Licenciatura em Engenharia Informática e Multimédia
1º Semestre Letivo 2020/2021

Sistemas de Base de Dados
2º Trabalho Prático

Docente:
Porfírio Filipe

Autores:
42341, Ana Coelho, A42341@alunos.isel.pt
42346, Luís Guimarães, A42346@alunos.isel.pt
42356, Érica Pereira, A42356@alunos.isel.pt

Lisboa, 31 de janeiro de 2021

Índice

1. Conceção

1.1 Modelo Entidade Associação (MEA)

Neste segundo trabalho prático, realizaram-se algumas alterações ao Modelo Entidade Associação (MEA), para englobar as novas funcionalidades do trabalho e para permitir uma melhor acessibilidade à base de dados. No atual MEA existem doze entidades: nacionalidade, utilizador, administrador, criador, convidado, recurso, comentario, filme, fotografia, poema, música e artistas.

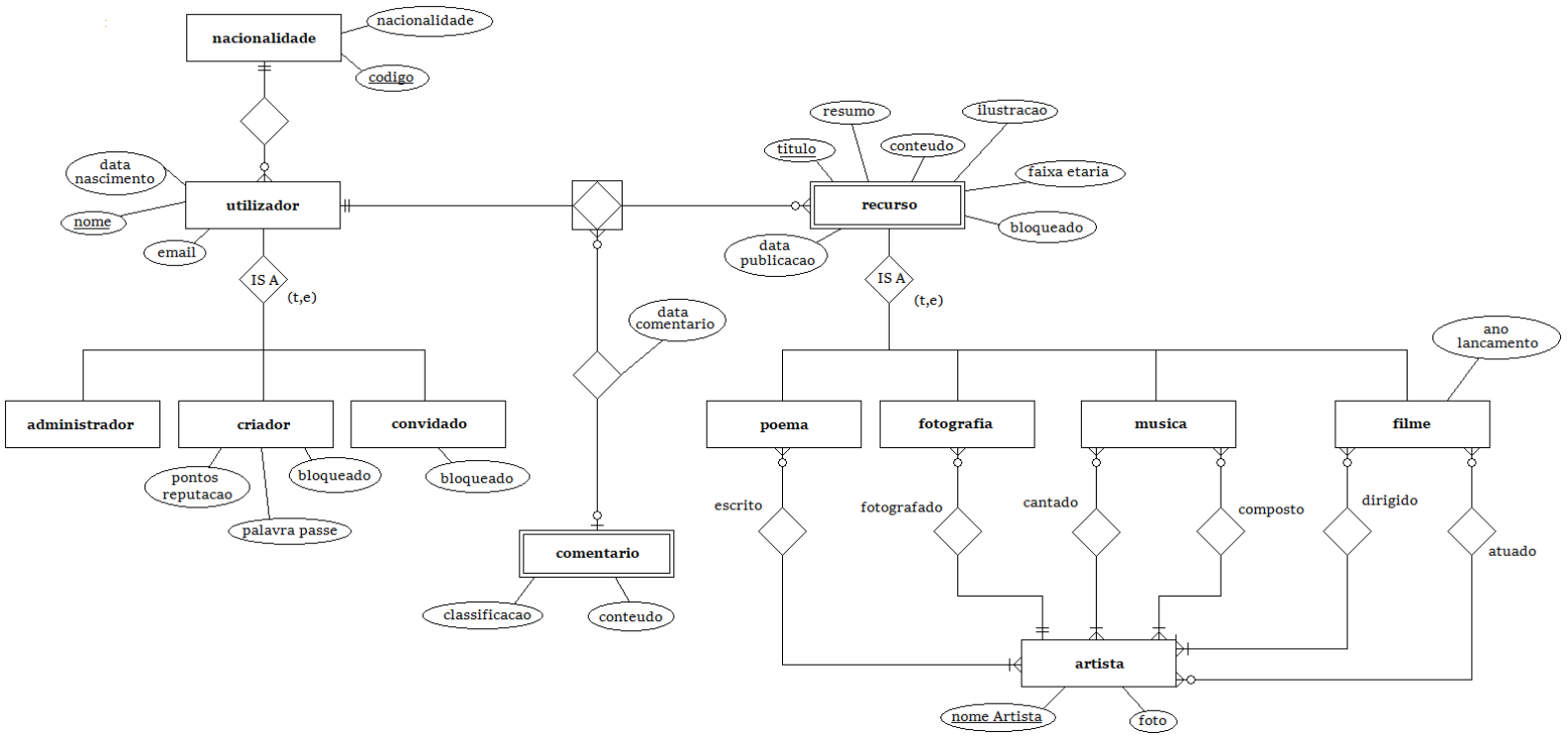


Figura 1 - Modelo Entidade Associação (MEA).

Foram adicionadas as entidades *administrador*, *criador* e *convidado* como sub-entidades da entidade *utilizador*, tratando-se agora de uma generalização.

Sendo assim, a entidade *utilizador* ficou com os atributos 'dataNascimento', 'nome' e 'email', sendo o 'nome' a chave primária para distinguir os utilizadores e o 'email' a maneira de bloquear e desbloquear os utilizadores, pois este é também um atributo único.

A entidade *criador* tem o atributo palavra passe para fazer login como criador, ou seja, publicador de recursos.

1.2 Modelo Relacional (MR)

- Nacionalidade (codigo, nacionalidade)
  - **Chave Candidata** = {{codigo}}
  - **Chave Estrangeira** = {{}}

- Utilizador (nomeUtilizador, email, dataNascimento, codigo)
  - **Chave Candidata** = {{NomeUtilizador, email}}
  - **Chave Estrangeira** = {{codigo}}

- Administrador (nomeUtilizador)
  - **Chave Candidata** = {{}}
  - **Chave Estrangeira** = {{nomeUtilizador}}

- Criador (nomeUtilizador, palavraPasse, pontosReputacao, bloqueado)
  - **Chave Candidata** = {{}}
  - **Chave Estrangeira** = {{nomeUtilizador}}

- Convidado (nomeUtilizador, bloqueado)
  - **Chave Candidata** = {{}}
  - **Chave Estrangeira** = {{nomeUtilizador}}

- Recurso (nomeUtilizador, titulo, resumo, ilustracao, bloqueado, ficheiro, faixaEtaria, dataPublicacao)
  - **Chave Candidata** = {{}}
  - **Chave Estrangeira** = {{nomeUtilizador}}

- Comentario (utilizadorComentario, titulo, utilizadorRecurso, conteudo, classificacao, dataComentario)
  - **Chave Candidata** = {{}}
  - **Chave Estrangeira** = {{nomeUtilizador, titulo}}

- Filme (nomeUtilizador, título, anoLancamento)
  - **Chave Candidata** = {{}}
  - **Chave Estrangeira** = {{nomeUtilizador, título}}

- Música (nomeUtilizador, título)
  - **Chave Candidata** = {{}}
  - **Chave Estrangeira** = {{nomeUtilizador, título}}

- Fotografia (nomeUtilizador, título)
  - **Chave Candidata** = {{}}
  - **Chave Estrangeira** = {{nomeUtilizador, título}}

- Poema (nomeUtilizador, título)
    - **Chave Candidata** = {{}}
    - **Chave Estrangeira** = {{nomeUtilizador, título}}

- Artista (nomeArtista, foto)
    - **Chave Candidata** = {{nomeArtista}}
    - **Chave Estrangeira** = {{}}

- artista_recurso (nomeArtista, tituloRecurso, carregadoPor, profissao)
    - **Chave Candidata** = {{}}
    - **Chave Estrangeira** = {{nomeArtista, titulo, nomeUtilizador}}

- tipo_recurso (titulo, carregadoPor, tipo)
    - **Chave Candidata** = {{}}
    - **Chave Estrangeira** = {{titulo, nomeUtilizador}}

- tipo_utilizador (nomeUtilizador, tipo)
    - **Chave Candidata** = {{}}
    - **Chave Estrangeira** = {{nomeUtilizador}}

- grupo_recurso (tituloRecurso, carregadoPor, idgrupo)
    - **Chave Candidata** = {{}}
    - **Chave Estrangeira** = {{titulo, nomeUtilizador}}

Atributos e Domínios:

- Nacionalidade – codigo: int
- Nacionalidade – nacionalidade: varchar(45)
- Utilizador – nomeUtilizador: varchar (45)
- Utilizador – email: varchar (45)
- Utilizador – dataNascimento: date
- Criador – bloqueado: tinyint
- Criador – pontosReputacao: decimal(10,1)
- Criador – palavraPasse: varchar (64)
- Convidado – bloqueado: tinyint
- Recurso – titulo: varchar (45)
- Recurso – resumo: varchar (200)
- Recurso – ilustracao: mediumblob
- Recurso – bloqueado: tinyint
- Recurso – ficheiro: longblob
- Recurso – faixa etaria: int
- Recurso – dataPublicacao: datetime
- Comentario – conteudo: varchar (600)
- Comentario – classificacao: enum('1','2','3','4','5')
- Comentario – dataComentario: datetime

- Filme – anoLancamento: date
- Artista – nomeArtista: varchar (45)
- Artista – foto: blob
- Artista_Recurso – profissao: varchar (45)
- Grupo_recurso – idGrupo: int
- tipo_recurso – tipo: varchar (45)
- tipo_utilizador – tipo: int

1.3 Restrições de integridade aplicacional

- Ilustração de um recurso *default*: um 'Blob' não aceita valores de *default*, tem de ser a aplicação a inserir uma imagem de *default*.

- A aplicação deve verificar a data de nascimento para saber se o utilizador pode ver ou não o recurso.

- A aplicação deve verificar se o utilizador está registado para interagir com recursos publicados.

- A aplicação tem que verificar a integridade das datas, por exemplo, se um utilizador não tem mais do que 150 anos.

- A aplicação te que verificar se o utilizador é o administrador ou não, para poder ter permissão para bloquear e desbloquear recursos, etc.

- Na associação de recursos, a aplicação tem que verificar se não existe um grupo de recursos associados duplicados.

2. Concretização da Base de Dados

2.1 Modelo Físico

A base de dados, para além das entidades, possuí também tabelas relativas a algumas associações entre entidades, tais como, artista_recurso, tipo_recurso, tipo_utilizador e grupo_recurso.

A tabela artista_recurso indica a profissão, ou seja, o(s) papel(is) que o artista tem num dado recurso.

A tabela artista_recurso indica se um recurso é um filme, um poema, uma música ou uma fotografia, facilitando a acessibilidade ao tipo de recurso que está a ser apresentado.

A tabela tipo_utilizador indica se um utilizador é um convidado, criador ou administrador, facilitando a acessibilidade ao tipo de utilizador que está "loggado".

A tabela grupo_recurso permite guardar as associações entre recursos que o administrador realiza.

3. Concretização da Aplicação

O trabalho encontra-se no ambiente eclipse C java 8-14, mysql 8 e tomcat 8.5.

Este é constituído essencialmente por classes *dataAccessObjects, transationalObjects*, *servlets e jsp's*.

As classes DAO (*dataAccessObjects)*, servem para ir buscar à base de dados a informação das tabelas através de *queries.* Estas fazem a conexão entre *Java* e *SQL.*

As *transationalObjects* são classes que servem de auxílio ao armazenamento da informação das colunas das tabelas do nosso SQL, sendo *getters* and *setters* o seu modo de acesso. Estas classes são instanciadas com os valores obtidos pelos métodos das classes DAO.

As *servlets* são classes que utilizam as DAO e *transationalObjects* para realizar um pedido proveniente do *HTML*, devolvendo a resposta através de *JSON.*

As funções dos *servlets* vão buscar os atributos ao objeto *HttpServletRequest* e em seguida validam os inputs criados no objeto *HttpSession*. Após obter a resposta das DAO, envia-se esta com *JSON*.

No browser apresentam-se os dados recebidos e fazem-se os tratamentos das exceções, exceções essas que aparecem caso não se consiga executar um dado comando ou não possa aceder a uma certa informação necessária do nosso SQL.

Para enviar os parâmetros para a servlet utilizamos a ferramenta *Ajax.* Para exibir os dados recebidos realizaram-se páginas *HTML* utilizando *JSP e JavaScript*.

Neste trabalho implementaram-se os seguintes requisitos:

Tipo 1:

1 - **Procurar recursos, com filtro pelo escalão etário, indicando uma palavra relevante num controle auto complete**

Para procurar um recurso fazemos primeiro a verificação do utilizador, vendo se ele pertence à base de dados e se as suas credenciais foram bem preenchidas. Depois dessa verificação, vamos obter os nossos recursos com filtro pelo escalão etário, criando um método que irá verificar em todos os recursos, quais os seus títulos, verificar se o mesmo não se encontra bloqueado e qual a sua faixa etária.

```
String selectRecursoByTituloAndFaixaEtaria = "select r.titulo, r.carregadoPor, m.media from (" +

                "Select titulo, carregadoPor from recurso "
                + "where titulo like ? "
                + "and bloqueado = 0 "
                + "and recurso.faixaEtaria < ? ) as r, "

                + "(SELECT tituloRecurso, utilizadorRecurso, ROUND(AVG(classificacao),1) as media "
                + "FROM comentario GROUP BY CONCAT(tituloRecurso, ' - ', utilizadorRecurso)) as m "
                + "where r.titulo = m.tituloRecurso "
                + "and r.carregadoPor = m.utilizadorRecurso; ";
```

Figura 2 - Método select criado para selecionar os recursos através do titulo e faixa etária

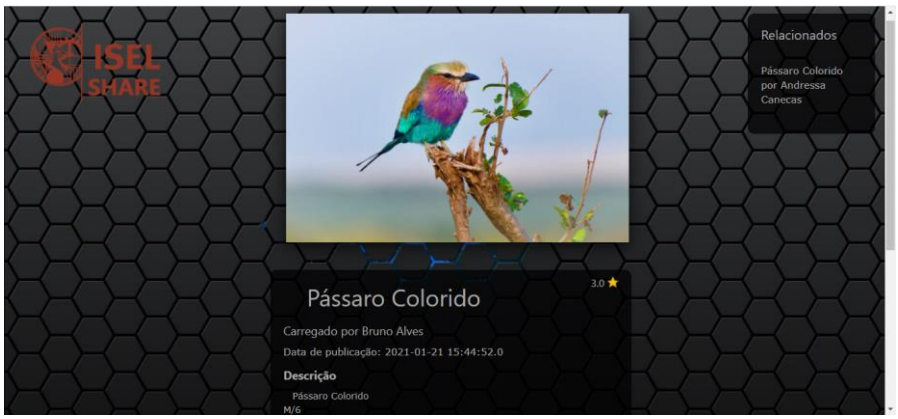
Para utilizar o filtro pelo escalão etário criámos o método *getAge():*

```
public static int getAge(Date dateOfBirth) {

    Calendar today = Calendar.getInstance();
    Calendar birthDate = Calendar.getInstance();

    birthDate.setTime(dateOfBirth);

    if (birthDate.after(today)) {
        throw new IllegalArgumentException("Ainda não nasceu");
    }

    int todayYear = today.get(Calendar.YEAR);
    int birthDateYear = birthDate.get(Calendar.YEAR);
    int todayDayOfYear = today.get(Calendar.DAY_OF_YEAR);
    int birthDateDayOfYear = birthDate.get(Calendar.DAY_OF_YEAR);
    int todayMonth = today.get(Calendar.MONTH);
    int birthDateMonth = birthDate.get(Calendar.MONTH);
    int todayDayOfMonth = today.get(Calendar.DAY_OF_MONTH);
    int birthDateDayOfMonth = birthDate.get(Calendar.DAY_OF_MONTH);
    int age = todayYear - birthDateYear;

    if ((birthDateDayOfYear - todayDayOfYear > 3) || (birthDateMonth > todayMonth)){
        age--;

    } else if ((birthDateMonth == todayMonth) && (birthDateDayOfMonth > todayDayOfMonth)){
        age--;
    }
    return age;
}
```

Figura 3 - Método getAge()

2 – **Consultar/visualizar todos os atributos de um determinado recurso**

A página contem a ilustração do recurso, todos os seus atributos, um botão para reproduzir o recurso apresenta os comentários do recurso, a opção de comentar neste e links para os recursos que lhe estão associados.



Obter atributos do recurso:

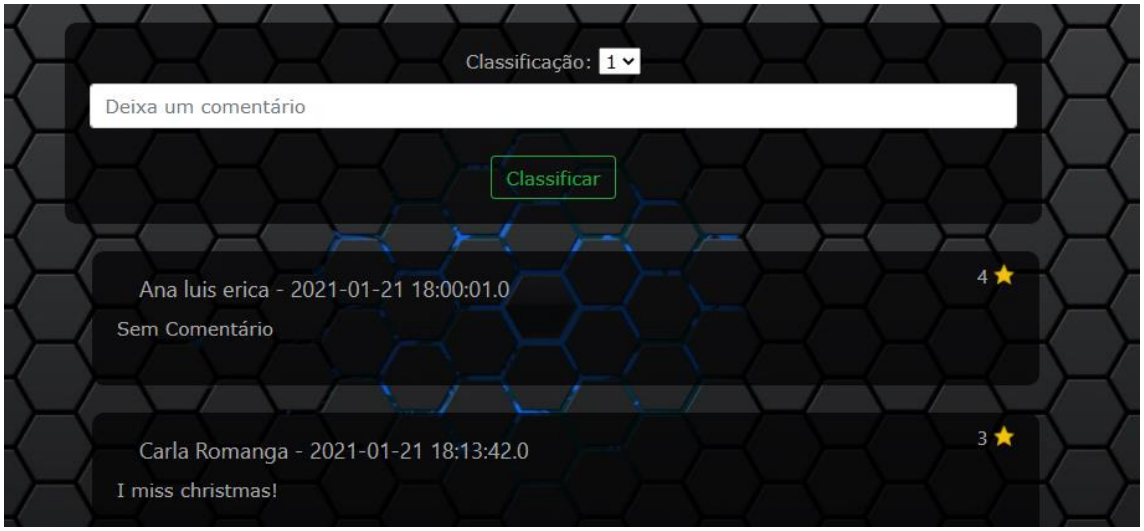
```
String selectRecursoByTituloAndFaixaEtaria =
                "Select * from recurso "
                + "where titulo = ? "
                + "and carregadoPor = ? ;";
try {
```

Obter recursos associados:

```
String selectGrupoRecursoByRecurso = "Select distinct 0, gr.tituloRecurso, gr.carregadoPor from grupo_recurso as gr "
        + "where idGrupo "
        + "in "
        + "(Select idGrupo from grupo_recurso where tituloRecurso = ? "
        + "and carregadoPor = ? ) "
        + "and not (tituloRecurso = ? "
        + "and carregadoPor = ? ) "
        + "and concat(gr.tituloRecurso, gr.carregadoPor) "
        + "in (Select concat(titulo, carregadoPor) from recurso where bloqueado = 0);";
```

3 – **Comentar e atribuir uma (única) classificação a um recurso**

Os utilizadores podem atribuir uma classificação com um comentário anexado a esta:

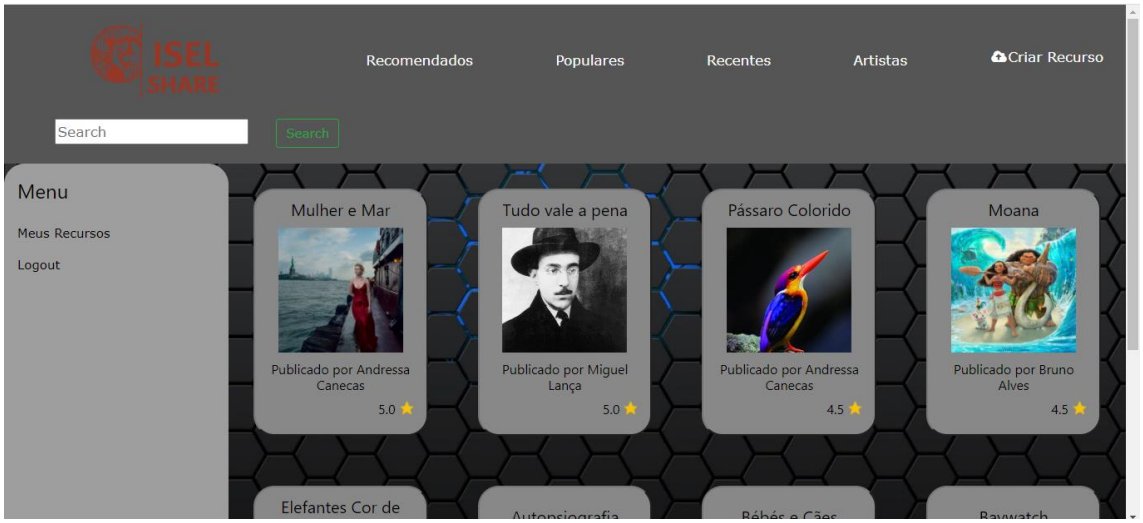


Se o utilizador já tiver comentado, faz-se um update:

```
String updateClassificacao = "UPDATE comentario SET classificacao = ?, conteudo = ?, dataComentario = CURRENT_TIMESTAMP " +
                        " WHERE (utilizadorComentario = ? ) " +
                        "and (tituloRecurso = ? ) " +
                        "and (utilizadorRecurso = ? ); ";

try {
```

4 – **Listar os N recursos mais recentes ordenados pela classificação**

Para listar os recursos começamos por verificar novamente os utilizadores, e em seguida listamos os N recursos verificando se estes não estão bloqueados, ordenando em seguida pela sua data de publicação e depois utilizando a fórmula de ROUND(AVG(classificação),1) fazemos a media da classificação para depois a poder-mos ordenar




```
String selectAllRecursos = "(SELECT titulo, carregadoPor, media from recurso, ("
        + "SELECT tituloRecurso, utilizadorRecurso, ROUND(AVG(classificacao),1) as media "
        + "FROM comentario GROUP BY CONCAT(tituloRecurso, '-', utilizadorRecurso)"
        + ") as c "
        + "where "
        + "titulo = c.tituloRecurso "
        + "and carregadoPor = c.utilizadorRecurso "
        + "and bloqueado = 0 "
        + "and recurso.faixaEtaria < ? "
        + "order by dataPublicacao DESC "
        + "limit ?) order by media DESC"
        + ";";
```

Figura 4 - Método select para ordenar os recursos pela sua classificação

5 – **Listar os recursos agrupados por pessoa mostrando o tipo de recurso e o papel da pessoa**

Para conseguirmos listar os recursos mostrando o tipo de recurso e o papel da pessoa vamos verificar através de cada tipo de recurso qual o seu nome e por quem foi inserido.

```
String selectTipoRecursoByTitleAndUser = "Select * from tipo_recurso where titulo = ? " +
                                         " and carregadoPor = ?;";
```

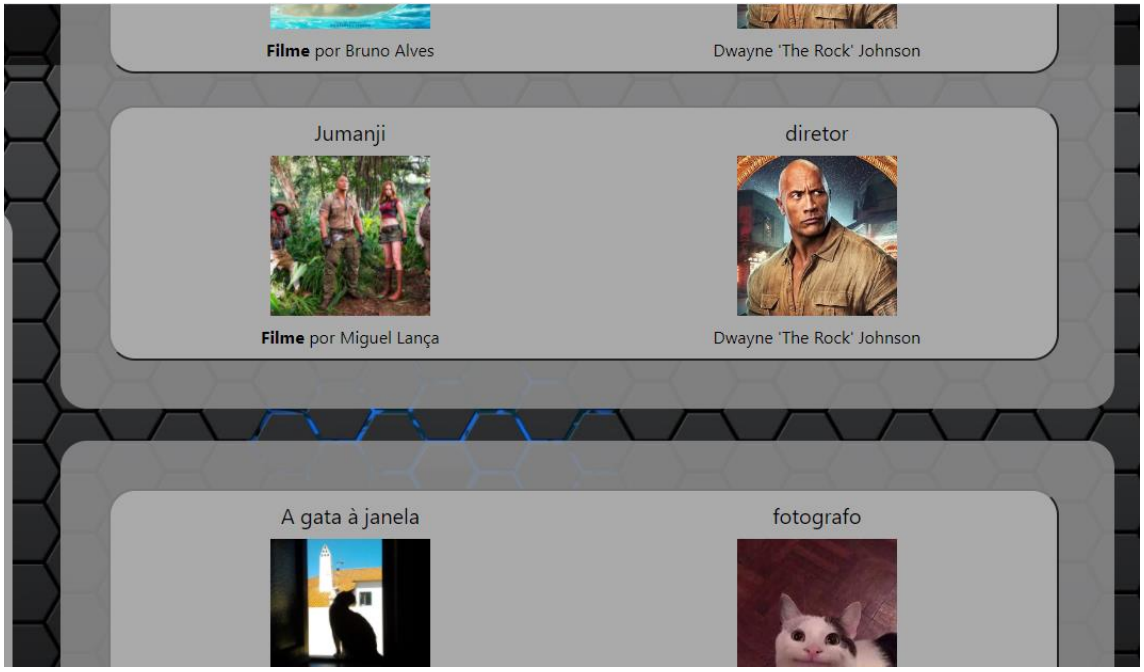
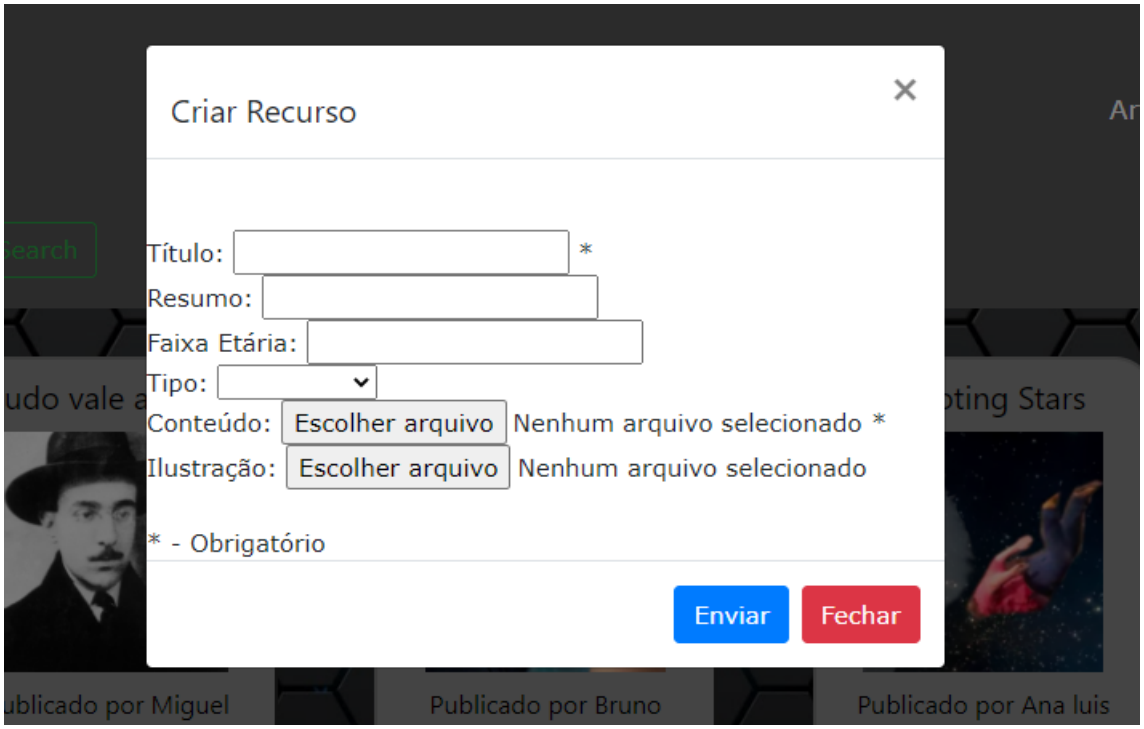


Figura 5 - Método select criado para listar os recursos mostrando o tipo de recurso e o papel da pessoa

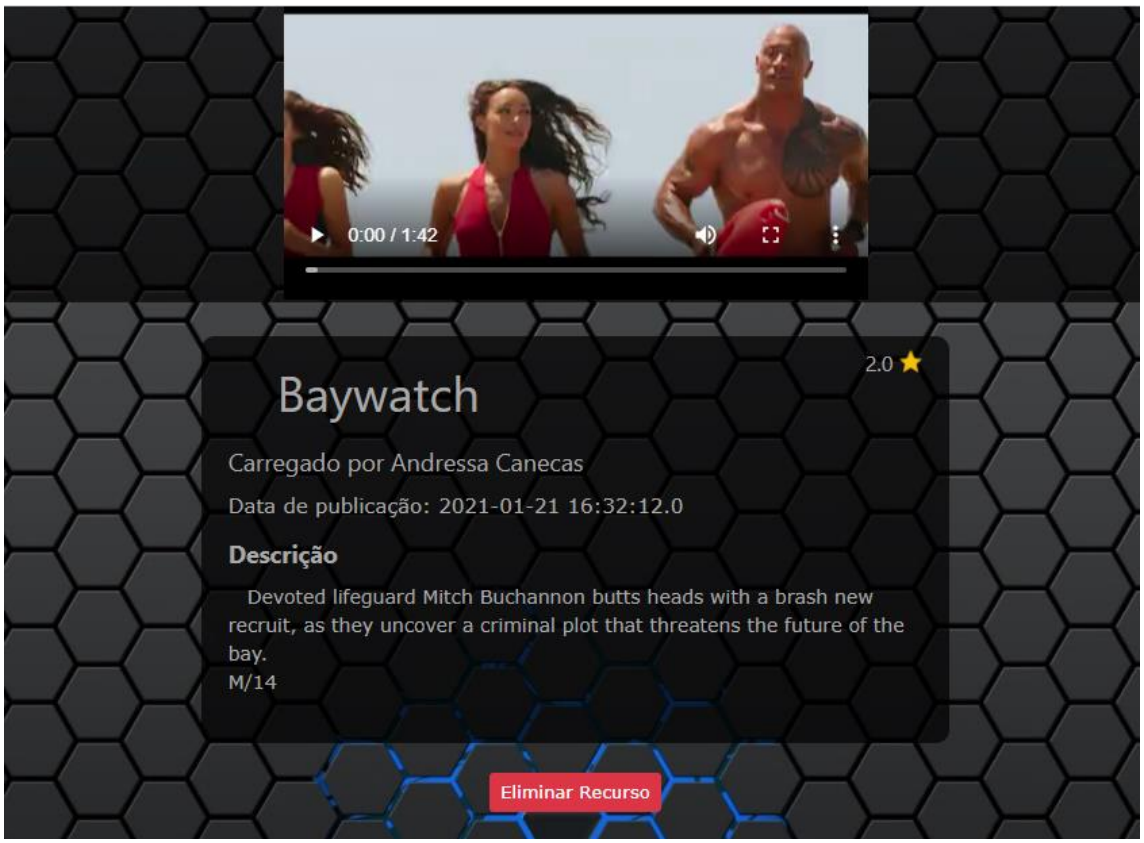
Tipo 2:

1 – **Criar/carregar recursos**



2 – **Remover/descarregar um recurso criado**

O criador do recurso tem um botao disponível na pagina do recurso para o poder apagar.



Tipo 3:

1 – **Associar recursos entre si, por exemplo, associar uma música a um filme ou a um poema**

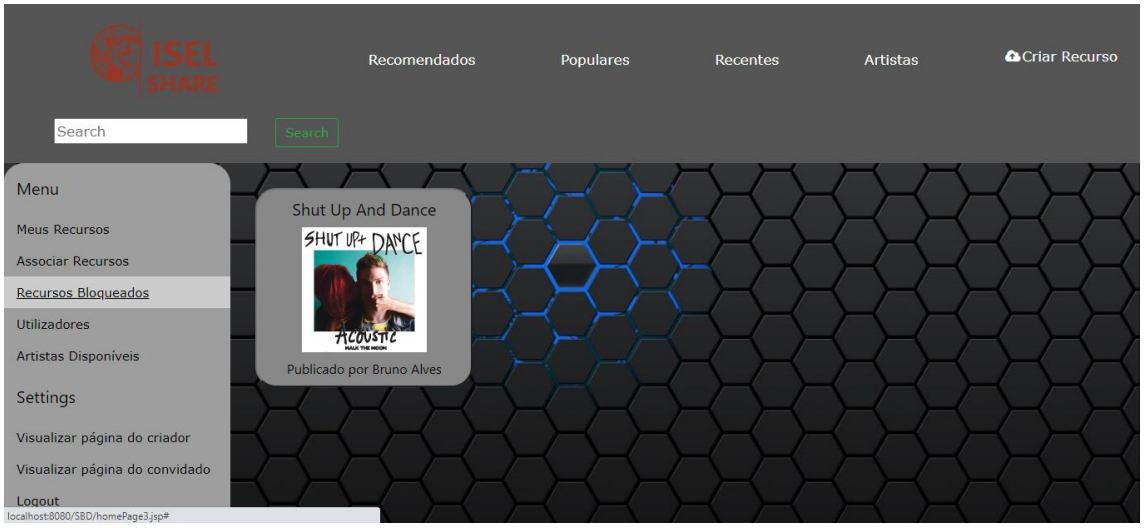
Para associar os recursos entre si, criámos uma classe que irá ser um grupo de recursos onde iremos inserir um id para o grupo, o nome do recurso e por quem foi carregado. Em seguida verificámos se os utilizadores estavam na base de dados e com as credenciais corretas, indo em seguida buscar todos os grupos com recursos e ordenar os mesmos através do id criado.

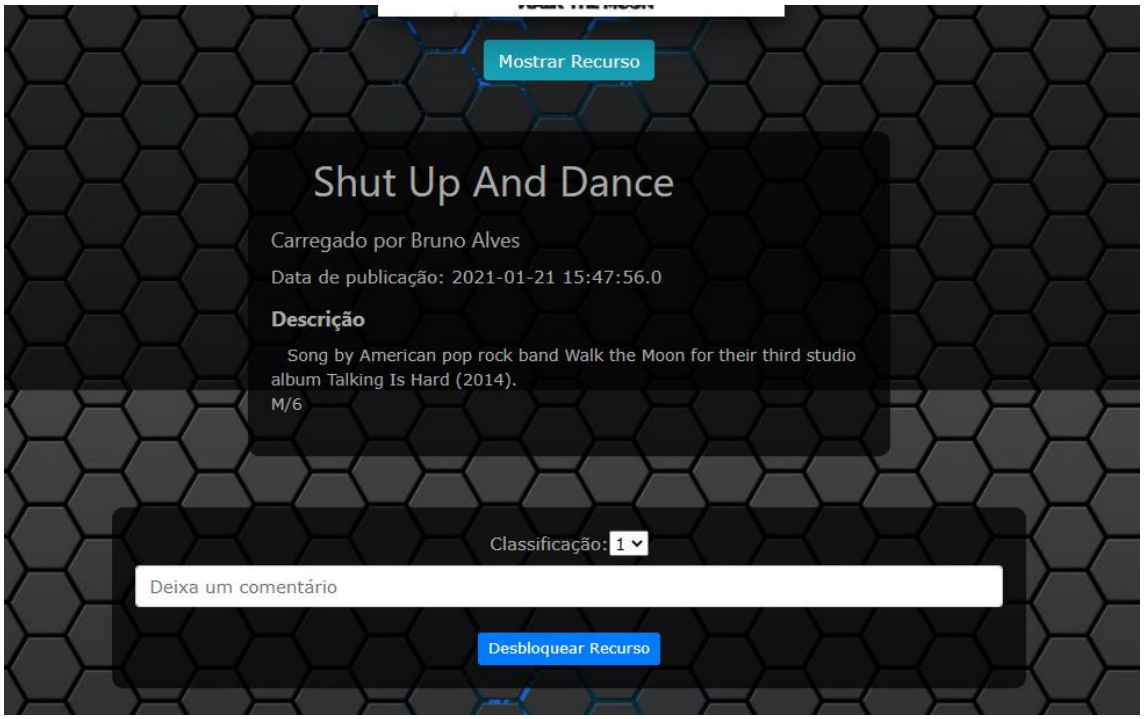
```
String selectAllGrupoRecursoes = "Select * from grupo_recurso order by idGrupo;";
```

Figura 6 - Método select criado para associar recursos entre si

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

O administrador tem um botao disponível na pagina do recurso para o poder bloquear ou desbloquear, e nesta ultima, obrigatoriamente dar uma classificacao.





3 – **Bloquear ou desbloquear um determinado utilizador**

Para bloquear ou desbloquear um utilizador necessitamos de voltar a fazer a verificação dos utilizadores, depois criamos uma classe Tipo_Utilizador que possuí o nome dos utilizadores e qual o seu tipo. Em seguida vamos visualizar todos os tipos de utilizadores pelo seu nome e em seguida definimos se estão ou não bloqueado.

```
String selectTipoUtilizadorByName = "Select * from tipo_utilizador where nomeUtilizador = ?;";
```

Figura 7 - Método select para verificar todos os tipos de utilizadores

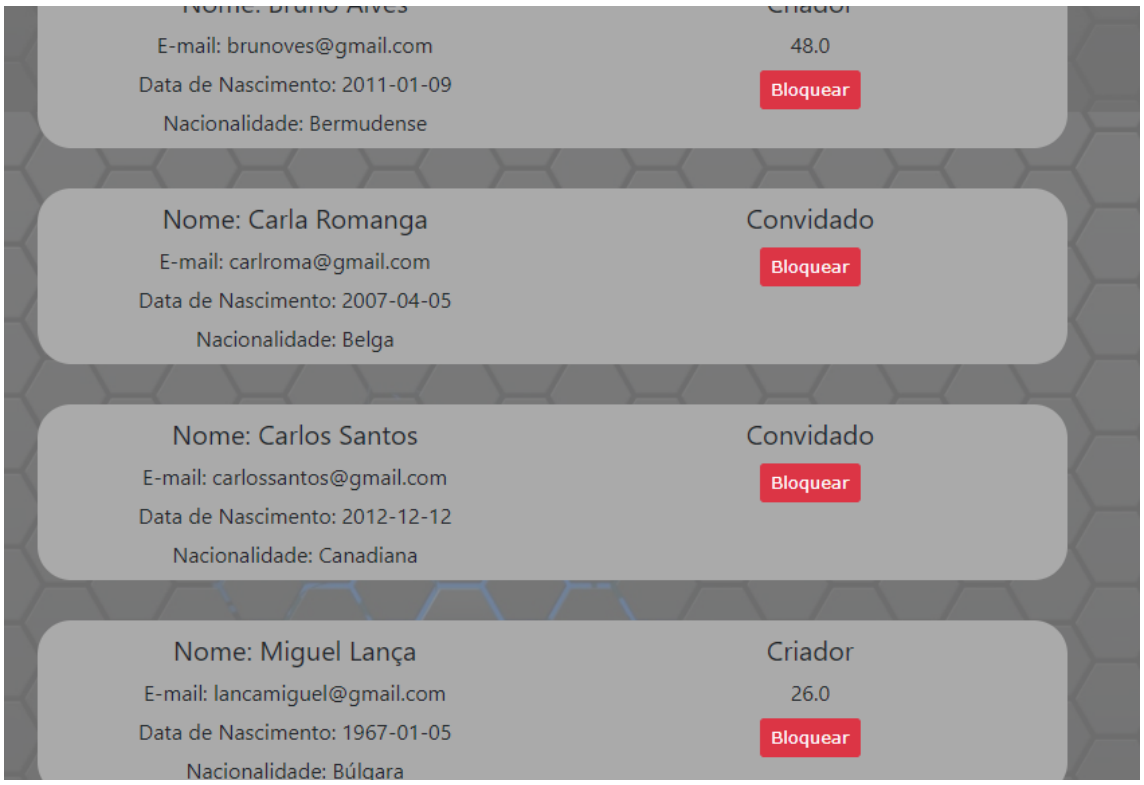
```
String selectConvidadoByNomeUtilizador = "Update convidado set bloqueado = ? where nomeUtilizador = ? ";
```

Figura 8 - Método select para definir se o convidado está ou não bloqueado

```
String selectCriadorByNomeUtilizador = "Update criador set bloqueado = ? where nomeUtilizador = ? ";
```

Figura 9 - Método select para definir se o criador está ou não bloqueado

4 – Listar todos os atributos dos utilizadores ordenados por reputação e com a quantidade de recursos criados



5 – Apresentar todos os atributos de pessoas com os respetivos papéis e quantidade de recursos associados

Para fazer-mos este requisito utilizámos um método que mostra todos os atributos que os utilizadores possuem e em seguida criámos um método para verificar a quantidade de recursos associados por cada utilizador.

```
String selectAllUtilizador = "SELECT "
        + "     u.nomeUtilizador, u.email, u.dataNascimento, n.nacionalidade "
        + "FROM "
        + "     utilizador as u "
        + "INNER JOIN nacionalidade as n ON u.nacionalidade = n.codigo;";
```

Figura 10 - Método select que mostra todos os atributos dos utilizadores

```
for (Utilizador u : users) {

    if (!u.getNomeUtilizador().equals(user)) {

        Tipo_Utilizador tipo = Tipo_UtilizadorDAO.getTipoUtilizadorByName(u.getNomeUtilizador());
        String attrs = "";

        if (tipo.getTipo() == 1) {
            Convidado guest = ConvidadoDAO.getConvidadoByNomeUtilizador(u.getNomeUtilizador());
            attrs = guest.toString();
        }
        else if (tipo.getTipo() == 2) {
            Criador criator = CriadorDAO.getCriadorByNomeUtilizador(u.getNomeUtilizador());
            attrs = criator.toString();
        }
        else {
            Administrador admins = AdministradorDAO.getAdministradorByNomeUtilizador(u.getNomeUtilizador());
            attrs = admins.toString();
        }

        recursosString += u.toString().substring(0, u.toString().length() - 1) + ", ";
        recursosString += attrs.substring(0, attrs.toString().length() - 1);
        recursosString += "|";
        //recursos.put(key, value)
    }
}
```

Figura 11 - Método para verificar a quantidade de recursos associados

6 – **Assumir o papel atribuído a outro utilizador**