

Modelos

Modelo físico do banco de dados

□

Modelo lógico do banco de dados

□

Modelo conceitual do banco de dados

□

O banco de dados contém as seguintes entidades:

- movie
- actor
- poster
- room
- session
- ticket
- snack
- transaction

Ao registrar num novo filme, o sistema irá criar um novo registro na tabela `movie` e um novo registro na tabela `poster`. O mesmo acontece com os atores, que são registrados na tabela `actor`. Para o caso dos atores na hora do registro do filme é passado uma grande String contendo o nome de todos os atores principais `mainCast` que estão presentes no filme. Então, o sistema irá separar os nomes dos atores, através das vírgulas, e irá criar um registro na tabela `actor` para cada ator presente na String.

Ao registrar uma nova sala, o sistema irá criar um novo registro na tabela `room`. É possível linkar cada filme cadastrado no sistema à uma sala através de uma `session`. Ao registrar uma nova sessão, o sistema irá criar um novo registro na tabela `session` e irá atualizar o número de assentos disponíveis na sala. O mesmo acontece com os ingressos, que são registrados na tabela `ticket`. Ao registrar um novo ingresso, o sistema irá criar um novo registro na tabela `ticket` e irá atualizar o número de assentos disponíveis na sessão que está atrelada àquele `ticket`.

A tabela `transaction` recebe os dados de uma transação de compra de ingresso, que pode ser feita com cartão de crédito ou dinheiro, e os lanches comprados na hora da compra do ingresso.

SQL

Criação das tabelas

A criação de tabela foi toda feita por intermédio da dependência `Hibernate` e as manipulações foram feitas por meio do `Spring Data JPA`.

Exemplos

```
create table poster (  
    id bigserial not null,  
    url varchar(255),  
    movie_id int8,  
    primary key (id)  
);
```

```
create table room (  
    id bigserial not null,  
    capacity int4,  
    primary key (id)  
);
```

```
create table session (  
    id bigserial not null,  
    available_seats int4,  
    date date,  
    end_time time,  
    session_movie_title varchar(255),  
    start_time time,  
    movie_id int8,  
    room_id int8,  
    primary key (id)  
);
```

```
create table ticket (  
    id bigserial not null,  
    is_credit_card boolean not null,  
    seat_number int4,  
    ticket_type int4,  
    session_id int8,  
    primary key (id)  
);
```

```
create table session_tickets (  
    session_id int8 not null,  
    tickets_id int8 not null  
);  
alter table session_tickets  
add constraint UK_l3xo4milgpredg1tmi2sb92dp unique (tickets_id);
```


Manipulação das tabelas


Exemplos

Para ilustrar a view abaixo o `Hibernate` executou a seguinte query:

Query:

```
select
    snack0_.id as id1_6_,
    snack0_.name as name2_6_,
    snack0_.price as price3_6_,
    snack0_.url as url4_6_
from
    snack snack0_
```

Resultado no HTML: 

Ao clicar em "Adicionar Snack" somos levados para a seguinte página: 

E o `Hibernate` para gerar essa imagem acima executou o seguinte comando:

```
Hibernate:
insert
into
    snack
    (name, price, url)
values
    (?, ?, ?)
-- Esses são os valores que foram passados pelo usuário
-- Através de um POST para o endpoint /snacks, e irá adicioná-los
-- na tabela snack.
```

Sobre

Alunos e matrículas

- Gabriel De Lucca Vieira - **20180042775**
- Herbert Gomes Mendes - **20200109822**
- Marcos Dantas Guimarães - **20200133309**

O que é o projeto?

O projeto é um sistema para o gerenciamento de um cinema.

O que o sistema faz?

O sistema permite que o usuário faça o cadastro de filmes, salas, sessões e compra de ingressos.

Quais são as entidades do sistema?

- Movie
- Actor
- Poster
- Room
- Session
- Ticket
- Snack
- SnackOrder
- Transaction

Qual a arquitetura do sistema?

O sistema é composto por uma API RESTful, que é consumida por um aplicativo web. O padrão de design seguido foi o MVC (*Model Viewer Controller*). □

Quais foram as tecnologias utilizadas?

- Java 17
- Spring Boot
- Spring Data JPA
- Hibernate
- PostgreSQL
- Maven
- HTML
- Thymeleaf