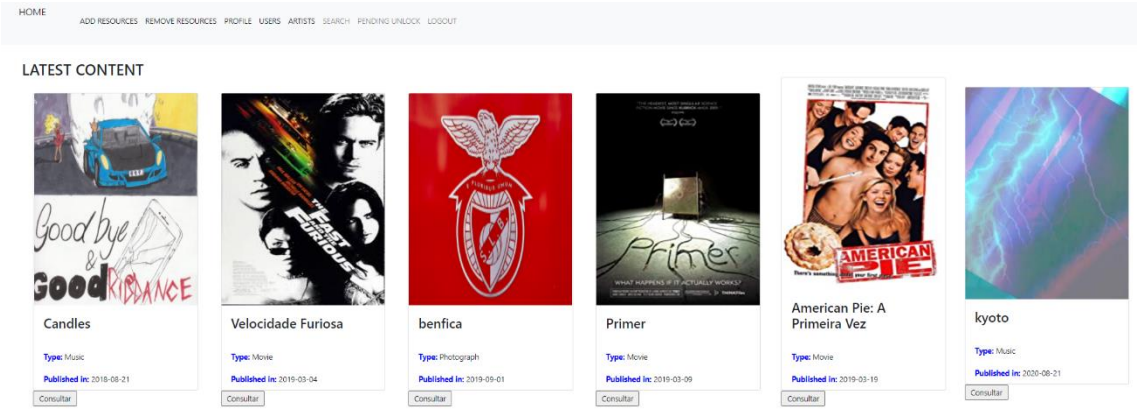
```
select classificacaoVal from sbd1.recurso_multimedia
   inner join sbd1.classificacao on classificacao.RecursoId = recurso_multimedia.idRecurso
   where recurso_multimedia.idRecurso = 2;
```

Exemplo da zona de comentários na página de um recurso multimédia.

Comments
Commented by: Jonny 2019-03-20
Bom Skyline, tá memo do aço
Commented by: Diogo 2021-01-28
che boy
Comment
Ratings
★★★★★
Rated by: Duarte
rate this movie

4– Listar os N recursos mais recentes ordenados pela classificação. Nesta etapa os utilizadores podem visualizar os recursos mais recentes ordenados pela classificação e com um filtro de faixa etária.

Recursos mais recentes.



5– Listar os recursos agrupados por pessoa mostrando o tipo de recurso e o papel da pessoa. Nesta etapa o utilizador consegue observar para um artista, o nome e os recursos onde o artista atua e o seu papel nos mesmos, um artista pode ter diferentes tipos de papel em recursos diferentes, como no seguinte exemplo:

ARTIST
Diogo Bernardão
Photographer in photo: benfica
Photographer in photo: Foto na piscina

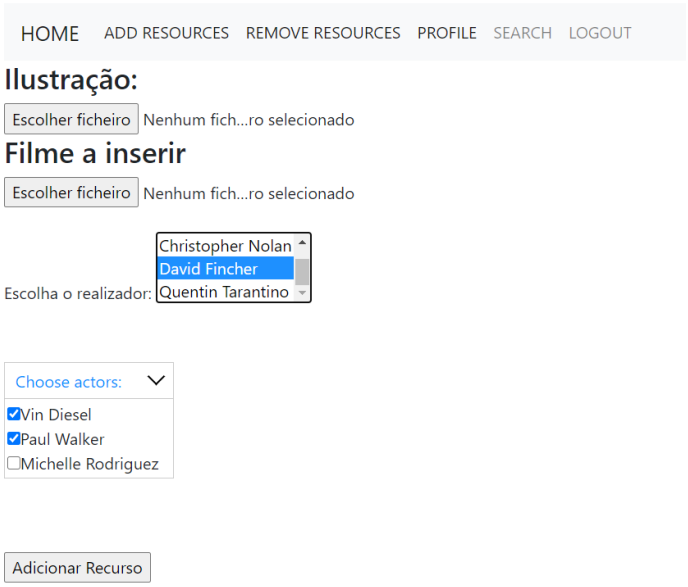
ARTIST
Vin Diesel
Director in movie: Velocidade Furiosa
Actor in movie: Velocidade Furiosa

Na página inicial existe um sistema de login/registo, e dependendo do tipo do users registado este vai ter as permissões respetivas como dito antes. Se o utilizador criar uma conta esta fica com um valor de permissões tipo 1.

Tarefas do utilizados do tipo 2:

Criar/carregar recursos

Primeiro o user escolhe o tipo de recurso que quer adicionar. No caso de ser um filme, é selecionado um ficheiro mp4, uma capa de ilustração para o recurso, um realizador e os atores que participam no filme.

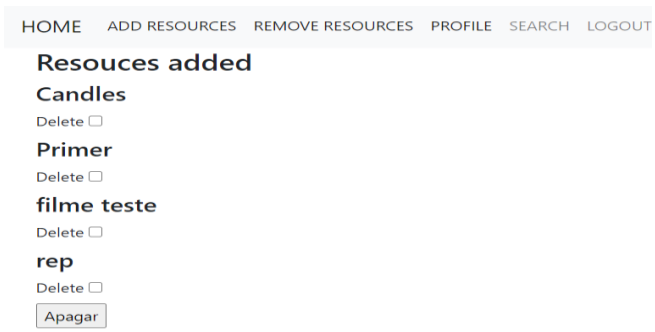


É chamada a seguinte função que ira preencher a tabela com os atributos que o user introduziu.

```
public void createRecurso2(String recursoTipo, String titulo, String ilustracao, String conteudo,
        String dataCarregamento, String resumo, int userCriador, String idadeTipo, Boolean  estadoRecurso)
                throws SQLException, FileNotFoundException {

    Connection connection = ConnectionConfigurate.getConnection();

    System.out.println("data de carregamento: "+dataCarregamento);

    File ilustracaoFile = new File(ilustracao);


    PreparedStatement pstmt = null;

    pstmt = connection.prepareStatement(
            "insert into recurso_multimedia(RecursoTipo,titulo,ilustracao,conteudo,dataCarregamento,resumo,userCriador,idadeTipo,estadoRecurso) "
                    + "values(?,?,?,?,?,?,?,?,?)");

    pstmt.setString(1, recursoTipo);
    pstmt.setString(2, titulo);
```

Remover/descarregar um recurso criado

Aparecem os recursos que o user que está logado criou e é possível selecionar 1 ou mais para serem removidos do site e consecutivamente da base de dados.



Esta função recebe um array de id de recursos e faz delete na tabela recurso_multimedia.

```java
public void removerRecursos(ArrayList<Integer> array) throws SQLException {

    Connection connection = ConnectionConfigurate.getConnection();

    String query = "delete from recurso_multimedia where idRecurso = ?";
    PreparedStatement preparedStmt = connection.prepareStatement(query);

    for(Integer r : array) {
        preparedStmt.setInt(1,r);
        preparedStmt.execute();
    }


}
```

Tarefas do utilizados do tipo 3:

1 –Associar recursos entre si, por exemplo, associar uma música a um filme ou a um poema. Nesta etapa o utilizador consegue criar associações entre diferentes tipos de recurso, para esta tarefa foi necessário criar uma tabela sql, denominada associação, que associa dois recursos pelo seu id, definindo o recurso que tem a associação e o que está a ser associado.

```sql
create table associacao(
idAssociacao int NOT NULL auto_increment ,
idRecursoPrincipal  int not null,
idRecursoSecundario  int not null,
constraint pk_associacao primary key(idAssociacao),
constraint fk_recurso_principal foreign key (idRecursoPrincipal) references recurso_multimedia(idRecurso),
constraint fk_recurso_secundario foreign key (idRecursoSecundario) references recurso_multimedia(idRecurso)

)
```

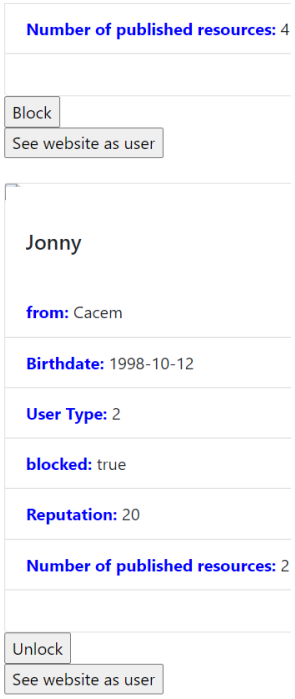
2-Bloquear ou desbloquear (1ª vez atribuir pontos ao utilizador que criou) um determinado recurso

Quando um utilizador tipo 3 inspeciona um recurso este tem como opção bloquear o recurso para todos os utilizadores.

Quando um utilizador cria um recurso, este começa logo bloqueado terá que ser visto por um utilizador tipo 3 para o desbloquear.

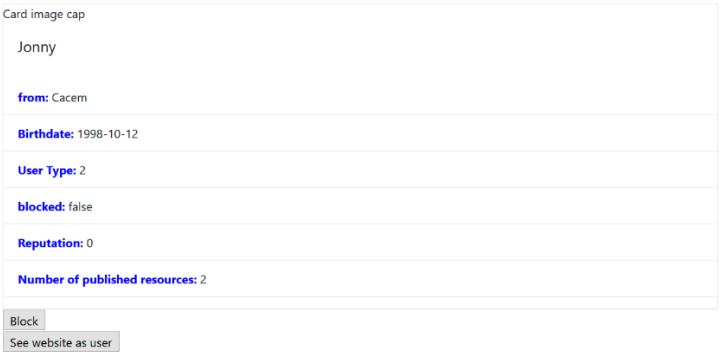
3-Bloquear ou desbloquear um determinado utilizador

Um utilizador tipo 3, consegue ver todos os atributos dos users do site e consegue bloquear-lo também,quando um user é boqueado este não consegue fazer login até ser desbloqueado por um tipo 3.



4–Listar os todos os atributos dos utilizadores ordenados por reputação e com a quantidade de recursos criados. Na página respetiva a este tipo de pesquisa é também possível bloquear ou desbloquear um user, ou assumir o seu papel.

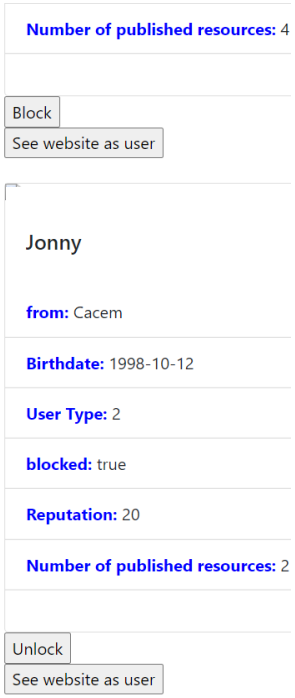
Vizualização de um user na página dos users:



5 –Apresentar todos os atributos de pessoas com os respetivos papéis e quantidade de recursos associados. Nesta etapa o utilizador consegue observar para diferentes artistas, o nome e os recursos onde o artista atua e o seu papel nos mesmos, um artista pode ter diferentes tipos de papel em recursos diferentes, como no seguinte exemplo:

6-Assumir o papel atribuído a outro utilizador

Um utilizador tipo 3 tem a opção de entrar na conta de qualquer outro user do site. Isto é feito com a alteração da sessão do iser id.



Conclusões

A elaboração desta primeira fase do trabalho serviu para colocar em prática o conhecimento adquirido na disciplina de Sistemas de Bases de Dados numa situação real de pequena escala. Ao longo do trabalho foi possível constatar e comprovar os pormenores necessários na elaboração de um modelo entidade-associação e de um modelo relacional, bem como a importância destes mesmos pois sem estes estarem corretos a base de dados não poderia ser implementada da maneira correta. Na segunda parte deu para colocar em pratica a comutação de comunicam entre java e base dados, criando uma aplicação web.

Foram criadas várias classes jsp, para realizar as páginas web, tentando tornar este trabalho um pouco mais cuidado em termos estéticos.