

# Documento - LocadoraFilmes

## ▼ ÍNDICE

### ▼ DESCRIÇÃO

- Resumo
- Objetivo
- Funções
- Instruções de importação e uso
  - Instruções de importação do banco de dados no MySQL Workbench
  - Instruções de importação do banco de dados no Eclipse for Java
  - Instruções de uso para MySQL Workbench
  - Instruções de uso para Eclipse for Java

### • MODELO CONCEITUAL

### • MODELO LÓGICO

### • CÓDIGO BANCO DE DADOS SQL (MODELO FÍSICO)

### ▼ CÓDIGO BANCO DE DADOS JAVA

- AluguelDAO
  - AluguelDAO
- Cadastrar
  - CadastrandoClientes
  - CadastrandoFilmes
  - CadastrandoAluguel
- ClassePrincipal
  - PaginaInicial
- ClientesDAO
  - ClientesDAO
- Conexao
  - Conexao
- Deletando
  - DeletandoAluguel
  - DeletandoClientes
  - DeletandoAluguel
- Editando
  - EditandoAluguel
  - EditandoClientes
  - EditandoFilmes
- FilmesDAO
  - FilmesDAO

- Modelos
  - Aluguel
  - Clientes
  - Filme
- Mostrar
  - MostrarAluguel
  - MostrarClientes
  - MostrarFilme
- FUNCIONAMENTO CÓDIGO SQL (MySQL Workbench).
- ▼ FUNCIONAMENTO CÓDIGO JAVA (Eclipse for Java).
  - AluguelDAO
    - AluguelDAO
  - Cadastrar
    - CadastrandoClientes
    - CadastrandoFilmes
    - CadastrarAluguel
    - ClassePrincipal
      - PaginaInicial
  - ClientesDAO
    - ClientesDAO
  - Conexao
    - Conexao
  - Deletando
    - DeletandoClientes
    - DeletandoFilmes
    - DeletandoAluguel
  - Editando
    - EditandoClientes
    - EditandoFilmes
    - EditandoAluguel
  - FilmesDAO
    - FilmesDAO
  - Modelos
    - Clientes
    - Filmes
    - Aluguel
  - Mostrar
    - MostrarAluguel
    - MostrarClientes
    - MostrarFilmes

## ▼ DESCRIÇÃO

### ▼ Resumo

Trata-se de um sistema de banco de dados de uma locadora de filmes. O sistema permite a inclusão, edição e exclusão nas tabelas - assim como em suas colunas - cliente, filme e aluguel, tanto pela execução do código MySQL como pela execução do código Java. Além da visualização completa das tabelas em questão

- Cliente possui as colunas: idcliente, nomecliente, logradouro, numlogradouro, bairro, cidade, estado;
  - idcliente é auto incrementado pelo sistema
- Filme possui as colunas: idfilme, nomefilme, anofilme, genero;
  - idfilme é auto incrementado pelo sistema
- Aluguel possui as colunas: idaluguel, dataaluguel, idcliente, idfilme.
  - idaluguel é auto incrementado pelo sistema

Todas as colunas de todas as tabelas estão como "not null", ou seja, precisam ser preenchidas. Não podem estar vazias.

Todas as colunas com nome "id..." são o id da tabela em questão e são auto incrementados pelo sistema, ou seja, não necessitam ser preenchidos manualmente.

A coluna dataaluguel está como "default current\_timestamp", de modo que será preenchida automaticamente pelo sistema com a data e o horário da inserção. Ou seja, não necessita ser preenchida manualmente.

### ▼ Objetivo

Funcionar como um sistema para gerenciamento de uma locadora de filmes, permitindo a inclusão, edição e exclusão nas tabelas e suas colunas (cliente, filme, aluguel), além da visualização completa dessas, seja pelo MySQL Workbench ou pelo Eclipse for Java.

### ▼ Funções

- Adição (inserção) / edição / exclusão nas tabelas de clientes, filmes e aluguéis;
- Visualização das tabelas de clientes, filmes e aluguéis;
- Tanto pela execução do código SQL como pela execução do código Java.

#### ▼ Adicionar cliente

- Adicionar nomecliente
- Adicionar logradouro
- Adicionar numlogradouro
- Adicionar bairro
- Adicionar cidade
- Adicionar estado

#### ▼ Adicionar filme

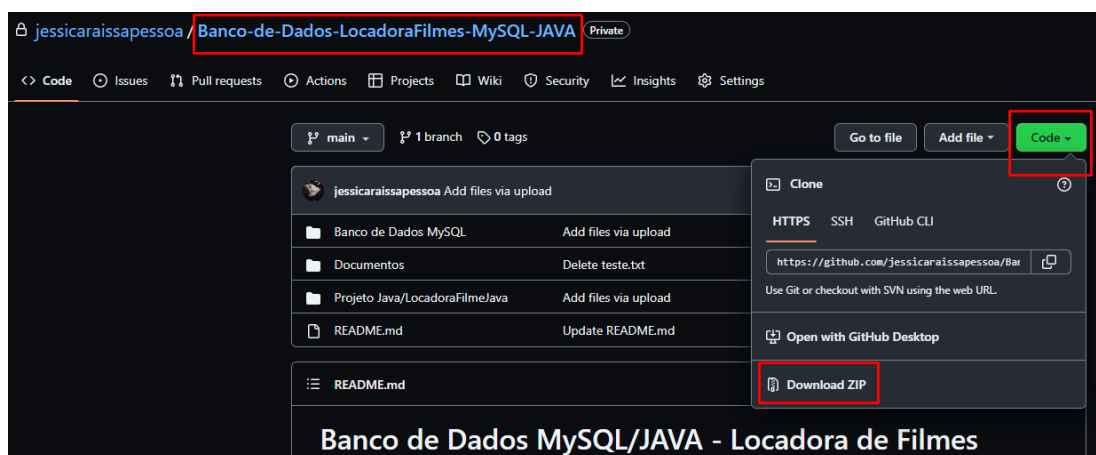
- Adicionar nomefilme
- Adicionar anofilme
- Adicionar generofilme

#### ▼ Adicionar aluguel

- Adicionar dataaluguel
  - Adicionar idcliente
  - Adicionar idfilme
- ▼ Editar cliente
- Editar nomecliente
  - Editar logradouro
  - Editar numlogradouro
  - Editar bairro
  - Editar cidade
  - Editar estado
- ▼ Editar filme
- Editar nomefilme
  - Editar anofilme
  - Editar generofilme
- ▼ Editar aluguel
- Editar dataaluguel
  - Editar idcliente
  - Editar idfilme
- Visualizar tabela de clientes
  - Visualizar tabela de filmes
  - Visualizar tabela de alugueis
  - Excluir cliente
  - Excluir filme
  - Excluir aluguel

#### ▼ Instruções de importação e uso

- Baixe os arquivos do repositório

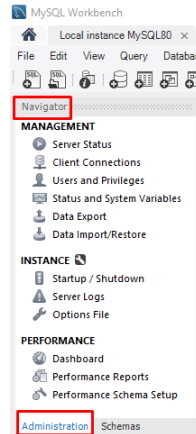


- No arquivo zipado baixado, clique com o botão direito e selecione “Extrair aqui” no menu de opções
- Para o banco de dados SQL siga os passos de [Instruções de importação do banco de dados no MySQL Workbench](#)

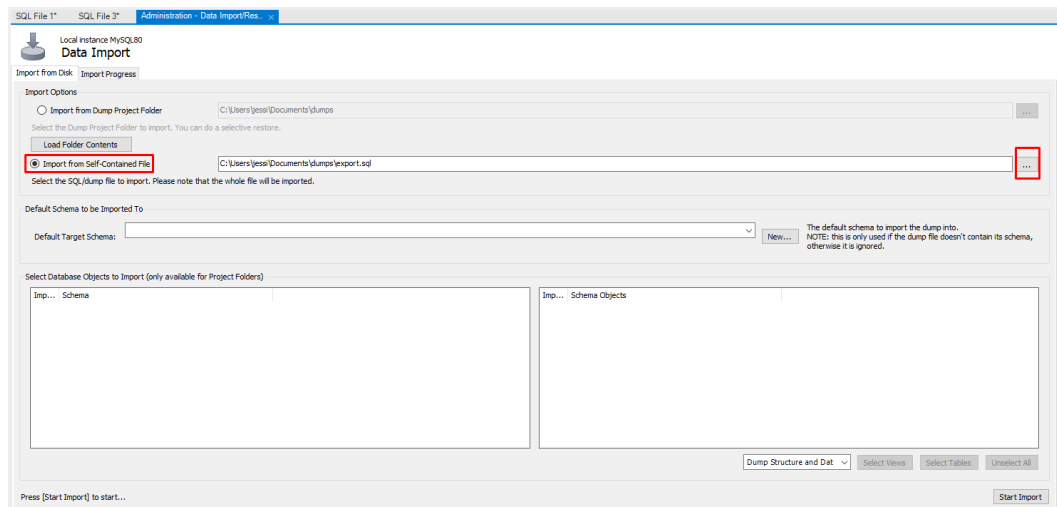
- Para o banco de dados SQL siga os passos de Instruções de importação do banco de dados no Eclipse for Java

#### ▼ Instruções de importação do banco de dados no MySQL Workbench

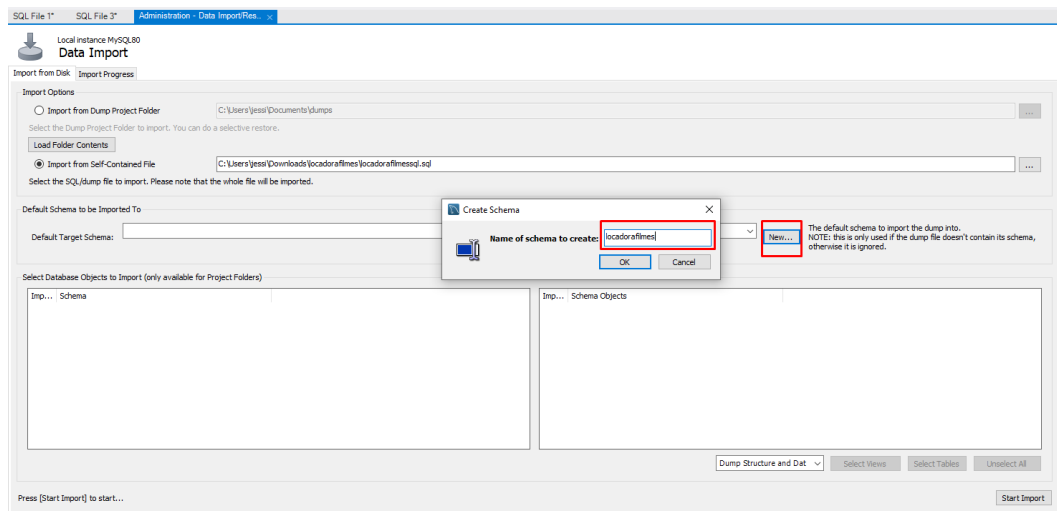
- Abra o MySQL Workbench e conecte em sua instância local
- No “Navegador”, vá em “Administration”



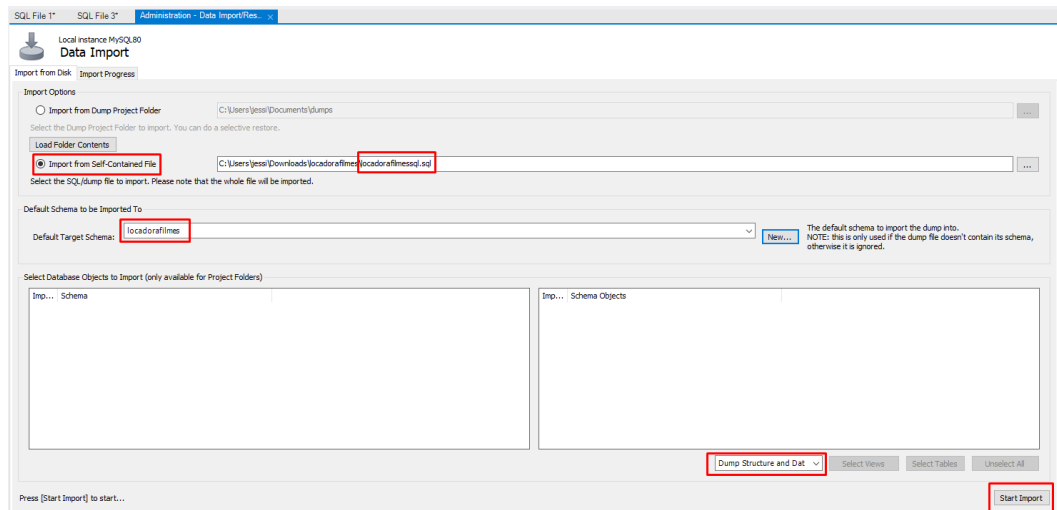
- Clique em “Data Import/Restore”
- Marque “Import from Self-Contained File” e clique em “...”



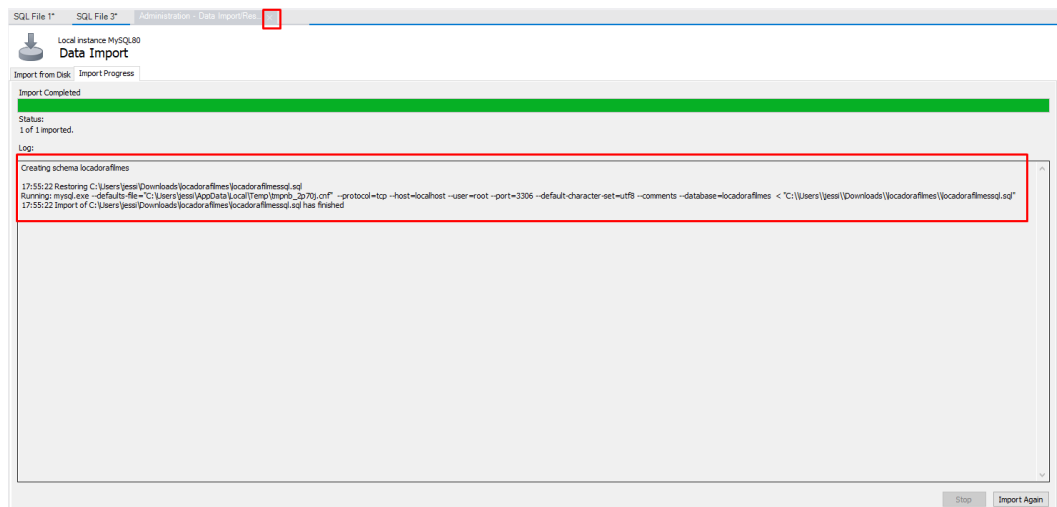
- Procure o arquivo “locadorafilmessql” anteriormente baixado pelo GitHub (geralmente vai para a pasta Downloads do computador)
- Selecione o arquivo e clique em abrir
- Clique para criar o banco de dados que receberá o arquivo em “New” e nomeie o mesmo como “locadorafilmes”



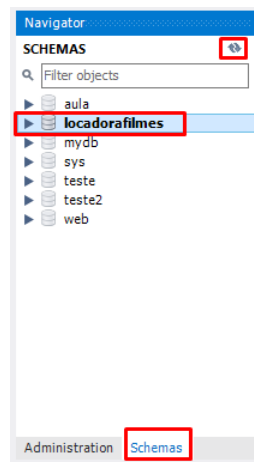
- Confirme se está tudo certo, marque “Dump Struture and Data” e clique em “Start Import



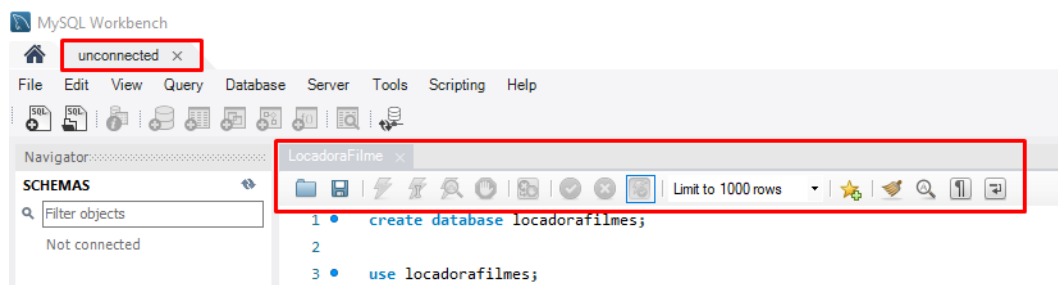
- Confira se deu certo a importação. Caso sim, clique no X para fechar o processo de importação e siga para o próximo passo. Caso não, refaça os passos anteriores



- Em “Navigator”, volte para “Schemas”, atualize os schemas (clcando no ícone com as setinhas) até aparecer o banco de dados “locadoraafilmes”



- Clique para abrir o arquivo LocadoraFilme anteriormente baixado do GitHub
- Ao abrir no MySQL Workbench, o código que você pode utilizar para inserir, editar, excluir dados do banco de dados pelo MySQL vai estar como desconectado



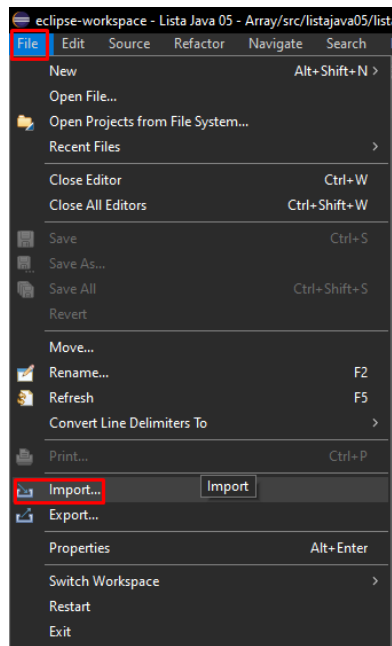
- Para conectar, volte para a home do MySQL Workbench e conecte na instância local novamente



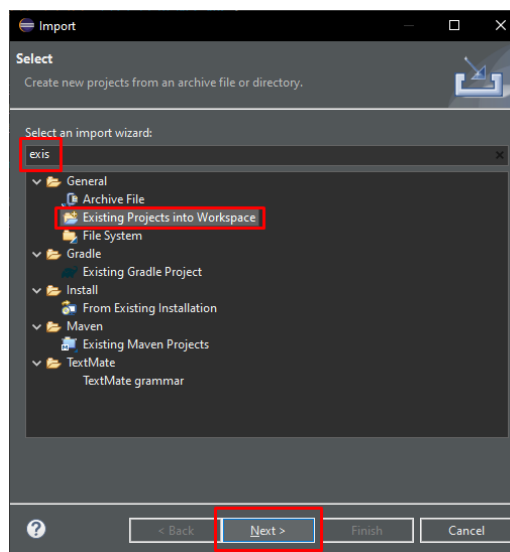
- Siga para [Instruções de uso para MySQL Workbench](#)

#### ▼ Instruções de importação do banco de dados no Eclipse for Java

- Abra o Eclipse, clique em "File" e depois em "Import..."

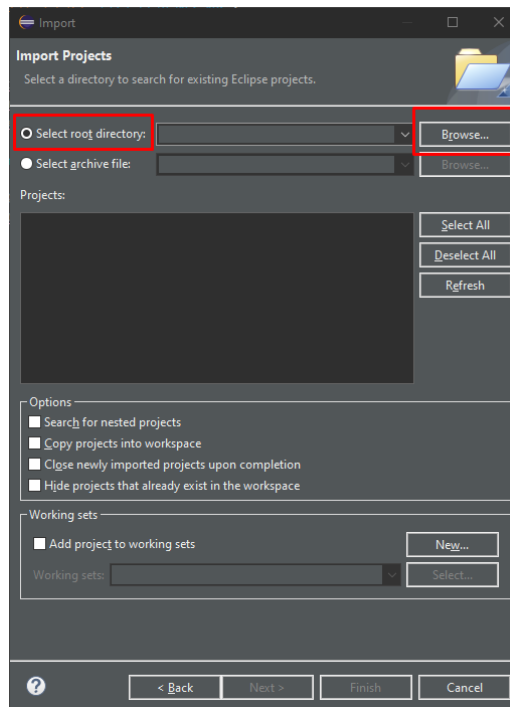


- Aparecerá uma janela pop-up com algumas formas de importação. Para facilitar a localização, na caixa de texto, digite “exis” para que apareça a opção “Existing Projects into Workspace”

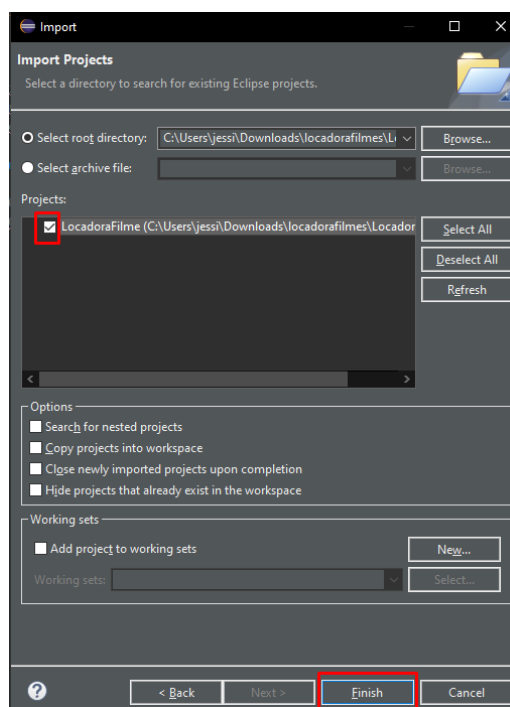


- Selecione a opção “Existing Projects into Workspace” e clique em Next
- Aparecerá a segunda parte do pop-up de importação. Marque “Select root directory” e clique em “Browse” para localizar no computador o arquivo Java anteriormente baixado do GitHub

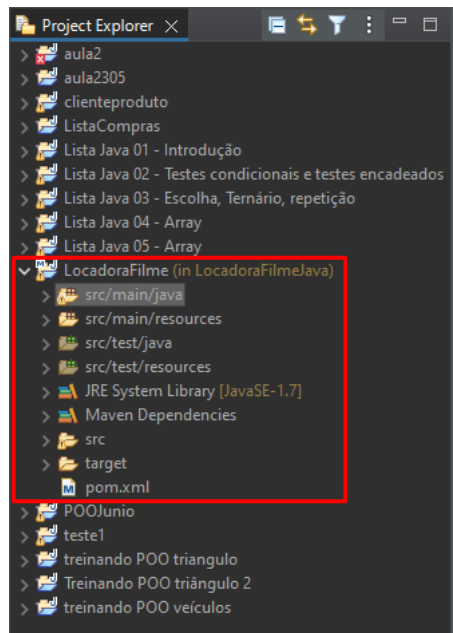




- Localize a pasta “LocadoraFilmeJava”, selecione-a e marque “Selecionar pasta”
- Aparecerá o Projeto Java em questão. Confirma que ele esteja marcado e clique em “Finish”



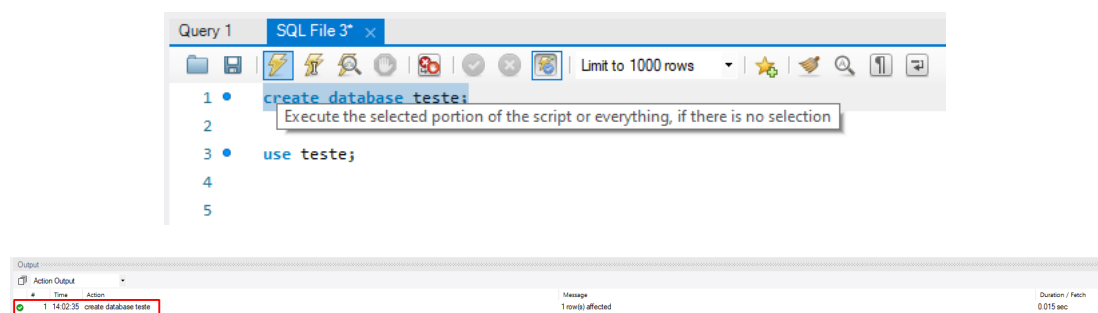
- O projeto deverá aparecer em seu Package Explorer



- Siga para [Instruções de uso para Eclipse for Java](#)
- Você pode inserir, editar, deletar e exibir as tabelas e seus dados tanto pela ferramenta de gerenciamento de banco de dados (no caso, foi utilizado e será esclarecido acerca da ferramenta MySQL Workbench) como pela IDE Java (no caso, foi utilizado e será esclarecido acerca da IDE Eclipse)
- Caso decida fazê-lo pela ferramenta de gerenciamento de banco de dados, siga as instruções de [Instruções de uso para MySQL Workbench](#)
- Caso decida fazê-lo pela IDE Java, siga as instruções de [Instruções de uso para Eclipse for Java](#)

#### ▼ Instruções de uso para MySQL Workbench

O funcionamento base do MySQL Workbench se dá da seguinte maneira: você deverá selecionar todo o texto do bloco em questão e clicar no ícone correspondente à execução. Assim será executada a ação no banco de dados, aparecendo uma mensagem de feedback, que a ação foi executada com sucesso, conforme as imagens do exemplo a seguir ilustram:



Com esse conhecimento em mente e com o arquivo SQL “LocadoraFilme” anteriormente baixado do GitHub e importado, siga as instruções seguinte, as quais dizem respeito ao banco de dados em questão.

OBSERVAÇÃO: Caso tenha vindo de [Instruções de importação do banco de dados no MySQL Workbench](#):

- Faça apenas a execução da linha do “use” descrita no passo [Selecione as linhas em questão e clique para executar a criação da database locadorafilmes e para passar a utilizar essa database \(locadorafilmes\)](#).

- e siga para Selecione a linha em questão e clique para executar a visualização de quais tabelas pertencem à database locadorafilmes. Esse comando pode ser executado todas as vezes que desejar ver as tabelas da database

▼ Selecione as linhas em questão e clique para executar a criação da database locadorafilmes e para passar a utilizar essa database (locadorafilmes)

```
1 • create database locadorafilmes;
2
3 • use locadorafilmes;
```

▼ Selecione as linhas em questão e clique para executar a criação das tabelas cliente, filme e aluguel

```
5 • create table cliente (
6     idcliente int primary key auto_increment,
7     nomecliente varchar(100) not null,
8     logradouro varchar(100) not null,
9     numlogradouro varchar(100) not null,
10    bairro varchar(100) not null,
11    cidade varchar(100) not null,
12    estado varchar(100) not null
13 );
14
15 • create table filme (
16     idfilme int primary key auto_increment,
17     nomefilme varchar(100) not null,
18     anofilme int not null,
19     genero varchar(100) not null
20 );
21
22 • create table aluguel (
23     idaluguel int primary key auto_increment,
24     dataaluguel timestamp default current_timestamp,
25     idcliente int not null,
26     idfilme int not null,
27     foreign key (idcliente) references cliente(idcliente),
28     foreign key (idfilme) references filme(idfilme)
29 );
```

▼ Selecione a linha em questão e clique para executar a visualização de quais tabelas pertencem à database locadorafilmes. Esse comando pode ser executado todas as vezes que desejar ver as tabelas da database

```
32 • show tables;
```

▼ Selecione a linha em questão - linha 34 para cliente, linha 35 para filme, linha 36 para aluguel - e clique para executar, exibindo a tabela escolhida (de acordo com a linha selecionada) e os registros nessa. Caso não tenha sido feito nenhum registro na tabela que decidiu exibir, não aparecerá nenhum registro. Esse comando pode ser executado todas as vezes que desejar ver uma tabela e seus registros

```
34 • select * from cliente;
35 • select * from filme;
36 • select * from aluguel;
```

▼ Selecione a linha e clique para executar. Ambas exibem as tabelas em questão (cliente) em ordem crescente: linha 38, pelo idcliente; linha 39 pelo nomecliente. Esse comando pode ser executado todas as vezes que desejar ver uma tabela e seus registros, dessa forma

```
38 • select * from cliente order by idcliente asc;
39 • select * from cliente order by nomecliente asc;
```

▼ Selecione a linha e clique para executar. Ambas exibem as tabelas em questão (filme) em ordem crescente: linha 41, pelo idfilme; linha 42 pelo nomefilme. Esse comando pode ser executado todas as vezes que desejar ver uma tabela e seus registros, dessa forma

```
41 • select * from filme order by idfilme asc;
42 • select * from filme order by nomefilme asc;
```

▼ Selecione a linha e clique para executar. Ambas exibem as tabelas em questão (aluguel) em ordem crescente: linha 38, pelo idcliente; linha 39 pelo nomecliente. Esse comando pode ser executado todas as vezes que desejar ver uma tabela e seus registros, dessa forma

```
44 • select * from cliente order by idaluguel asc;  
45 • select * from aluguel order by dataaluguel asc;
```

▼ As inserções, edições e exclusões estão com suas “fórmulas” na forma de comentário. Para fazer uma inserção, edição ou exclusão, você deve copiar a fórmula, colar fora do espaço de comentário (que vai de /\* a \*/) e preencher com os dados, conforme as imagens do exemplo a seguir ilustram:

#### ▼ Exemplo

- Espaço de comentários (vai de /\* a \*/)

```
47  /* Copiar comando desejado e preencher:  
48  
49  insert into cliente (nomecliente, logradouro, numlogradouro, bairro, cidade, estado)  
50  values ("", "", "", "", "", "");  
51  
52  insert into filme (nomefilme, anofilme, genero)  
53  values ("", "", "");  
54  
55  insert into aluguel (idcliente_cliente, idfilme_filme)  
56  values ( , );  
57  
58  
59  UPDATE cliente SET nomecliente = "", logradouro = "", numlogradouro = "", bairro = "", cidade = "", estado = "" WHERE idcliente = ;  
60  UPDATE filme SET nomefilme = "", anofilme = , genero = "" WHERE idfilme = ;  
61  UPDATE aluguel SET dataaluguel = '--', idcliente = , idaluguel = WHERE idaluguel = ;  
62  
63  DELETE FROM cliente WHERE idcliente = ;  
64  DELETE FROM filme WHERE idfilme = ;  
65  DELETE FROM aluguel WHERE idaluguel = ;  
66  
67  */
```

- Desejo fazer a inserção de um aluguel. Logo, copio as linhas correspondentes

```
47  /* Copiar comando desejado e preencher:  
48  
49  insert into cliente (nomecliente, logradouro, numlogradouro, bairro, cidade, estado)  
50  values ("", "", "", "", "", "");  
51  
52  insert into filme (nomefilme, anofilme, genero)  
53  values ("", "", "");  
54  
55  insert into aluguel (idcliente_cliente, idfilme_filme)  
56  values ( , );  
57  
58  
59  UPDATE cliente SET nomecliente = "", logradouro = "", numlogradouro = "", bairro = "", cidade = "", estado = "" WHERE idcliente = ;  
60  UPDATE filme SET nomefilme = "", anofilme = , genero = "" WHERE idfilme = ;  
61  UPDATE aluguel SET dataaluguel = '--', idcliente = , idaluguel = WHERE idaluguel = ;  
62  
63  DELETE FROM cliente WHERE idcliente = ;  
64  DELETE FROM filme WHERE idfilme = ;  
65  DELETE FROM aluguel WHERE idaluguel = ;  
66  
67  */
```

- Colo fora do espaço de comentários, preencho e executo

```
47  /* Copiar comando desejado e preencher:  
48  
49  insert into cliente (nomecliente, logradouro, numlogradouro, bairro, cidade, estado)  
50  values ("", "", "", "", "", "");  
51  
52  insert into filme (nomefilme, anofilme, genero)  
53  values ("", "", "");  
54  
55  insert into aluguel (idcliente_cliente, idfilme_filme)  
56  values ( , );  
57  
58  
59  UPDATE cliente SET nomecliente = "", logradouro = "", numlogradouro = "", bairro = "", cidade = "", estado = "" WHERE idcliente = ;  
60  UPDATE filme SET nomefilme = "", anofilme = , genero = "" WHERE idfilme = ;  
61  UPDATE aluguel SET dataaluguel = '--', idcliente = , idaluguel = WHERE idaluguel = ;  
62  
63  DELETE FROM cliente WHERE idcliente = ;  
64  DELETE FROM filme WHERE idfilme = ;  
65  DELETE FROM aluguel WHERE idaluguel = ;  
66  
67  */  
68  
69  insert into aluguel (idcliente_cliente, idfilme_filme)  
70  values (3,3);
```

Para cada inserção, edição e exclusão, o processo de copiar, colar, preencher e executar a fórmula deve ser repetido.

#### ▼ Inserção

Copia, cola, preenche e executa a fórmula da tabela para a qual deseja fazer uma inserção: linhas 49 e 50 para inserir cliente; linhas 52 e 53 para inserir filme; linhas 55 e 56 para inserir aluguel

```
49 insert into cliente (nomecliente, logradouro, numlogradouro, bairro, cidade, estado)
50 values ("", "", "", "", "", "");
51
52 insert into filme (nomefilme, anofilme, genero)
53 values ("", , "");
54
55 insert into aluguel (idcliente_cliente, idfilme_filme)
56 values ( , );
```

#### ▼ Edição

Copia, cola, preenche e executa a fórmula da tabela para a qual deseja fazer uma edição: linha 59 para editar cliente; linha 60 para editar filme; linhas 61 para editar aluguel

```
59 UPDATE cliente SET nomecliente = "", logradouro = "", numlogradouro = "", bairro = "", cidade = "", estado = "" WHERE idcliente = ;
60 UPDATE filme SET nomefilme = "", anofilme = , genero = "" WHERE idfilme = ;
61 UPDATE aluguel SET dataaluguel = "--", idcliente = , idaluguel = WHERE idaluguel = ;
```

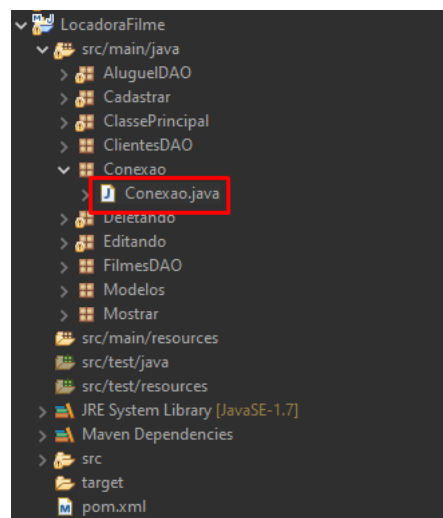
#### ▼ Exclusão

Copia, cola, preenche e executa a fórmula da tabela para a qual deseja fazer uma exclusão: linha 63 para excluir aluguel; linhas 64 para excluir filme; linha 65 para excluir aluguel

```
63 DELETE FROM cliente WHERE idcliente = ;
64 DELETE FROM filme WHERE idfilme = ;
65 DELETE FROM aluguel WHERE idaluguel = ;
```

#### ▼ Instruções de uso para Eclipse for Java

- Para o uso, abra (selecionando e clicando) a classe “Conexao” pelo Package Explorer do Eclipse, cujo caminho de localização é LocadoraFilme > src > Conexao



- Preencha esse trecho com os dados da sua conexão do banco de dados

```

1 package Conexao;
2 import java.sql.Connection;
3
4
5 public class Conexao {
6
7
8     public static Connection conectaDB() {
9         Connection conexao = null;
10        try {
11            Class.forName("com.mysql.cj.jdbc.Driver");
12            try {
13                conexao = DriverManager.getConnection("jdbc:mysql://localhost:3306/locadorafilmes", "root", "");
14            }
15        }
16    }
17 }

```

- Execute a classe clicando no botão de execução ou pressionando F11

```

1 package Conexao;
2 import java.sql.Connection;
3
4
5 public class Conexao {
6
7
8     public static Connection conectaDB() {
9         Connection conexao = null;
10        try {
11            Class.forName("com.mysql.cj.jdbc.Driver");
12            try {
13                conexao = DriverManager.getConnection("jdbc:mysql://localhost:3306/locadorafilmes", "root", "");
14            }
15        }
16    }
17 }

```

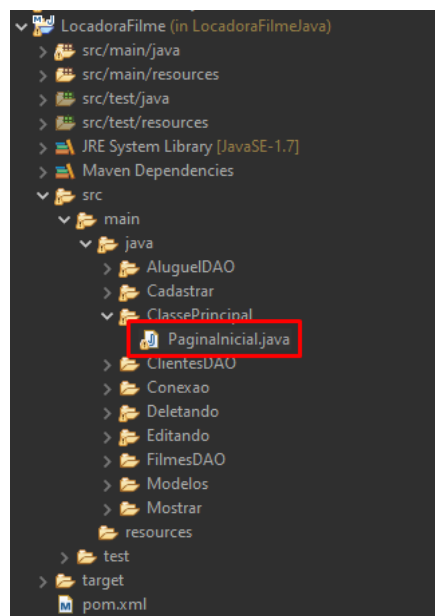
- Aparecerá uma mensagem no terminal confirmando que a conexão entre o Java e o banco de dados foi executada com sucesso

```

terminated> Conexao [Java Application] C:\Program Files\Java\jdk-11.0.13\bin\javaw.exe (19 de jun de 2022 21:03:22 - 21:03:23)
Foi conectado com sucesso!

```

- Abra (selecionando e clicando) a classe "PaginaInicial" pelo Package Explorer do Eclipse, cujo caminho de localização é LocadoraFilme > src > ClassePrincipal



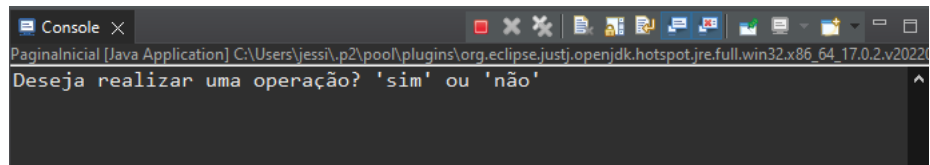
- Execute a classe clicando no botão de execução ou pressionando F11

```

1 package ClassePrincipal;
2 import java.util.Scanner;
3
4
5 public class PaginaInicial {
6
7
8     public static void main(String[] args) {
9
10        Scanner sc = new Scanner(System.in);
11    }
12 }

```

- O terminal funcionará como uma interface por meio da qual poderão ser realizadas as operações do banco de dados do sistema

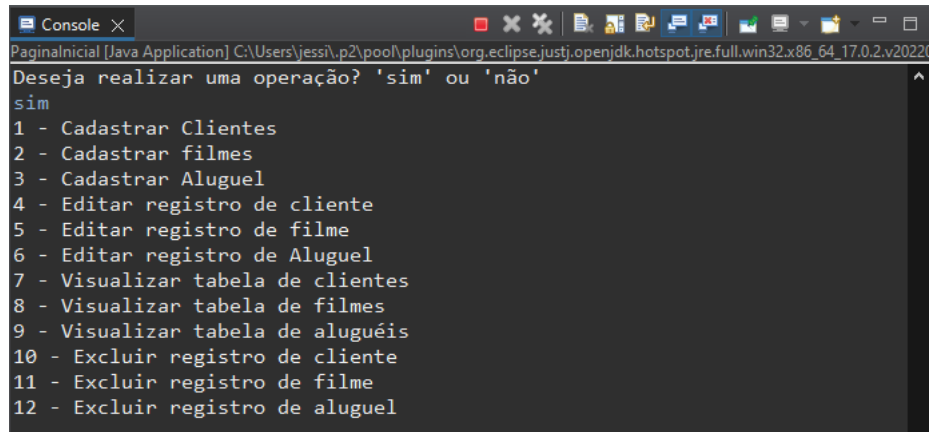


```

Console X
PaginaInicial [Java Application] C:\Users\jessi\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v2022...
Deseja realizar uma operação? 'sim' ou 'não'

```

- Ao digitar "sim" e confirmar pressionando enter, surgirão as opções de operações



```

Console X
PaginaInicial [Java Application] C:\Users\jessi\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v2022...
Deseja realizar uma operação? 'sim' ou 'não'
sim
1 - Cadastrar Clientes
2 - Cadastrar filmes
3 - Cadastrar Aluguel
4 - Editar registro de cliente
5 - Editar registro de filme
6 - Editar registro de Aluguel
7 - Visualizar tabela de clientes
8 - Visualizar tabela de filmes
9 - Visualizar tabela de alugueis
10 - Excluir registro de cliente
11 - Excluir registro de filme
12 - Excluir registro de aluguel

```

- Para executar uma operação, bata digitar o número correspondente à operação desejada e pressionar enter. Começará a ser executado o formulário da operação
- Ao final, ele exibe uma mensagem de confirmação do sucesso da operação e reinicia para mais uma operação a ser escolhida

```
Console X
PaginaInicial [Java Application] C:\Program Files\Java\jdk-11.0.13\
Deseja realizar uma operação? 'sim' ou 'não'
sim
1 - Cadastrar Clientes
2 - Cadastrar filmes
3 - Cadastrar Aluguel
4 - Editar registro de cliente
5 - Editar registro de filme
6 - Editar registro de Aluguel
7 - Visualizar tabela de clientes
8 - Visualizar tabela de filmes
9 - Visualizar tabela de alugueis
10 - Excluir registro de cliente
11 - Excluir registro de filme
12 - Excluir registro de aluguel
2
Informe qual filme você deseja adicionar:
Homem de Ferro
Agora informe o ano em que o filme foi lançado:
2008
Para finalizar, informe o gênero do filme em que
Aventura
Filme : Homem de Ferro Adicionado com sucesso!
1 - Cadastrar Clientes
2 - Cadastrar filmes
3 - Cadastrar Aluguel
4 - Editar registro de cliente
5 - Editar registro de filme
6 - Editar registro de Aluguel
7 - Visualizar tabela de clientes
8 - Visualizar tabela de filmes
9 - Visualizar tabela de alugueis
10 - Excluir registro de cliente
11 - Excluir registro de filme
12 - Excluir registro de aluguel
```

- Você pode confirma o sucesso do registro pelo MySQL Workbench

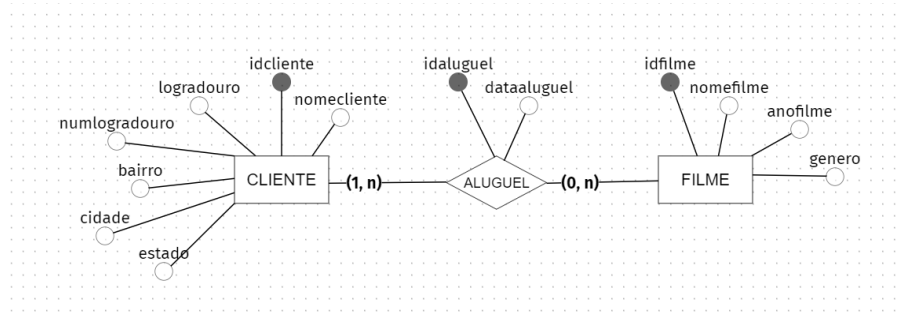
```
ProjetoFacul X
foreign key (idfilme) references filme(idfilme)
);
show tables;
select * from cliente;
35 select * from filme;
select * from aluguel;
select * from cliente order by idcliente asc;
select * from cliente order by nomecliente asc;
select * from filme order by idfilme asc;
select * from filme order by nomefilme asc;
select * from cliente order by idaluguel asc;
select * from aluguel order by dataaluguel asc;
```

idfilme	nomefilme	anofilme	genero
1	O homem de Ferro	2008	Ação
2	Batman	2022	Ação
3	O Homem de ferro	2008	Ação/Aventura
4	Homem de Ferro	2008	Aventura



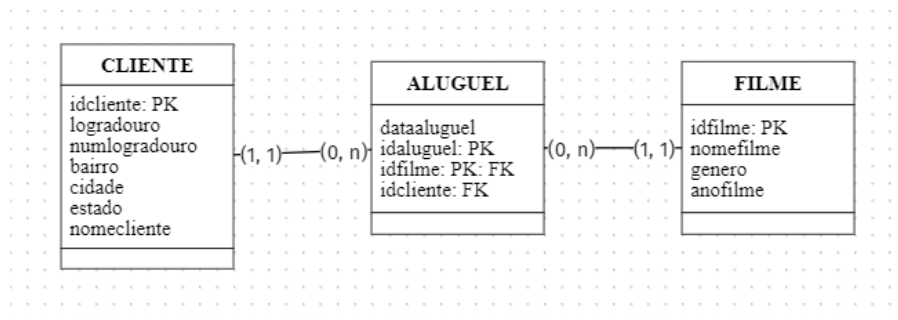
- As operações na base de dados pelo Eclipse Java também podem ser executadas pelas classes respectivas. Para mais informações sobre essas classes e demais, acessar [FUNCIONAMENTO CÓDIGO JAVA \(Eclipse for Java\)](#)

### ▼ MODELO CONCEITUAL



\* Modelo conceitual elaborado no BR Modelo

### ▼ MODELO LÓGICO



\* Modelo lógico elaborado no BR Modelo

### ▼ CÓDIGO BANCO DE DADOS SQL (MODELO FÍSICO)

locadorafilmes

```

create database locadorafilmes;

use locadorafilmes;

create table cliente (
  idcliente int primary key auto_increment,
  nomecliente varchar (100) not null,
  logradouro varchar (100) not null,
  numlogradouro varchar(100) not null,
  bairro varchar (100) not null,
  cidade varchar (100) not null,
  estado varchar (100) not null
);

create table filme (
  idfilme int primary key auto_increment,
  nomefilme varchar (100) not null,
  anofilme int not null,
  genero varchar (100) not null
);

create table aluguel (
  idaluguel int primary key auto_increment,
  dataaluguel timestamp default current_timestamp,
  idcliente int not null,
  idfilme int not null,

```

```

        foreign key (idcliente) references cliente(idcliente),
        foreign key (idfilme) references filme(idfilme)
    );

show tables;

select * from cliente;
select * from filme;
select * from aluguel;

select * from cliente order by idcliente asc;
select * from cliente order by nomecliente asc;

select * from filme order by idfilme asc;
select * from filme order by nomefilme asc;

select * from cliente order by idaluguel asc;
select * from aluguel order by dataaluguel asc;

/* Copiar comando desejado e preencher:

insert into cliente (nomecliente, logradouro, numlogradouro, bairro, cidade, estado)
values ("", "", "", "", "", "");

insert into filme (nomefilme, anofilme, genero)
values ("", , "");

insert into aluguel (idcliente, idfilme)
values ( , );

UPDATE cliente SET nomecliente = "", logradouro = "", numlogradouro = "", bairro = "", cidade = "", estado = "" WHERE idcliente = ;
UPDATE filme SET nomefilme = "", anofilme = , genero = "" WHERE idfilme = ;
UPDATE aluguel SET idcliente = , idaluguel =  WHERE idaluguel = ;

DELETE FROM cliente WHERE idcliente = ;
DELETE FROM filme WHERE idfilme = ;
DELETE FROM aluguel WHERE idaluguel = ;

*/

```

\* Código SQL executado no MySQL Workbench

## ▼ CÓDIGO BANCO DE DADOS JAVA

- Cada toggle list é um package. Dentro desse package, cada toggle list é uma classe do package. Dentro das classes, estão seus respectivos códigos;
- Todos os packages fazem parte do Java Project LocadoraFilme;
- O código foi criado e executado na IDE Eclipse.

## ▼ AluguelDAO

### ▼ AluguelDAO

```

package AluguelDAO;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

import Conexao.Conexao;
import Modelos.Aluguel;

public class AluguelDAO {

    // Conexão
    static Conexao cl = new Conexao();
    static Connection conexao = Conexao.conectaDB();

    public static void main(String[] args) {
        mostrarAluguel();
    }
}

```

```

public static void cadastrarAluguel(Aluguel aluguel) {

    String sql = "INSERT INTO aluguel (idcliente, idfilme) VALUES (?, ?)";
    Connection con = null;
    PreparedStatement ps = null;

    try {

        con = Conexao.conectaDB();

        ps = con.prepareStatement(sql);

        ps.setInt(1, aluguel.getIdCl());
        ps.setInt(2, aluguel.getIdFilm());

        ps.execute();

    } catch (Exception e) {
        e.printStackTrace();
    }

}

public static void mostrarAluguel() {

    ResultSet aluguel1;
    try {
        aluguel1 = conexao.createStatement().executeQuery("SELECT * FROM aluguel");
        while (aluguel1.next()) {
            System.out.println(aluguel1.getInt("idaluguel") + " - " + "ID cliente: " + aluguel1.getInt("idcliente") + ", I
D Filme: " + aluguel1.getInt("idfilme") + ", Data: "
+ aluguel1.getDate("dataaluguel"));
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }

}

public static void editarAluguel(Aluguel aluguel) {

    String sql = "UPDATE aluguel SET idcliente = ?, idfilme = ? WHERE idaluguel = ?";

    Connection con = null;

    PreparedStatement ps = null;

    try {
        con = Conexao.conectaDB();

        ps = con.prepareStatement(sql);

        ps.setInt(1, aluguel.getIdCl());
        ps.setInt(2, aluguel.getIdFilm());

        ps.setInt(3, aluguel.getIdaluguel());

        ps.execute();

    } catch (Exception e) {
        e.printStackTrace();
    }

}

public static void apagarAluguel(int idAluguel) {

    String sql = "DELETE FROM aluguel WHERE idaluguel = ?";

    Connection con = null;
    PreparedStatement ps = null;

    try {

        con = Conexao.conectaDB();

        ps = con.prepareStatement(sql);

        ps.setInt(1, idAluguel);

        ps.execute();

    } catch (Exception e) {
        e.printStackTrace();
    }

}

```

```
}  
  
}
```

## ▼ Cadastrar

### ▼ CadastrandoClientes

```
package Cadastrar;  
  
import java.util.Scanner;  
  
import ClientesDAO.ClientesDAO;  
import Modelos.Clientes;  
  
public class CadastrandoClientes {  
  
    public static void main(String[] args) {  
  
        // CADASTRAR CLIENTES  
        Scanner sc = new Scanner(System.in);  
  
        Clientes cliente = new Clientes();  
  
        System.out.println("Insira o nome do cliente: ");  
        String nome = sc.nextLine();  
        cliente.setnomeCliente(nome);  
  
        System.out.println("Insira o Logradouro do cliente: ");  
        String logradouro = sc.nextLine();  
        cliente.setLogradouro(logradouro);  
  
        System.out.println("Insira o numero do logradouro: ");  
        String NumLogradouro = sc.nextLine();  
        cliente.setNumLogradouro(NumLogradouro);  
  
        System.out.println("Insira o bairro onde o cliente reside: ");  
        String bairro = sc.nextLine();  
        cliente.setBairro(bairro);  
  
        System.out.println("Insira a cidade onde o cliente reside: ");  
        String cidade = sc.nextLine();  
        cliente.setCidade(cidade);  
  
        System.out.println("Insira o estado onde o cliente reside: ");  
        String estado = sc.nextLine();  
        cliente.setEstado(estado);  
  
        ClientesDAO.criarCliente(cliente);  
  
        System.out.println("Cliente " + nome + " Cadastrado com sucesso!\n");  
  
    }  
  
}
```

### ▼ CadastrandoFilmes

```
package Cadastrar;  
  
import java.util.Scanner;  
  
import FilmesDAO.FilmesDAO;  
import Modelos.Filmes;  
  
public class CadastrandoFilmes {  
  
    public static void main(String[] args) {  
  
        Scanner sc = new Scanner(System.in);  
  
        Filmes filme = new Filmes();  
  
        System.out.println("Informe qual filme você deseja adicionar: ");  
        String novofilme = sc.nextLine();  
  
    }  
  
}
```

```

        filme.setNomeFilme(novofilme);

        System.out.println("Agora informe o ano em que o filme foi lançado: ");
        int anolanc = sc.nextInt();
        filme.setAnoFilme(anolanc);

        System.out.println("Para finalizar, informe o gênero do filme em questão: ");
        sc.nextLine();
        String genfilm = sc.nextLine();
        filme.setGenero(genfilm);

        FilmesDAO.AdicionarFilmes(filme);

        System.out.println("Filme : " + novofilme + " Adicionado com sucesso!\n");
    }
}

```

#### ▼ CadastrandoAluguel

```

package Cadastrar;
import java.util.Scanner;

import AluguelDAO.AluguelDAO;
import ClientesDAO.ClientesDAO;
import FilmesDAO.FilmesDAO;
import Modelos.Aluguel;

public class CadastrarAluguel {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        Aluguel alg = new Aluguel();

        ClientesDAO.mostrarClientes();

        System.out.println("Informe o ID do cliente que deseja realizar o aluguel: ");
        int idcl = sc.nextInt();
        alg.setIdCl(idcl);

        FilmesDAO.mostrarFilmes();
        System.out.println("Agora informe o ID do filme que ele deseja Alugar:\n ");
        int idfm = sc.nextInt();
        alg.setIdFilm(idfm);

        AluguelDAO.cadastrarAluguel(alg);

        System.out.println("Aluguel realizado com sucesso!\n");
    }
}

```

#### ▼ ClassePrincipal

##### ▼ Paginalnicial

```

package ClassePrincipal;
import java.util.Scanner;

import Cadastrar.CadastrandoClientes;
import Cadastrar.CadastrandoFilmes;
import Cadastrar.CadastrarAluguel;
import Deletando.DeletandoAluguel;
import Deletando.DeletandoClientes;
import Deletando.DeletandoFilmes;
import Editando.EditandoAluguel;
import Editando.EditandoClientes;
import Editando.EditandoFilmes;
import Mostrar.MostrarAluguel;
import Mostrar.MostrarClientes;
import Mostrar.MostrarFilmes;

```

```

public class PaginaInicial {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.println("Deseja realizar uma operação? 'sim' ou 'não'");
        String opt = sc.nextLine();
        while (opt.equalsIgnoreCase("sim")) {

            System.out.println(
                "1 - Cadastrar Clientes \n2 - Cadastrar filmes \n3 - Cadastrar Aluguel \n4 - Editar registro de cliente "
                + "\n5 - Editar registro de filme \n6 - Editar registro de Aluguel \n7 - Visualizar tabela de clientes
                \n8 - Visualizar tabela de filmes "
                + "\n9 - Visualizar tabela de alugueis "
                + "\n10 - Excluir registro de cliente \n11 - Excluir registro de filme \n12 - Excluir registro de alugue
                l");

            int opcao = sc.nextInt();

            if (opcao == 1) {
                CadastrarClientes.main(args);
            } else if (opcao == 2) {
                CadastrarFilmes.main(args);
            } else if (opcao == 3) {
                CadastrarAluguel.main(args);
            } else if (opcao == 4) {
                EditandoClientes.main(args);
            } else if (opcao == 5) {
                EditandoFilmes.main(args);
            } else if (opcao == 6) {
                EditandoAluguel.main(args);
            } else if (opcao == 7) {
                MostrarClientes.main(args);
            } else if (opcao == 8) {
                MostrarFilmes.main(args);
            } else if (opcao == 9) {
                MostrarAluguel.main(args);
            } else if (opcao == 10) {
                DeletandoClientes.main(args);
            } else if (opcao == 11) {
                DeletandoFilmes.main(args);
            } else if (opcao == 12) {
                DeletandoAluguel.main(args);
            } else {
                System.out.println("Opção inválida!!!");
            }

        }
        System.out.println("Até a próxima pequeno gafanhoto!!!");
    }
}

```

## ▼ ClientesDAO

### ▼ ClientesDAO

```

package ClientesDAO;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

import Conexao.Conexao;
import Modelos.Clientes;

public class ClientesDAO {

    public static void main(String[] args) {
        mostrarClientes();
    }

    // Conexão
    static Conexao cl = new Conexao();
    static Connection conexao = Conexao.conectaDB();

    // Métodos para modificar no banco
    //      V      V      V      V

```

```

public static void criarCliente(Clientes cliente) {
    String sql = "INSERT INTO cliente (nomecliente, logradouro, numlogradouro, bairro, cidade, estado) VALUES (?, ?, ?, ?, ?, ?)";
    Connection conexao = null;
    PreparedStatement ps = null;

    try {
        // Criando conexão com o banco de dados.
        conexao = Conexao.conectaDB();

        ps = conexao.prepareStatement(sql);

        // Adicionando valores para query
        ps.setString(1, cliente.getNomeCliente());
        ps.setString(2, cliente.getLogradouro());
        ps.setString(3, cliente.getNumLogradouro());
        ps.setString(4, cliente.getBairro());
        ps.setString(5, cliente.getCidade());
        ps.setString(6, cliente.getEstado());

        // Executando
        ps.execute();
    }
    catch (Exception e) {
        e.printStackTrace();
    }
}

public static void mostrarClientes() {
    ResultSet cliente1;
    try {
        cliente1 = conexao.createStatement().executeQuery("SELECT * FROM cliente");
        while (cliente1.next()) {
            System.out.println(cliente1.getInt("idcliente") + " " + cliente1.getString("nomecliente"));
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

public static void editarClientes(Clientes cliente) {

    String sql = "UPDATE cliente SET nomecliente = ?, logradouro = ?, numlogradouro = ?, bairro = ?, cidade = ?, estado = ? WHERE idcliente = ?";
    Connection con = null;
    PreparedStatement ps = null;

    try {
        // Criando conexão com o banco
        con = Conexao.conectaDB();

        // Criando a classe pra executar a linha do MySQL
        ps = con.prepareStatement(sql);

        //Adicionando valores pra editar no banco
        ps.setString(1, cliente.getNomeCliente());
        ps.setString(2, cliente.getLogradouro());
        ps.setString(3, cliente.getNumLogradouro());
        ps.setString(4, cliente.getBairro());
        ps.setString(5, cliente.getCidade());
        ps.setString(6, cliente.getEstado());

        // Qual é o ID do cliente que será editado
        ps.setInt(7, cliente.getIdcliente());

        ps.execute();

    } catch (Exception e) {
        e.printStackTrace();
    }
}

public static void deletarClientes(int idCliente) {

    String sql = "DELETE FROM cliente WHERE idcliente = ?";

    Connection con = null;
    PreparedStatement ps = null;

    try {
        con = Conexao.conectaDB();

        ps = con.prepareStatement(sql);

        ps.setInt(1, idCliente);

```

```

        ps.execute();
    }
    catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

## ▼ Conexao

### ▼ Conexao

```

package Conexao;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class Conexao {

    public static Connection conectaDB() {
        Connection conexao = null;
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            try {
                conexao = DriverManager.getConnection("jdbc:mysql://localhost:3306/locadoraFilmes", "root", "");
            } catch (SQLException e) {
                e.printStackTrace();
            }
        } catch (ClassNotFoundException e) {
            System.out.println("Problema no driver JDBC" + e);
        }

        return conexao;
    }

    public static void main(String[] args) throws Exception {
        Connection con = conectaDB();
        if(con!=null) {
            System.out.println("Foi conectado com sucesso!");
            con.close();
        } else {
            System.out.println("Não foi possível estabelecer conexão.");
        }
    }
}

```

## ▼ Deletando

### ▼ DeletandoAluguel

```

package Deletando;

import java.util.Scanner;

import AluguelDAO.AluguelDAO;

public class DeletandoAluguel {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
        System.out.println("Aluguéis cadastrados: \n");
        AluguelDAO.mostrarAluguel();

        System.out.println("\nSelecione o ID do aluguel que você deseja apagar: ");
        int idDelAl = sc.nextInt();

        AluguelDAO.apagarAluguel(idDelAl);

        System.out.println("Aluguel apagado com sucesso!\n");

    }
}

```



```
}
```

#### ▼ DeletandoClientes

```
package Deletando;

import java.util.Scanner;
import ClientesDAO.ClientesDAO;

public class DeletandoClientes {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        ClientesDAO.mostrarClientes();

        System.out.println("Selecione o ID do cliente que você deseja DELETAR do banco de dados (Por números): ");
        int idDel = sc.nextInt();

        ClientesDAO.deletarClientes(idDel);

        System.out.println("Cliente apagado com sucesso!\n");
    }

}
```

#### ▼ DeletandoAluguel

```
package Deletando;
import java.util.Scanner;
import FilmesDAO.FilmesDAO;

public class DeletandoFilmes {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        FilmesDAO.mostrarFilmes();

        System.out.println("Selecione o ID do filme que você deseja deletar: ");
        int idDelFilm = sc.nextInt();

        FilmesDAO.apagarFilmes(idDelFilm);

        System.out.println("Filme apagado com sucesso!\n");

    }

}
```

#### ▼ Editando

##### ▼ EditandoAluguel

```
package Editando;
import java.util.Scanner;

import AluguelDAO.AluguelDAO;
import ClientesDAO.ClientesDAO;
import FilmesDAO.FilmesDAO;
import Modelos.Aluguel;
public class EditandoAluguel {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
```

```

        Aluguel al1 = new Aluguel();

        System.out.println("Informe o ID do aluguel que você deseja editar: ");
        int idal = sc.nextInt();
        al1.setIdaluguel(idal);

        ClientesDAO.mostrarClientes();
        System.out.println("Informe o novo ID do cliente que você deseja inserir: ");
        int idcl1 = sc.nextInt();
        al1.setIdCl(idcl1);

        FilmesDAO.mostrarFilmes();
        System.out.println("Agora informe o novo ID do filme que deseja inserir: ");
        int idfm1 = sc.nextInt();
        al1.setIdFilm(idfm1);

        AluguelDAO.editarAluguel(al1);

        System.out.println("Dados do aluguel " + idal + " modificados com sucesso!\n");
    }
}

```

### ▼ EditandoClientes

```

package Editando;

import java.util.Scanner;
import ClientesDAO.ClientesDAO;
import Modelos.Clientes;

public class EditandoClientes {

    public static void main(String[] args) {

        ClientesDAO.mostrarClientes();

        Scanner sc = new Scanner(System.in);

        Clientes c1 = new Clientes();

        System.out.println("\nInforme o ID do cliente que você deseja editar o registro: ");
        int id = sc.nextInt();
        c1.setIdcliente(id);

        System.out.println("Informe o novo nome do cliente: ");
        sc.nextLine();
        String novonome = sc.nextLine();
        c1.setnomeCliente(novonome);

        System.out.println("Informe o novo Logradouro: ");
        String novolog = sc.nextLine();
        c1.setLogradouro(novolog);

        System.out.println("Informe o novo numero do logradouro: ");
        String novonumlog = sc.nextLine();
        c1.setNumLogradouro(novonumlog);

        System.out.println("Informe o bairro atual do cliente: ");
        String novobairro = sc.nextLine();
        c1.setBairro(novobairro);

        System.out.println("Informe a cidade atual do cliente: ");
        String novacid = sc.nextLine();
        c1.setCidade(novacid);

        System.out.println("Agora informe o estado atual do cliente: ");
        String novoest = sc.nextLine();
        c1.setEstado(novoest);

        ClientesDAO.editarClientes(c1);

        System.out.println("Cliente " + novonome + " editado com sucesso!\n");
    }
}

```

### ▼ EditandoFilmes

```
package Editando;

import java.util.Scanner;
import FilmesDAO.FilmesDAO;
import Modelos.Filmes;

public class EditandoFilmes {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        FilmesDAO.mostrarFilmes();

        Filmes f1 = new Filmes();

        System.out.println("\nInforme o ID do filme que você deseja modificar: ");
        int id = sc.nextInt();
        f1.setIdFilme(id);

        System.out.println("Agora informe o nome que você deseja inserir: ");
        sc.nextLine();
        String novonomefilme = sc.nextLine();
        f1.setNomeFilme(novonomefilme);

        System.out.println("Agora informe o ano que você deseja modificar: ");
        int novoano = sc.nextInt();
        f1.setAnoFilme(novoano);

        System.out.println("Para finalizarmos, informe o novo gênero: ");
        sc.nextLine();
        String novogen = sc.nextLine();
        f1.setGenero(novogen);

        FilmesDAO.editarFilmes(f1);

        System.out.println("Filme " + novonomefilme + " editado com sucesso!\n");
    }
}
```

### ▼ FilmesDAO

#### ▼ FilmesDAO

```
package FilmesDAO;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import Conexao.Conexao;
import Modelos.Filmes;

public class FilmesDAO {

    public static void main(String[] args) {
        mostrarFilmes();
    }

    // Conexão
    static Conexao film = new Conexao();
    static Connection conexao = Conexao.conectaDB();

    public static void AdicionarFilmes(Filmes filme) {

        String sql = "INSERT INTO filme (nomefilme, anofilme, genero) VALUES (?, ?, ?)";
        Connection conexao = null;
        PreparedStatement ps = null;

        try {
            // Criando conexão com o banco de dados.
            conexao = Conexao.conectaDB();

            ps = conexao.prepareStatement(sql);
```

```

        // Adicionando valores para query
        ps.setString(1, filme.getNomeFilme());
        ps.setInt(2, filme.getAnoFilme());
        ps.setString(3, filme.getGenero());

        // Executando
        ps.execute();
    }
    catch(Exception e) {
        e.printStackTrace();
    }
}

public static void mostrarFilmes() {
    ResultSet filmes1;
    try {
        filmes1 = conexao.createStatement().executeQuery("SELECT * FROM filme");
        while (filmes1.next()) {
            System.out.println(filmes1.getInt("idfilme") + " " + filmes1.getString("nomefilme"));
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

public static void editarFilmes(Filmes filme) {

    String sql = "UPDATE filme SET nomefilme = ?, anofilme = ?, genero = ? WHERE idfilme = ?";
    Connection con = null;
    PreparedStatement ps = null;

    try {
        // Criando conexão com o banco
        con = Conexao.conectaDB();

        // Criando a classe pra executar a linha do MySQL
        ps = con.prepareStatement(sql);

        //Adicionando valores pra editar no banco
        ps.setString(1, filme.getNomeFilme());
        ps.setInt(2, filme.getAnoFilme());
        ps.setString(3, filme.getGenero());

        // Qual é o ID do Filme que será editado
        ps.setInt(4, (filme.getIdFilme()));

        ps.execute();

    } catch (Exception e) {
        e.printStackTrace();
    }
}

public static void apagarFilmes(int idFilme) {
    String sql = "DELETE FROM filme WHERE idfilme = ?";

    Connection con = null;
    PreparedStatement ps = null;

    try {
        con = Conexao.conectaDB();

        ps = con.prepareStatement(sql);

        ps.setInt(1, idFilme);

        ps.execute();
    }
    catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

## ▼ Modelos

### ▼ Aluguel

```
package Modelos;
```

```

import java.util.Date;

public class Aluguel {

    public int idaluguel;
    public Date dataaluguel;
    public int idCl;
    public int idFilm;

    public int getIdCl() {
        return idCl;
    }

    public void setIdCl(int idCl) {
        this.idCl = idCl;
    }

    public int getIdFilm() {
        return idFilm;
    }

    public void setIdFilm(int idFilm) {
        this.idFilm = idFilm;
    }

    public int getIdaluguel() {
        return idaluguel;
    }

    public void setIdaluguel(int idaluguel) {
        this.idaluguel = idaluguel;
    }

    public Date getDataaluguel() {
        return dataaluguel;
    }

    public void setDataaluguel(Date dataaluguel) {
        this.dataaluguel = dataaluguel;
    }
}

```

## ▼ Clientes

```

package Modelos;

public class Clientes {

    public int idcliente;
    public String NomeCliente;
    public String Logradouro;
    public String NumLogradouro;
    public String Bairro;
    public String Cidade;
    public String Estado;

    // Getters e Setters dos atributos do codigo
    public int getIdcliente() {
        return idcliente;
    }

    public void setIdcliente(int idcliente) {
        this.idcliente = idcliente;
    }

    public String getNomeCliente() {
        return NomeCliente;
    }

    public void setnomeCliente(String nomeCliente) {
        NomeCliente = nomeCliente;
    }

    public String getLogradouro() {
        return Logradouro;
    }

    public void setLogradouro(String logradouro) {
        Logradouro = logradouro;
    }
}

```

```

    }

    public String getNumLogradouro() {
        return NumLogradouro;
    }

    public void setNumLogradouro(String numLogradouro) {
        NumLogradouro = numLogradouro;
    }

    public String getBairro() {
        return Bairro;
    }

    public void setBairro(String bairro) {
        Bairro = bairro;
    }

    public String getCidade() {
        return Cidade;
    }

    public void setCidade(String cidade) {
        Cidade = cidade;
    }

    public String getEstado() {
        return Estado;
    }

    public void setEstado(String estado) {
        Estado = estado;
    }
}

```

#### ▼ Filme

```

package Modelos;

public class Filmes {

    public int idFilme;
    public String NomeFilme;
    public int AnoFilme;
    public String Genero;

    public int getIdFilme() {
        return idFilme;
    }

    public void setIdFilme(int idFilme) {
        this.idFilme = idFilme;
    }

    public String getNomeFilme() {
        return NomeFilme;
    }

    public void setNomeFilme(String nomeFilme) {
        NomeFilme = nomeFilme;
    }

    public int getAnoFilme() {
        return AnoFilme;
    }

    public void setAnoFilme(int anoFilme) {
        AnoFilme = anoFilme;
    }

    public String getGenero() {
        return Genero;
    }

    public void setGenero(String genero) {
        Genero = genero;
    }

}

```

## ▼ Mostrar

### ▼ MostrarAluguel

```
package Mostrar;

import AluguelDAO.AluguelDAO;

public class MostrarAluguel {

    public static void main(String[] args) {

        System.out.println("Aqui estão os alugueis realizados pelos clientes pelo ID de Cada Cliente/Filme: \n");
        AluguelDAO.mostrarAluguel();
        System.out.println("\n");

    }

}
```

### ▼ MostrarClientes

```
package Mostrar;
import ClientesDAO.ClientesDAO;

public class MostrarClientes {

    public static void main(String[] args) {
        System.out.println("Aqui constam os clientes cadastrados em banco atualmente: \n");
        ClientesDAO.mostrarClientes();
        System.out.println("\n");

    }

}
```

### ▼ MostrarFilme

```
package Mostrar;

import FilmesDAO.FilmesDAO;

public class MostrarFilmes {

    public static void main(String[] args) {

        System.out.println("Aqui está a lista com os filmes disponíveis no momento: \n");

        FilmesDAO.mostrarFilmes();
        System.out.println("\n");

    }

}
```

## ▼ FUNCIONAMENTO CÓDIGO SQL (MySQL Workbench)

### ▼ Criar database

```
create database locadorafilmes;
```

#### ▼ Usar database

```
use locadorafilmes;
```

#### ▼ Criar tabela cliente com seus atributos

```
create table cliente (  
  idcliente int primary key auto_increment,  
  nomecliente varchar (100) not null,  
  logradouro varchar (100) not null,  
  numlogradouro varchar(100) not null,  
  bairro varchar (100) not null,  
  cidade varchar (100) not null,  
  estado varchar (100) not null  
);
```

#### ▼ Criar tabela filme com seus atributos

```
create table filme (  
  idfilme int primary key auto_increment,  
  nomefilme varchar (100) not null,  
  anofilme int not null,  
  genero varchar (100) not null  
);
```

#### ▼ Criar tabela aluguel com seus atributos

```
create table aluguel (  
  idaluguel int primary key auto_increment,  
  dataaluguel date not null,  
  idcliente int not null,  
  idfilme int not null,  
  foreign key (idcliente) references cliente(idcliente),  
  foreign key (idfilme) references filme(idfilme)  
);
```

#### ▼ Mostrar que tabelas pertencem a database (cliente, filme, aluguel)

```
show tables;
```

#### ▼ Exibir todos os registros (\*) da tabela cliente

```
select * from cliente;
```

#### ▼ Exibir todos os registros (\*) da tabela filme

```
select * from filme;
```

#### ▼ Exibir todos os registros (\*) da tabela aluguel

```
select * from aluguel;
```



▼ Exibir todos os registros (\*) da tabela cliente na ordem crescente de idcliente

```
select * from cliente order by idcliente asc;
```

▼ Exibir todos os registros (\*) da tabela cliente na ordem crescente de nomecliente

```
select * from cliente order by nomecliente asc;
```

▼ Exibir todos os registros (\*) da tabela filme na ordem crescente de idfilme

```
select * from filme order by idfilme asc;
```

▼ Exibir todos os registros (\*) da tabela filme na ordem crescente de nomefilme

```
select * from filme order by nomefilme asc;
```

▼ Exibir todos os registros (\*) da tabela aluguel na ordem crescente de idaluguel

```
select * from cliente order by idaluguel asc;
```

▼ Exibir todos os registros (\*) da tabela aluguel na ordem crescente de dataaluguel

```
select * from aluguel order by dataaluguel asc;
```

▼ Inserir cliente (na tabela cliente)

```
insert into cliente (nomecliente, logradouro, numlogradouro, bairro, cidade, estado)
values ("", "", "", "", "", "");

/* Exemplo de preenchimento:
insert into cliente (nomecliente, logradouro, numlogradouro, bairro, cidade, estado)
values ("Fulano Oliveira", "Rua Beltrano", "1111", "Cicrano", "Fulanópolis", "Beltrano do Sul");
*/
```

▼ Inserir filme (na tabela filme)

```
insert into filme (nomefilme, anofilme, genero)
values ("", , "");

/* Exemplo de preenchimento:
insert into filme (nomefilme, anofilme, genero)
values ("The Batman", 2022, "Aventura");
*/
```

#### ▼ Inserir aluguel (na tabela aluguel)

```
insert into aluguel (idcliente_cliente, idfilme_filme)
values ( , );

/* Exemplo de preenchimento:
insert into aluguel (idaluguel, dataaluguel, idcliente, idfilme)
values (3, 3);
*/
```

#### ▼ Editar cliente (selecionando pelo idcliente qual cliente será editado)

```
/* Update completo do registro do cliente: */
UPDATE cliente SET nomecliente = "", logradouro = "", numlogradouro = "", bairro = "", cidade = "", estado = "" WHERE idcli
ente = ;
/* Caso seja uma edição parcial (apenas parte dos atributos do registro), excluir do SET os atributos que não deseja editar
*/

/* Exemplo de preenchimento:
UPDATE cliente SET nomecliente = "cliente teste", logradouro = "testelog", numlogradouro = "numlogteste", bairro = "bairrot
este, cidade = "cidadeteste", estado = "estadoteste" WHERE idcliente = 1;

Exemplo de preenchimento:
UPDATE cliente SET nomecliente = "cliente teste" WHERE idcliente = 1;
*/
```

#### ▼ Editar filme (selecionando pelo idfilme qual filme será editado)

```
/* Update completo do registro do filme: */
UPDATE filme SET nomefilme = "", anofilme = , genero = "" WHERE idfilme = ;
/* Caso seja uma edição parcial (apenas parte dos atributos do registro), excluir do SET os atributos que não deseja editar
*/

/* Exemplo de preenchimento:
UPDATE filme SET nomefilme = "filmeteste", anofilme = 1, genero = "teste" WHERE idfilme = 1;

Exemplo de preenchimento:
UPDATE filme SET nomefilme = "filmeteste" WHERE idfilme = 1;
*/
```

#### ▼ Editar aluguel (selecionando pelo idaluguel qual aluguel será editado)

```
/* Update completo do registro do aluguel: */
UPDATE aluguel SET idcliente = , idaluguel = WHERE idaluguel = ;
/* Caso seja uma edição parcial (apenas parte dos atributos do registro), excluir do SET os atributos que não deseja editar
*/

/* Exemplo de preenchimento:
UPDATE aluguel SET dataaluguel = '2022-12-31', idcliente = 1, idaluguel = 3 WHERE idaluguel = 3;

Exemplo de preenchimento:
UPDATE aluguel SET dataaluguel = '2022-12-31' WHERE idaluguel = 3;
*/
```

#### ▼ Excluir cliente (selecionando pelo idcliente qual cliente será excluído)

```
DELETE FROM cliente WHERE idcliente = ;

/* Exemplo de preenchimento:
DELETE FROM cliente WHERE idcliente = 1;
*/
```

▼ Excluir filme (selecionando pelo idfilme qual cliente será excluído)

```
DELETE FROM filme WHERE idfilme = ;

/* Exemplo de preenchimento:
DELETE FROM filme WHERE idfilme = 1;
*/
```

▼ Excluir aluguel (selecionando pelo idaluguel qual cliente será excluído)

```
DELETE FROM aluguel WHERE idaluguel = ;

/* Exemplo de preenchimento:
DELETE FROM aluguel WHERE idaluguel = 1;
*/
```

▼ FUNCIONAMENTO CÓDIGO JAVA (Eclipse for Java)

- Cada toggle list é um package. Dentro desse package, cada toggle list é uma classe do package. Dentro das classes, estão seus respectivos códigos;
- Todos os packages fazem parte do Java Project LocadoraFilme;
- O código foi criado e executado na IDE Eclipse.

▼ AluguelDAO

▼ AluguelDAO

Contém todos os métodos que serão utilizados para realizar a inserção, visualização, atualização e deleção no banco de dados.

```
package AluguelDAO;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

import Conexao.Conexao;
import Modelos.Aluguel;
```

Cria uma nova conexão com o banco.

```
public class AluguelDAO {

    // Conexão
    static Conexao cl = new Conexao();
    static Connection conexao = Conexao.conectaDB();
```

Método CadastrarAluguel (Aluguel aluguel) tenta adicionar um novo aluguel pelo método try do tipo aluguel, caso não consiga, retorna com o método catch com uma mensagem de erro.

```
public static void cadastrarAluguel(Aluguel aluguel) {

    String sql = "INSERT INTO aluguel (idcliente, idfilme) VALUES (?, ?)";
    Connection con = null;
    PreparedStatement ps = null;

    try {

        con = Conexao.conectaDB();
```

```

        ps = conexao.prepareStatement(sql);

        ps.setInt(1, aluguel.getIdCl());
        ps.setInt(2, aluguel.getIdFilme());

        ps.execute();

    } catch (Exception e) {
        e.printStackTrace();
    }
}

```

Método `mostrarAluguel()` tenta exibir os alugueis cadastrados em banco pelo método `try`, caso não obtenha sucesso, retorna uma mensagem de erro pelo método `catch`.

```

public static void mostrarAluguel() {
    ResultSet aluguel1;
    try {
        aluguel1 = conexao.createStatement().executeQuery("SELECT * FROM aluguel");
        while (aluguel1.next()) {
            System.out.println(aluguel1.getInt("idaluguel") + " - " + "ID cliente: " + aluguel1.getInt("idcliente") + ", ID F
            + aluguel1.getDate("dataaluguel"));
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

```

Método `editarAluguel(Aluguel aluguel)` tenta editar um aluguel do tipo `Aluguel` cadastrado em um banco pelo método `try`, caso não obtenha sucesso, retorna uma mensagem de erro pelo método `catch`.

```

public static void editarAluguel(Aluguel aluguel) {

    String sql = "UPDATE aluguel SET idcliente = ?, idfilme = ? WHERE idaluguel = ?";

    Connection con = null;
    PreparedStatement ps = null;

    try {
        con = Conexao.conectaDB();

        ps = con.prepareStatement(sql);

        ps.setInt(1, aluguel.getIdCl());
        ps.setInt(2, aluguel.getIdFilme());

        ps.setInt(3, aluguel.getIdAluguel());

        ps.execute();

    } catch (Exception e) {
        e.printStackTrace();
    }

}

```

Método `apagarAluguel(int idAluguel)` tenta deletar um aluguel do tipo `Aluguel` cadastrado em banco pela chave primária (`idaluguel`), caso não consiga, retorna uma mensagem de erro pelo método `catch`.

```

public static void apagarAluguel(int idAluguel) {

    String sql = "DELETE FROM aluguel WHERE idaluguel = ?";

    Connection con = null;
    PreparedStatement ps = null;

    try {

        con = Conexao.conectaDB();

        ps = con.prepareStatement(sql);

```

```

        ps.setInt(1, idAluguel);

        ps.execute();

    } catch (Exception e) {
        e.printStackTrace();
    }

}

}

```

#### ▼ Cadastrar

Cria dados para serem inseridos no BD, podendo ser cadastrados: Clientes, Filmes e Aluguéis realizados.

##### ▼ CadastrandoClientes

Cria um novo Cliente no banco de dados, com os atributos idcliente(Auto incremental), nomecliente, logradouro, numlogradouro, bairro, cidade, estado.

```

package Cadastrar;

import java.util.Scanner;

import ClientesDAO.ClientesDAO;
import Modelos.Clientes;

```

A classe public class CadastrandoClientes é uma classe que contém o método executável main para cadastrar novos clientes em banco.

```

public class CadastrandoClientes {

    public static void main(String[] args) {

        // CADASTRAR CLIENTES
        Scanner sc = new Scanner(System.in);

        Clientes cliente = new Clientes();

        System.out.println("Insira o nome do cliente: ");
        String nome = sc.nextLine();
        cliente.setnomeCliente(nome);

        System.out.println("Insira o Logradouro do cliente: ");
        String logradouro = sc.nextLine();
        cliente.setLogradouro(logradouro);

        System.out.println("Insira o numero do logradouro: ");
        String NumLogradouro = sc.nextLine();
        cliente.setNumLogradouro(NumLogradouro);

        System.out.println("Insira o bairro onde o cliente reside: ");
        String bairro = sc.nextLine();
        cliente.setBairro(bairro);

        System.out.println("Insira a cidade onde o cliente reside: ");
        String cidade = sc.nextLine();
        cliente.setCidade(cidade);

        System.out.println("Insira o estado onde o cliente reside: ");
        String estado = sc.nextLine();
        cliente.setEstado(estado);

        ClientesDAO.criarCliente(cliente);

    }

}

```

##### ▼ CadastrandoFilmes

Cria um novo Filme no banco de dados, com os atributos idfilme(Auto incremental), nomefilme, anofilme, genero.

```

package Cadastrar;

import java.util.Scanner;

```

```
import FilmesDAO.FilmesDAO;
import Modelos.Filmes;
```

A classe public class CadastrandoFilmes é uma classe que contém o método executável main para cadastrar novos filmes em banco.

```
public class CadastrandoFilmes {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        Filmes filme = new Filmes();

        System.out.println("Informe qual filme você deseja adicionar: ");
        String novofilme = sc.nextLine();
        filme.setNomeFilme(novofilme);

        System.out.println("Agora informe o ano em que o filme foi lançado: ");
        int anolanc = sc.nextInt();
        filme.setAnoFilme(anolanc);

        System.out.println("Para finalizar, informe o gênero do filme em questão: ");
        sc.nextLine();
        String genfilm = sc.nextLine();
        filme.setGenero(genfilm);

        FilmesDAO.AdicionarFilmes(filme);

    }

}
```

#### ▼ CadastrarAluguel

Cria um novo Aluguel no banco de dados com os atributos idaluguel(Auto incremental), dataaluguel, idcliente(Foreign key), idfilme (Foreign key)

```
package Cadastrar;
import java.util.Scanner;

import AluguelDAO.AluguelDAO;
import ClientesDAO.ClientesDAO;
import FilmesDAO.FilmesDAO;
import Modelos.Aluguel;
```

A classe public class CadastrarAluguel é uma classe que contém o método executável main para cadastrar novos alugueis em banco.

```
public class CadastrarAluguel {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        Aluguel alg = new Aluguel();

        ClientesDAO.mostrarClientes();

        System.out.println("Informe o ID do cliente que deseja realizar o aluguel: ");
        int idcl = sc.nextInt();
        alg.setIdCl(idcl);

        FilmesDAO.mostrarFilmes();
        System.out.println("Agora informe o ID do filme que ele deseja Alugar:\n ");
        int idfm = sc.nextInt();
        alg.setIdFilm(idfm);

        AluguelDAO.cadastrarAluguel(alg);

        System.out.println("Aluguel realizado com sucesso!\n");

    }

}
```

## ▼ ClassePrincipal

ClassePrincipal contém imports de todas as classes e métodos utilizados no código

### ▼ PaginaInicial

```
package ClassePrincipal;
import java.util.Scanner;

import Cadastrar.CadastrandoClientes;
import Cadastrar.CadastrandoFilmes;
import Cadastrar.CadastrarAluguel;
import Deletando.DeletandoAluguel;
import Deletando.DeletandoClientes;
import Deletando.DeletandoFilmes;
import Editando.EditandoAluguel;
import Editando.EditandoClientes;
import Editando.EditandoFilmes;
import Mostrar.MostrarAluguel;
import Mostrar.MostrarClientes;
import Mostrar.MostrarFilmes;
```

Classe executável para centralizar todos os métodos usados em todas as classes, para serem realizadas individualmente por meio de opções escolhidas pelo usuário.

```
public class PaginaInicial {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.println("Deseja realizar uma operação? 'sim' ou 'não'");
        String opt = sc.nextLine();
        while (opt.equalsIgnoreCase("sim")) {

            System.out.println(
                "1 - Cadastrar Clientes \n2 - Cadastrar filmes \n3 - Cadastrar Aluguel \n4 - Editar registro de cliente "
                + "\n5 - Editar registro de filme \n6 - Editar registro de Aluguel \n7 - Visualizar tabela de clientes \n8
                + "\n9 - Visualizar tabela de alugueis "
                + "\n10 - Excluir registro de cliente \n11 - Excluir registro de filme \n12 - Excluir registro de aluguel")

            int opcao = sc.nextInt();

            if (opcao == 1) {
                CadastroClientes.main(args);
            } else if (opcao == 2) {
                CadastroFilmes.main(args);
            } else if (opcao == 3) {
                CadastroAluguel.main(args);
            } else if (opcao == 4) {
                EditandoClientes.main(args);
            } else if (opcao == 5) {
                EditandoFilmes.main(args);
            } else if (opcao == 6) {
                EditandoAluguel.main(args);
            } else if (opcao == 7) {
                MostrarClientes.main(args);
            } else if (opcao == 8) {
                MostrarFilmes.main(args);
            } else if (opcao == 9) {
                MostrarAluguel.main(args);
            } else if (opcao == 10) {
                DeletandoClientes.main(args);
            } else if (opcao == 11) {
                DeletandoFilmes.main(args);
            } else if (opcao == 12) {
                DeletandoAluguel.main(args);
            } else {
                System.out.println("Opção inválida!!!");
            }

        }

        System.out.println("Até a próxima pequeno gafanhoto!!!");
    }
}
```

## ▼ ClientesDAO

### ▼ ClientesDAO

Contém todos os métodos que serão utilizados para realizar a inserção, visualização, atualização e deleção no banco de dados.

```
package ClientesDAO;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

import com.mysql.cj.PreparedQuery;
import com.mysql.cj.jdbc.ClientPreparedStatement;

import Conexao.Conexao;
import Modelos.Clientes;
```

Cria uma nova conexão com o banco.

```
public class ClientesDAO {

    public static void main(String[] args) {
        mostrarClientes();
    }

    // Conexão
    static Conexao cl = new Conexao();
    static Connection conexao = Conexao.conectaDB();
```

Método criarCliente (Clientes cliente) tenta realizar a criação de um novo cliente pelo método try do tipo Clientes, caso não consiga, retorna com o método catch com uma mensagem de erro.

```
public static void criarCliente(Clientes cliente) {
    String sql = "INSERT INTO cliente (nomecliente, logradouro, numlogradouro, bairro, cidade, estado) VALUES (?, ?, ?, ?, ?, ?)";
    Connection conexao = null;
    PreparedStatement ps = null;

    try {
        // Criando conexão com o banco de dados.
        conexao = Conexao.conectaDB();

        ps = conexao.prepareStatement(sql);

        // Adicionando valores para query
        ps.setString(1, cliente.getNomeCliente());
        ps.setString(2, cliente.getLogradouro());
        ps.setString(3, cliente.getNumLogradouro());
        ps.setString(4, cliente.getBairro());
        ps.setString(5, cliente.getCidade());
        ps.setString(6, cliente.getEstado());

        // Executando
        ps.execute();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Método mostrarClientes() tenta exibir os clientes cadastrados em banco pelo método try, caso não obtenha sucesso, retorna uma mensagem de erro pelo método catch.

```
public static void mostrarClientes() {
    ResultSet cliente1;
    try {
        cliente1 = conexao.createStatement().executeQuery("SELECT * FROM cliente");
        while (cliente1.next()) {
            System.out.println(cliente1.getInt("idcliente") + " " + cliente1.getString("nomecliente"));
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```



Método `editarClientes(Clientes cliente)` tenta editar um cliente do tipo `Cientes` cadastrado em um banco pelo método `try`, caso não obtenha sucesso, retorna uma mensagem de erro pelo método `catch`.

```
public static void editarClientes(Clientes cliente) {

    String sql = "UPDATE cliente SET nomecliente = ?, logradouro = ?, numlogradouro = ?, bairro = ?, cidade = ?, estado = ?";
    Connection con = null;
    PreparedStatement ps = null;

    try {
        // Criando conexão com o banco
        con = Conexao.conectaDB();

        // Criando a classe pra executar a linha do MySQL
        ps = con.prepareStatement(sql);

        //Adicionando valores pra editar no banco
        ps.setString(1, cliente.getNomeCliente());
        ps.setString(2, cliente.getLogradouro());
        ps.setString(3, cliente.getNumLogradouro());
        ps.setString(4, cliente.getBairro());
        ps.setString(5, cliente.getCidade());
        ps.setString(6, cliente.getEstado());

        // Qual é o ID do cliente que será editado
        ps.setInt(7, cliente.getIdcliente());

        ps.execute();

    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Método `deletarClientes(int idcliente)` tenta deletar um cliente do tipo `Cientes` cadastrado em banco pela chave primária (`idcliente`), caso não consiga, retorna uma mensagem de erro pelo método `catch`.

```
public static void deletarClientes(int idCliente) {

    String sql = "DELETE FROM cliente WHERE idcliente = ?";

    Connection con = null;
    PreparedStatement ps = null;

    try {
        con = Conexao.conectaDB();

        ps = con.prepareStatement(sql);

        ps.setInt(1, idCliente);

        ps.execute();
    }
    catch (Exception e) {
        e.printStackTrace();
    }
}
```

#### ▼ Conexao

##### ▼ Conexao

Classe criada para estabelecer uma ponte de conexão entre o código Java, e o Banco de Dados MySQL.

```
package Conexao;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
```

Classe do tipo `Connection` que tenta estabelecer e no final retornar uma conexão com a database pelo método `try`, com o endereço local, usuário e senha, caso não obtenha sucesso, retorna uma mensagem de erro pelo método

catch.

```
public static Connection conectaDB() {
    Connection conexao = null;
    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
        try {
            conexao = DriverManager.getConnection("jdbc:mysql://localhost:3306/locadoraFilmes", "root", "");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    } catch (ClassNotFoundException e) {
        System.out.println("Problema no driver JDBC" + e);
    }

    return conexao;
}
```

Classe executável que verifica se a conexão foi realizada. Caso a conexão seja diferente de "null", retorna uma mensagem de sucesso. Caso não obtenha êxito, retorna uma mensagem de erro.

```
public static void main(String[] args) throws Exception {
    Connection con = conectaDB();
    if(con!=null) {
        System.out.println("Foi conectado com sucesso!");
        con.close();
    } else {
        System.out.println("Não foi possível estabelecer conexão.");
    }
}
}
```

#### ▼ Deletando

Deleta os dados que já estão cadastrados no BD pelo ID. DeletandoClientes para tabela Clientes. DeletandoFilmes para tabela Filmes. DeletandoAluguel para tabela Aluguel. As classes desse package são executáveis.

##### ▼ DeletandoClientes

Classe que contém todos os métodos para deletar os clientes que forem selecionados pelo id

```
package Deletando;

import java.util.Scanner;
import ClientesDAO.ClientesDAO;
```

Classe executável que mostra a lista de clientes cadastrados pra que possam ser escolhidos através do id, deletando o registro do cliente do banco de dados

```
public class DeletandoClientes {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        ClientesDAO.mostrarClientes();

        System.out.println("Selecione o ID do cliente que você deseja DELETAR do banco de dados (Por números): ");
        int idDel = sc.nextInt();

        ClientesDAO.deletarClientes(idDel);

    }
}
```

##### ▼ DeletandoFilmes

Classe que contém todos os métodos para deletar os filmes que foram cadastrados no banco de dados que forem selecionados pelo id

```
package Deletando;
import java.util.Scanner;
```

Classe executável que mostra a lista de filmes cadastrados pra que possam ser escolhidos através do id, deletando o registro do filme do banco de dados

```
public class DeletandoFilmes {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        FilmesDAO.mostrarFilmes();

        System.out.println("Selecione o ID do filme que você deseja deletar: ");
        int idDelFilm = sc.nextInt();

        FilmesDAO.apagarFilmes(idDelFilm);

    }

}
```

#### ▼ DeletandoAluguel

Classe que contém todos os métodos para deletar os alugueis que foram cadastrados no banco de dados que forem selecionados pelo id

```
package Deletando;

import java.util.Scanner;

import AluguelDAO.AluguelDAO;
```

Classe executável que mostra a lista de alugueis cadastrados pra que possam ser escolhidos através do id, deletando o registro do aluguel do banco de dados

```
public class DeletandoAluguel {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
        System.out.println("Aluguéis cadastrados: \n");
        AluguelDAO.mostrarAluguel();

        System.out.println("\nSelecione o ID do aluguel que você deseja apagar: ");
        int idDelAl = sc.nextInt();

        AluguelDAO.apagarAluguel(idDelAl);

        System.out.println("Aluguel apagado com sucesso!\n");

    }

}
```

#### ▼ Editando

Edita os dados que já estão cadastrados no BD pelo ID. EditandoClientes para tabela Clientes. EditandoFilmes para tabela Filmes. EditandoAluguel para tabela Aluguel. As classes desse package são executáveis.

#### ▼ EditandoClientes

Edita dados de um cliente já cadastrado em banco.

```
package Editando;

import java.util.Scanner;
import ClientesDAO.ClientesDAO;
import Modelos.Clientes;
```

A classe public class EditandoClientes é uma classe que contém o método executável main para executar a edição dos clientes já cadastrados em banco.

```
public class EditandoClientes {

    public static void main(String[] args) {

        ClientesDAO.mostrarClientes();

        Scanner sc = new Scanner(System.in);

        Clientes c1 = new Clientes();

        System.out.println("\nInforme o ID do cliente que você deseja editar o registro: ");
        int id = sc.nextInt();
        c1.setIdcliente(id);

        System.out.println("Informe o novo nome do cliente: ");
        sc.nextLine();
        String novonome = sc.nextLine();
        c1.setnomeCliente(novonome);

        System.out.println("Informe o novo Logradouro: ");
        String novolog = sc.nextLine();
        c1.setLogradouro(novolog);

        System.out.println("Informe o novo numero do logradouro: ");
        String novonumlog = sc.nextLine();
        c1.setNumLogradouro(novonumlog);

        System.out.println("Informe o bairro atual do cliente: ");
        String novobairro = sc.nextLine();
        c1.setBairro(novobairro);

        System.out.println("Informe a cidade atual do cliente: ");
        String novacid = sc.nextLine();
        c1.setCidade(novacid);

        System.out.println("Agora informe o estado atual do cliente: ");
        String novoest = sc.nextLine();
        c1.setEstado(novoest);

        ClientesDAO.editarClientes(c1);

    }
}
```

#### ▼ EditandoFilmes

Edita dados de um filme já cadastrado em banco.

```
package Editando;

import java.util.Scanner;
import FilmesDAO.FilmesDAO;
import Modelos.Filmes;
```

A classe public class EditandoFilmes é uma classe que contém o método main para executar a edição dos filmes já cadastrados em banco.

```
public class EditandoFilmes {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        FilmesDAO.mostrarFilmes();

        Filmes f1 = new Filmes();

        System.out.println("\nInforme o ID do filme que você deseja modificar: ");
        int id = sc.nextInt();
        f1.setIdFilme(id);

        System.out.println("Agora informe o nome que você deseja inserir: ");
        sc.nextLine();
        String novonomefilme = sc.nextLine();
        f1.setNomeFilme(novonomefilme);

    }
}
```

```

        System.out.println("Agora informe o ano que você deseja modificar: ");
        int novoano = sc.nextInt();
        f1.setAnoFilme(novoano);

        System.out.println("Para finalizarmos, informe o novo gênero: ");
        sc.nextLine();
        String novogen = sc.nextLine();
        f1.setGenero(novogen);

        FilmesDAO.editarFilmes(f1);
    }
}

```

#### ▼ EditandoAluguel

Edita dados de um aluguel já cadastrado em banco.

```

package Editando;
import java.util.Scanner;

import AluguelDAO.AluguelDAO;
import ClientesDAO.ClientesDAO;
import FilmesDAO.FilmesDAO;
import Modelos.Aluguel;

```

A classe public class EditandoAluguel é uma classe que contém o método main para executar a edição dos aluguéis já cadastrados em banco.

```

public static void main(String[] args) {

    Scanner sc = new Scanner(System.in);

    Aluguel al1 = new Aluguel();

    System.out.println("Informe o ID do aluguel que você deseja editar: ");
    int idal = sc.nextInt();
    al1.setIdaluguel(idal);

    ClientesDAO.mostrarClientes();
    System.out.println("Informe o novo ID do cliente que você deseja inserir: ");
    int idcl1 = sc.nextInt();
    al1.setIdCl(idcl1);

    FilmesDAO.mostrarFilmes();
    System.out.println("Agora informe o novo ID do filme que deseja inserir: ");
    int idfm1 = sc.nextInt();
    al1.setIdFilm(idfm1);

    AluguelDAO.editarAluguel(al1);

    System.out.println("Dados do aluguel " + idal + " modificados com sucesso!\n");

}
}

```

#### ▼ FilmesDAO

##### ▼ FilmesDAO

Contém todos os métodos que serão utilizados para realizar a inserção, visualização, atualização e deleção de filmes no banco de dados.

```

package FilmesDAO;

import java.sql.Connection;

```

Cria uma nova conexão com o banco

```

public class FilmesDAO {

    public static void main(String[] args) {

```

```

    mostrarFilmes();
}

// Conexão
static Conexao film = new Conexao();
static Connection conexao = Conexao.conectaDB();

```

Método AdicionarFilmes (Filmes filme) tenta adicionar um novo filme pelo método try do tipo filmes, caso não consiga, retorna com o método catch com uma mensagem de erro.

```

public static void AdicionarFilmes(Filmes filme) {

    String sql = "INSERT INTO filme (nomefilme, anofilme, genero) VALUES (?, ?, ?)";
    Connection conexao = null;
    PreparedStatement ps = null;

    try {
        // Criando conexão com o banco de dados.
        conexao = Conexao.conectaDB();

        ps = conexao.prepareStatement(sql);

        // Adicionando valores para query
        ps.setString(1, filme.getNomeFilme());
        ps.setInt(2, filme.getAnoFilme());
        ps.setString(3, filme.getGenero());

        // Executando
        ps.execute();
    }
    catch (Exception e) {
        e.printStackTrace();
    }
}

```

Método mostrarFilmes() tenta exibir os filmes cadastrados em banco pelo método try, caso não obtenha sucesso, retorna uma mensagem de erro pelo método catch.

```

public static void mostrarFilmes() {
    ResultSet filmes1;
    try {
        filmes1 = conexao.createStatement().executeQuery("SELECT * FROM filme");
        while (filmes1.next()) {
            System.out.println(filmes1.getInt("idfilme") + " " + filmes1.getString("nomefilme"));
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

```

Método editarFilmes(Filmes filme) tenta editar um filme do tipo filmes cadastrado em um banco pelo método try, caso não obtenha sucesso, retorna uma mensagem de erro pelo método catch.

```

public static void editarFilmes(Filmes filme) {

    String sql = "UPDATE filme SET nomefilme = ?, anofilme = ?, genero = ? WHERE idfilme = ?";
    Connection con = null;
    PreparedStatement ps = null;

    try {
        // Criando conexão com o banco
        con = Conexao.conectaDB();

        // Criando a classe pra executar a linha do MySQL
        ps = con.prepareStatement(sql);

        //Adicionando valores pra editar no banco
        ps.setString(1, filme.getNomeFilme());
        ps.setInt(2, filme.getAnoFilme());
        ps.setString(3, filme.getGenero());

        // Qual é o ID do Filme que será editado
        ps.setInt(4, filme.getIdFilme());

        ps.execute();
    } catch (Exception e) {

```

```

        e.printStackTrace();
    }
}

```

Método `deletarFilmes(int idfilme)` tenta deletar um filme do tipo Filmes cadastrado em banco pela chave primária (idfilme), caso não consiga, retorna uma mensagem de erro pelo método `catch`.

```

public static void apagarFilmes(int idFilme) {
    String sql = "DELETE FROM filme WHERE idfilme = ?";

    Connection con = null;
    PreparedStatement ps = null;

    try {
        con = Conexao.conectaDB();

        ps = con.prepareStatement(sql);

        ps.setInt(1, idFilme);

        ps.execute();
    }
    catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

## ▼ Modelos

### ▼ Clientes

Contém todos os Atributos e métodos da classe clientes que serão utilizados nas outras classes

```

package Modelos;

```

Atributos da classe Clientes

```

public class Clientes {

    public int idcliente;
    public String NomeCliente;
    public String Logradouro;
    public String NumLogradouro;
    public String Bairro;
    public String Cidade;
    public String Estado;
}

```

Dando valores a os atributos da classe clientes através dos métodos de encapsulamento `get` e `set` que padroniza o gerenciamento sobre o acesso dos atributos

```

public int getIdcliente() {
    return idcliente;
}

public void setIdcliente(int idcliente) {
    this.idcliente = idcliente;
}

public String getNomeCliente() {
    return NomeCliente;
}

public void setnomeCliente(String nomeCliente) {
    NomeCliente = nomeCliente;
}

public String getLogradouro() {
    return Logradouro;
}

public void setLogradouro(String logradouro) {
    Logradouro = logradouro;
}

```

```

    public String getNumLogradouro() {
        return NumLogradouro;
    }

    public void setNumLogradouro(String numLogradouro) {
        NumLogradouro = numLogradouro;
    }

    public String getBairro() {
        return Bairro;
    }

    public void setBairro(String bairro) {
        Bairro = bairro;
    }

    public String getCidade() {
        return Cidade;
    }

    public void setCidade(String cidade) {
        Cidade = cidade;
    }

    public String getEstado() {
        return Estado;
    }

    public void setEstado(String estado) {
        Estado = estado;
    }
}

```

#### ▼ Filmes

Contém todos os Atributos e métodos da classe Filmes que serão utilizados nas outras classes

```
package Modelos;
```

#### Atributos da classe Filmes

```

public class Filmes {

    public int idFilme;
    public String NomeFilme;
    public int AnoFilme;
    public String Genero;
}

```

Dando valores a os atributos da classe Filme através dos métodos de encapsulamento get e set que padroniza o gerenciamento sobre o acesso dos atributos

```

    public int getIdFilme() {
        return idFilme;
    }

    public void setIdFilme(int idFilme) {
        this.idFilme = idFilme;
    }

    public String getNomeFilme() {
        return NomeFilme;
    }

    public void setNomeFilme(String nomeFilme) {
        NomeFilme = nomeFilme;
    }

    public int getAnoFilme() {
        return AnoFilme;
    }

    public void setAnoFilme(int anoFilme) {
        AnoFilme = anoFilme;
    }

    public String getGenero() {
        return Genero;
    }
}

```



```

        public void setGenero(String genero) {
            Genero = genero;
        }

    }
}

```

#### ▼ Aluguel

Contém todos os atributos da Classe associativa Aluguel:

```
package Modelos;
```

Atributos da classe Aluguel:

```

import java.util.Date;

public class Aluguel {

    public int idaluguel;
    public Date dataaluguel;
    public int idCl;
    public int idFilm;
}

```

Dando valores a os atributos da classe Aluguel através dos métodos de encapsulamento get e set que padroniza o gerenciamento sobre o acesso dos atributos:

```

public int getIdCl() {
    return idCl;
}

public void setIdCl(int idCl) {
    this.idCl = idCl;
}

public int getIdFilm() {
    return idFilm;
}

public void setIdFilm(int idFilm) {
    this.idFilm = idFilm;
}

public int getIdaluguel() {
    return idaluguel;
}

public void setIdaluguel(int idaluguel) {
    this.idaluguel = idaluguel;
}

public Date getDataaluguel() {
    return dataaluguel;
}

public void setDataaluguel(Date dataaluguel) {
    this.dataaluguel = dataaluguel;
}
}

```

#### ▼ Mostrar

Mostra os dados que já estão cadastrados no BD pelo ID. MostrarClientes para tabela Clientes. MostrarFilmes para tabela Filmes. MostrarAluguel para tabela Aluguel. As classes desse package são executáveis.

#### ▼ MostrarAluguel

Mostra dados de um aluguel já cadastrado em banco.

```

package Mostrar;

import AluguelDAO.AluguelDAO;

```

A classe public class MostrarAluguel é uma classe que contém o método main para executar a visualização dos alugueis já cadastrados em banco.

```
public class MostrarAluguel {  
  
    public static void main(String[] args) {  
  
        System.out.println("Aqui estão os alugueis realizados pelos clientes pelo ID de Cada Cliente/Filme: \n");  
        AluguelDAO.mostrarAluguel();  
        System.out.println("\n");  
  
    }  
  
}
```

#### ▼ MostrarClientes

Mostra os dados do cliente já cadastrados no banco de dados

```
package Mostrar;  
import ClientesDAO.ClientesDAO;
```

A classe public class MostrarClientes é uma classe que contém o método main para executar a visualização dos nomes dos clientes já cadastrados em banco.

```
public class MostrarClientes {  
  
    public static void main(String[] args) {  
  
        ClientesDAO.mostrarClientes();  
  
    }  
  
}
```

Mostra os dados de filme já cadastrados no banco de dados

```
package Mostrar;  
  
import FilmesDAO.FilmesDAO;
```

#### ▼ MostrarFilmes

A classe public class MostrarFilmes é uma classe que contém o método main para executar a visualização da lista dos filmes já cadastrados em banco.

```
public class MostrarFilmes {  
  
    public static void main(String[] args) {  
  
        System.out.println("Aqui está a lista com os filmes disponíveis no momento: \n");  
  
        FilmesDAO.mostrarFilmes();  
  
    }  
  
}
```