**UNIVERSIDADE DO OESTE DE SANTA CATARINA**

**CAMPUS DE SÃO MIGUEL DO OESTE**

**BACHARELADO EM CIÊNCIAS DA COMPUTAÇÃO**

EVERTO CARLOS HOLLEWEIGER

GUILHERME ZENIN

**DATACINE: GESTÃO PARA CINEMAS**

São Miguel Do Oeste/SC

2022

EVERTO CARLOS HOLLEWEIGER

GUILHERME ZENIN

DATACINE: GESTÃO PARA CINEMAS

Trabalho Acadêmico dos componentes de Banco de Dados II, Engenharia de Software I e Programação II do curso de Ciências da Computação da Universidade do Oeste de Santa Catarina no campus de São Miguel do Oeste.

Professores: Roberson Junior Fernandes Alves, Franciele Carla Petry e Otilia Donato Barbosa.

São Miguel Do Oeste/SC

2022

**LISTA DE ILUSTRAÇÕES**

**FIGURAS**

[Figura 1: Diagrama relacional proposto para o DataCine 8](#_Toc107246462)

[Figura 2: Script utilizado para criação do banco de dados. 9](#_Toc107246463)

[Figura 3: Script utilizado para criação da tabela filme\_premiacao. 10](#_Toc107246464)

[Figura 4: Script utilizado para criação da tabela filme\_premiacao. 10](#_Toc107246465)

[Figura 5: Informações inseridas na tabela cinema. 10](#_Toc107246466)

**SUMÁRIO**

[1 INTRODUÇÃO 5](#_Toc121151275)

[2 DESENVOLVIMENTO 6](#_Toc121151276)

[2.1 BANCO DE DADOS II 6](#_Toc121151277)

[**2.1.1 Contextualização e ideias iniciais** 6](#_Toc121151278)

[**2.1.2 Modelo Relacional** 7](#_Toc121151279)

[**2.1.3 *Scripts*** 9](#_Toc121151280)

[2.1.3.1 *Script* de criação do banco de dados. 9](#_Toc121151281)

[2.1.3.2 *Script* de geração das tabelas 10](#_Toc121151282)

[2.1.3.3 *Script* de inserção das tabelas 11](#_Toc121151283)

[2.1.4 **Índices (*Index*)** 11](#_Toc121151284)

[**2.1.5 Gatilhos (Triggers)** 11](#_Toc121151285)

[**2.1.6 Funções (Functions) e Procedures** 12](#_Toc121151286)

[**2.1.7 Configuração de políticas de acesso** 12](#_Toc121151287)

[**2.1.7 Políticas e configuração do backup e *restore*** 13](#_Toc121151288)

[2.1.8 **Relatórios do Modelo de Negócio** 16](#_Toc121151289)

[2.1.9 **Dicionário de Dados** 17](#_Toc121151290)

[2.2 ENGENHARIA DE SOFTWARE I 17](#_Toc121151291)

[**2.2.1 Diagrama de caso de uso** 17](#_Toc121151292)

[**2.2.2 Fluxo de caso de uso** 18](#_Toc121151293)

[**2.2.3 Diagrama de atividades** 19](#_Toc121151294)

[**2.2.4 Diagrama de sequências** 20](#_Toc121151295)

[**2.2.5 Diagrama de estados** 21](#_Toc121151296)

[2.3 PROGRAMAÇÃO II 22](#_Toc121151297)

[**2.3.1 Estruturação do software** 22](#_Toc121151298)

[**2.3.2 Index.html** 24](#_Toc121151299)

[**2.3.3 Filme.html** 25](#_Toc121151300)

[**2.3.4 Sala.html** 26](#_Toc121151301)

[**2.3.5 Sessao.html** 27](#_Toc121151302)

[**2.3.6 Ingresso.html** 27](#_Toc121151303)

[3 CONCLUSÃO 29](#_Toc121151304)

[APÊNDICE A (*Script* da criação dos índices) 30](#_Toc121151305)

[APÊNDICE B - (*Script* de criação da trigger “filme\_after\_tg”) 32](#_Toc121151306)

[APÊNDICE C - (Script de criação da trigger “disponibilidade\_before\_tg”) 33](#_Toc121151307)

[APÊNDICE D - (Script de criação da function “func\_bilheteria”) 34](#_Toc121151308)

[APÊNDICE E - (Script de criação da function “func\_qtd\_pro”) 35](#_Toc121151309)

[APÊNDICE F - (Script da criação dos privilégios de usuários) 36](#_Toc121151310)

[APÊNDICE G – (Políticas e configuração do backup e *restore*) 38](#_Toc121151311)

[APÊNDICE H – (Relatórios) 41](#_Toc121151312)

# 1 INTRODUÇÃO

O Sistema DataCine, é um projeto de *Software* desenvolvido para as empresas, cujo ramo de atividade trata-se de salas de cinema. O projeto foi desenvolvido pensando em atender cinemas de pequeno, médio e até grande porte, devido a suas inúmeras possibilidades de gestão.

A elaboração de um *Software* pode ser dividida em alguns seguimentos: o uso de metodologias conhecidas dentro da Engenharia de Software para análise de requisitos e necessidades do *Software* em si, bem como a utilização de métodos ágeis e diagramas, para que o projeto se mantenha organizado e explicativo durante todo o desenvolvimento para que chegue à conclusão conforme o planejado. A estruturação de um banco de dados de acordo com regras de negócio e por fim, o desenvolvimento do código de programação de acordo com todo o pressuposto antes realizado.

É importante ressaltar que o banco de dados e os relatórios utilizando linguagem SQL são os mesmos utilizados na fase inicial desse projeto realizado durante a 3ª fase do curso de Ciência da Computação, no decorrer da disciplina de Banco de Dados I. A partir deste modelo, foram introduzidas novas atualizações de melhorias e desenvolvimento de atividades para atender as novas requisições propostas.

Neste artigo serão apresentados, os métodos, *scripts*, diagramas e a descrição de cada fase do projeto, explicando a realização de cada tarefa. Todos os arquivos aqui mencionados poderão ser encontrados no repositório do GitHub, através do link: <https://github.com/evertoch/DataCine> .

# 2 DESENVOLVIMENTO

O desenvolvimento deste trabalho ocorreu de forma multidisciplinar, envolvendo os seguintes componentes curriculares: Banco de Dados II, Engenharia de Software I e Programação II, sendo assim, a fim de transcrever o projeto de maneira didática, o desenvolvimento teórico foi elaborado e dividido da mesma forma.

## 2.1 BANCO DE DADOS II

## **2.1.1 Contextualização e ideias iniciais**

Inicialmente, ao ser proposto a ideia de criação de um banco de dados, foi pensado em nichos pouco explorados, onde as opções de sistema fossem limitadas. Baseado nisso, surgiu a ideia do DataCine, um sistema voltado para o gerenciamento de cinemas. Com o segmento definido, foi realizado uma análise das reais necessidades que um Cinema pode possuir em sua rotina de trabalho.

Com todas as informações necessárias, foi elaborado o Modelo Relacional do banco de dados, sendo possível definir as entidades e atributos, bem como as especificações de cada um, sejam estes chave primária (*primary key*, ou simplesmente, PK), chave estrangeira (*foreign key*, ou simplesmente FK), campos não nulos (*not null),* e únicos *(unique).* Em adição, foram criados *scripts* de criação das tabelas, *inserts* de informações e o desenvolvimento de relatórios previamente requisitados.

Com base no modelo inicialmente feito, foi dado sequência com o desenvolvimento do projeto.

## **2.1.2 Modelo Relacional**

O modelo relacional representa os dados num Banco de Dados como uma coleção de tabelas (relações). Cada tabela terá um nome, que será único, e um conjunto de atributos com seus respectivos nomes e domínios. O objetivo do modelo relacional elaborado é explanação da estrutura do banco de dados em um esquema de fácil visualização, sendo uma peça fundamental para manutenções, implementações e correções que podem ocorrer no banco de dados, visto que o usuário terá a estrutura completa para eventuais consultas.

Para a nomenclatura dos componentes do diagrama, foi adotado a seguinte padronização: Tabelas principais utilizam o nome da informação que será guardada, sem utilizar letra maiúscula, acentuação ou caracteres especiais, exemplo: filme. Tabelas intermediárias (fruto da relação “N:N” ou “N:M”, lê-se muitos para muitos), utiliza o nome das duas tabelas principais separados por *underline*, exemplo: filme\_premiacao. Para as colunas, presente em cada tabela, utiliza as três primeiras letras de cada palavra, sem nenhum componente para separação, exemplo: durfil (ou seja, duração do filme).

O diagrama relacional elaborado foi construído utilizando a ferramenta Visual Paradigm e pode ser visto na figura 1 abaixo:

*Figura 1: Primeira versão modelo relacional*

Diagrama

Descrição gerada automaticamente

Fonte: Os autores (2022).

Visando uma estrutura sem redundância e com a possibilidade mínima de erros, o diagrama foi reestruturado seguindo as boas práticas propostas pela normalização das Formas Normais e implementando alterações previstas no projeto, como a adição de novos campos, alterações em campos já existentes e a adição de um controle de tickets e cardápio. Além disso, foi adicionado a tabela filme\_auditoria, responsável pelo armazenamento de logs de alterações que possam ocorrer na tabela filme, que será abordado com mais detalhes no decorrer deste trabalho.

O diagrama relacional em sua versão final elaborado pode ser visto na figura 2 abaixo e na íntegra, no diretório do projeto no GitHub (<https://github.com/evertoch/DataCine> Banco de Dados II >Mapa Relacional > DataCine.vpp).

*Figura 2: Modelo Relacional versão final.*

Diagrama

Descrição gerada automaticamente

Fonte: Os autores (2022).

## **2.1.3 *Scripts***

Com a parte teórica atualizada, foi iniciado a parte prática, ou seja, a criação do banco de dados baseado em todas as novas informações e no Modelo Relacional (MR) criado.

### 2.1.3.1 *Script* de criação do banco de dados.

O *script* de criação do banco de dados não sofreu, consistindo no seguinte padrão: create database + “nome do banco de dados”;, neste caso: create database datacine;.

A figura a seguir mostra o SQL utilizado:

Figura 3: Script utilizado para criação do banco de dados.



Fonte: Os autores (2022).

O arquivo completo poderá ser encontrado no repositório do GitHub com o nome “Create\_Database\_DataCine.sql”.

### 2.1.3.2 *Script* de geração das tabelas

Com novas informações acrescentadas no MR, foi necessário atualizar o *script* da geração das tabelas.

No padrão utilizado, o SQL é divido entra a criação da tabela, definição dos atributos, os tipos de dados de cada atributo, a chave primária da tabela e os comentários. O fim do código é reservado para geração de todas as demais *constraints,* composto, em sua grande maioria, pelas chaves estrangeiras.

A Figura 4 mostra o SQL utilizado para a criação da tabela filme\_premiacao:

Figura 4: Script utilizado para criação da tabela “filme\_premiacao”.

Texto

Descrição gerada automaticamente

Fonte: Os autores (2022).

A Figura 4 mostra o SQL utilizado para a criação das *constraints* da tabela, especificadas no final do arquivo.sql:

Figura 5: Script utilizado para criação da tabela filme\_premiacao.



Fonte: Os autores (2022).

O arquivo completo poderá ser encontrado no repositório do GitHub com o nome “Create\_Database\_Tables.sql”.

### 2.1.3.3 *Script* de inserção das tabelas

Com a estrutura do banco de dados pronta, foi realizado a inserção de dados em todas as tabelas, a fim de certificado o pleno funcionamento de toda a estrutura e para geração dos relatórios propostos no Modelo de Negócio. Para campos “auto incremento”, as informações sequenciais foram geradas pelo próprio Banco de Dados. Para os demais campos, a inserção ocorreu de forma manual com informações fictícias.

A Figura 6 mostra o SQL utilizado para a inserção na tabela cinema:

Figura 6: Script utilizado para a inserção na tabela “cinema”.



Fonte: Os autores (2022).

O arquivo completo poderá ser encontrado no repositório do GitHub com o nome “Insert\_Database\_Tables.sql”.

## 2.1.4 **Índices (*Index*)**

Por tratar-se de um banco de dados com pouco cadastro de dados, a criação de índices passa a ser apenas teórico, visto que o banco de dados não irá utilizá-los. Os índices foram adicionados em todas as chaves estrangeiras, e nas colunas que serão mais utilizadas na prática, como pode ser observado no Apêndice A (*Script* da criação dos índices). O arquivo na íntegra poderá ser encontrado no repositório do GitHub com o nome “Create\_Index.sql”.

## **2.1.5 Gatilhos (Triggers)**

As triggers são mecanismos ativados automaticamente quando uma ação pré-estabelecida acontece. As triggers podem ser utilizadas para estabelecer algumas regras de negócio, auditoria, controle das regras de integridade etc. No PostgreSQL, elas vêm acompanhadas de uma procedure ou uma function

A primeira trigger implementada é a “filme\_afeter\_tg”, que executa a function “filme\_log\_func” e é utilizada para gravar na tabela “filme\_auditoria” as informações que forem inseridas, alteradas ou excluídas da tabela “filme”. O script do gatilho pode ser visualizado no Apêndice B

A segunda trigger implementada é a disponibilidade\_before\_tg, que executa a function “verifica\_disponibilidade” e é utilizada sempre antes da venda de um ingresso para verificar se ainda há assentos vazios na sessão do filme. A busca fará uma comparação com a capacidade da sala com o total de ingressos vendidos para aquela sessão, caso o número de ingressos vendidos seja igual a capacidade da sala, o banco de dados não permitirá novas inserções na tabela de ingresso. O script do gatilho pode ser visualizado no Apêndice C.

O arquivo na íntegra contendo as duas triggers citadas, bem como seu funcionamento em detalhes, poderá ser encontrado no repositório do GitHub com o nome “Triggers.sql”.

## **2.1.6 Funções (Functions) e Procedures**

As functions ou também chamadas de procedure, são recursos muito utilizados no gerenciamento de banco de dados, pois mantém concentrada a lógica necessária para determinadas funções, tendo assim uma maior agilidade no retorno de informações importantes. Com base nas regras de negócios, foi implementado duas functions para o banco de dados.

A primeira function implementada é a “func\_bilheteria”, utilizada para calcular a bilheteria de um filme, trazendo o total de ingressos e o valor arrecadado. O parâmetro a ser passado na chamada da function deverá ser o ID do filme. O script da function pode ser visualizado no Apêndice D.

A segunda function implementada é a “func\_qtd\_pro”, utilizada para calcular a quantidade total de um determinado alimento vendido e o valor total de suas vendas (somando o valor de cada item). O parâmetro a ser passado na chamada da function deverá ser o ID do alimento. O script da function pode ser visualizado no Apêndice E.

O arquivo na íntegra contendo as duas functions citadas, bem como seu funcionamento em detalhes, poderá ser encontrado no repositório do GitHub com o nome “Procedures.sql”.

## **2.1.7 Configuração de políticas de acesso**

Com base nas possibilidades de gestão que o software DataCine oferece, foram pensadas em permissões para os usuários e suas respectivas funções. Para controle de funções, foram criados os seguintes grupos: TI, Gerente, Administrativo e Atendente.

Para o grupo TI foi concedido permissão total em todas as tabelas do banco de dados, pois este será o responsável por gerir a fim de buscar o melhor aproveitamento do sistema.

Para o grupo Gerente foram concedidas as permissões de visualizar (*select*), cadastrar (*insert*), atualizar (*update*) e excluir (*delete*), tais permissões aplicam-se em todas as tabelas do sistema, visto que o mesmo realiza a gestão da empresa e deve ter acesso em todas as áreas.

Para o grupo Administrativo, foram concedidas as permissões visualizar (*select*), cadastrar (*insert*), atualizar (*update*) e excluir (*delete*) apenas nas tabelas respectivas às suas atribuições, ou seja, filme, diretor, genero, sala, sala\_filme, premiacao, filme\_premiacao.

Para o grupo Atendente, foram concedidas as permissões visualizar (*select*), cadastrar (*insert*), atualizar (*update*) e excluir (*delete*) apenas nas tabelas respectivas às suas atribuições, ou seja, filme, ticket, ingresso.

O script foi dividido em algumas etapas, começando pela criação dos grupos com base nos setores, e atribuições das permissões para estas. Por último, foi criado um usuário para cada grupo, atribuindo login e senha. O script pode ser visualizado no Apêndice F (Script da criação dos privilégios de usuários). O arquivo na íntegra poderá ser encontrado no repositório do GitHub com o nome “Criação das políticas de acesso.sql”.

## **2.1.7 Políticas e configuração do backup e *restore***

Tratando-se da execução das rotinas de backup e *restore*, foi utilizado um *script* CMD (terminal) definido como um conjunto de comandos executados sequencialmente, ou seja, um arquivo com extensão .bat, que funciona de forma semiautomática, a fim de automatizar o processo de execução das linhas de comando. Para tornar o processo mais intuitivo e melhorar a experiência do usuário, foi criado um pequeno menu interativo onde o usuário só faz a escolha das opções e o *script* se encarrega de realizar o restante.

Figura 6: menu de interação para realização do backup ou restore

Texto

Descrição gerada automaticamente

Fonte: Os autores (2022).

Para realização do backup, o .bat irá realizar uma sequência lógica para a operação: Inicialmente, cria-se uma pasta na área de trabalho com o nome “Backup\_DataCine”, e em seguida o CMD irá se posicionar dentro dessa pasta. Com o local de backup definido, será gerado um log para armazenar a data e hora de inicialização do backup e em seguida inicializará o procedimento de backup utilizando a função “pg\_dump” do PostgreSQL. Por padrão, os arquivos de backup gerados irá possuir a seguinte nomenclatura: Backup\_datacine\_+Data Atual\_+Hora do fim do backup.

Figura 7: sequência lógica para a execução do backup.

Uma imagem contendo Interface gráfica do usuário

Descrição gerada automaticamente

Fonte: Os autores (2022).

Para realização do restore é possível utilizar o mesmo script em .bat. No menu inicial do *script,* basta escolher a opção 2. Para definir o arquivo de backup a ser restaurado, deve-se direcionar o arquivo para o diretório C:\Backup\_DataCine com o nome de backup.bak, a ideia é separar da pasta onde os backups foram gerados para evitar conflitos com a restauração de bases indevidas ou desatualizadas.

Figura 8: menu de interação para realização do restore.

Texto

Descrição gerada automaticamente

Fonte: Os autores (2022).

Com o arquivo no diretório definido, o .bat irá realizar uma sequência lógica para a operação: inicialmente será feito a exclusão de qualquer base com o mesmo nome “datacine”, a fim de evitar conflitos com os dados, utilizando a função “dropdb” do PostgreSQL. Após, a exclusão ter sido efetivada, inicia-se o processo de *restore* utilizando a função psql.exe. Ao término do processo, a base de dados estará restaurada com todas as informações existentes no momento do backup.

Figura 9: menu de interação para realização do restore.

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

Fonte: Os autores (2022).

O script pode ser visualizado no Apêndice G (Políticas e configuração do backup restore). O arquivo na íntegra poderá ser encontrado no repositório do GitHub com o nome “MENU Backup e Restore”.

Algumas boas práticas quanto às políticas de backup: Os backups do banco de dados devem

ser realizados utilizando-se as seguintes frequências temporais: I– Diária; II – Semanal;

Os backups do banco de dados devem ser resguardados sob um padrão mínimo, o qual deve observar a correlação frequência/retenção de dados estabelecida a seguir: I – Diária: 1 meses; II – Semanal: 2 meses;

## 2.1.8 **Relatórios do Modelo de Negócio**

Os relatórios propostos na fase inicial do projeto se mantiveram os mesmos, contudo, devido às alterações realizadas na estrutura do banco de dados, houve a necessidade de atualizar os relatórios. Os relatórios solicitados foram os seguintes:

* Relacionar o nome do filme, tempo de duração e o gênero de todos os filmes. Ordene o relatório do filme mais longo(tempo) para o filme mais curto;
* Relacionar o nome do filme e o nome da sala para todos os filmes. Filtre somente filmes com duração mínima de 90 minutos. Ordene o relatório de forma descendente pelo nome da sala;
* Relacionar o código do filme, nome do filme, mês da bilheteria, quantidade total de ingressos vendidos no mês para meses entre anos de 2020 e 2021. Ordene o relatório do filme com mais vendas(em termos de quantidade de ingressos) para o filme com menos vendas;
* Relacionar o código do filme, nome do filme e o total de bilheteria(valores). Filtrar somente filmes, excetuando-se dramas, lançados a partir de 2018 e com restrição de idade para maiores de 12 anos. Ordene o relatório do filme com mais bilheteria para o filme com menos bilheteria.

Para o projeto final, todos os relatórios foram transformados em *views*, e relatórios que utilizavam *subselects* tiveram sua estrutura alterada para uso de *joins,* no qual podem ser vistos no Apêndice H.

O arquivo completo com os comandos SQL utilizados para gerar os relatórios propostos, poderá ser encontrado no repositório do GitHub com o nome “Relatório.sql”.

## 2.1.9 **Dicionário de Dados**

O dicionário de dados é um arquivo contendo toda a estrutura detalhada do banco de dados, bem como os comentários e informações inseridas. O mesmo foi gerado utilizando as informações cadastradas no Visual Paradigm.

O arquivo completo com o dicionário de dados completo, poderá ser encontrado no repositório do GitHub com o nome “Dicionário de dados.pdf”.

# 2.2 ENGENHARIA DE SOFTWARE I

Para a parte de Engenharia de Software I, foram desenvolvidos alguns diagramas no intuito de auxiliar no processo de desenvolvimento do projeto, bem como compreender de forma didática o funcionamento de determinados processos.

## **2.2.1 Diagrama de caso de uso**

O diagrama de caso de uso dá uma visão geral do relacionamento entre casos de uso, usuários do sistema (também conhecidos como atores) e as interações deles com o sistema. No diagrama representado na figura 09, os atores são: Atende, Pipoqueiro, Segurança Entrada, RH, Supervisor, Comercial e Administrativo. Cada elipse representa uma ação que pode estar relacionada a um ou mais atores. As setas que possuem E*xtends*, significam que existe a opção de realizar uma ação, e a seta *Include*, indica obrigatoriedade de ação.

O arquivo com o diagrama na íntegra poderá ser encontrado no repositório do GitHub com o nome “CDU\_Cinema.vpp”.

Figura 9: Diagrama de caso de uso

Diagrama

Descrição gerada automaticamente

Fonte: Os autores (2022).

## **2.2.2 Fluxo de caso de uso**

Trata-se de um fluxo de evento que descreve cada fase do caso de uso. Compõe-se por um fluxo principal e eventuais fluxos alternativos. No fluxo de caso de uso desenvolvido, ocorre o detalhamento de processo de emissão de ingressos, onde por caminhos alternativos existem a não disponibilidade de ingressos para a sessão. A figura 10 é o fluxo de caso utilizado.

O arquivo com fluxo de caso de uso na íntegra poderá ser encontrado no repositório do GitHub com o nome “Fluxo de caso de uso.docx”.

Figura 10: Fluxo de caso de uso

Texto

Descrição gerada automaticamente

Fonte: Os autores (2022).

## **2.2.3 Diagrama de atividades**

Representa o fluxo de tarefas executadas por algum ator. Fornece uma visualização do comportamento de um sistema descrevendo a sequência de ações em um processo. Os diagramas de atividades são semelhantes a fluxogramas porque mostram o fluxo entre as ações em uma atividade.

No diagrama de atividades representado na figura 11, é representado o processo de compra de ingresso. O Cliente inicia o fluxo escolhendo um filme, sua escolha é direcionada ao atendente, que verificará a disponibilidade o que gera duas possibilidades: prosseguir para a verificação de assentos ou prosseguir para verificação de pré-vendas. Caso haja assentos disponíveis, o fluxo se encerra com a emissão de tickets, caso o filme não possua pré-venda disponível, o fluxo se encerra sem a venda acontecer.

O arquivo com o diagrama na íntegra poderá ser encontrado no repositório do GitHub com o nome “Diagrama de Atividades.vpp”.

Figura 11: Diagrama de atividades

Diagrama

Descrição gerada automaticamente

Fonte: Os autores (2022).

## **2.2.4 Diagrama de sequências**

Um diagrama de sequência tem como objetivo principal representar graficamente o comportamento de uma funcionalidade, considerando a interação entre todos os componentes de software relacionados ao seu uso através de troca de mensagens.

Na representação da figura 12, é demonstrado a troca de mensagem entre o atendente e o fluxo dentro do sistema para a busca de assentos disponíveis e de como essa informação é retornada ao atendente. O mesmo fluxo acontece com a busca de tickets, contudo, utilizando mensagens diferentes.

O arquivo com o diagrama na íntegra poderá ser encontrado no repositório do GitHub com o nome “Diagrama de Sequências.vpp”.

Figura 12: Diagrama de sequências

Diagrama

Descrição gerada automaticamente com confiança média

Fonte: Os autores (2022).

## **2.2.5 Diagrama de estados**

Esse diagrama representa os estados em que um objeto pode estar e os gatilhos que promovem essa transição de um estado para outro. Na representação da figura 13, o diagrama inicia-se com o atendimento presencial pelo funcionário e passa pelo processo de “Verificar filmes em sessão” e “Verificar disponibilidade de assentos”, caso neste último não houver disponibilidade, o fluxo segue para finalizar o atendimento com o cancelamento do ingresso, caso houver disponibilidade, o fluxo segue para a conclusão da venda e o atendimento sendo finalizado.

O arquivo com o diagrama na íntegra poderá ser encontrado no repositório do GitHub com o nome “Diagrama de Estados.vpp”.

Figura 13: Diagrama de estados

Diagrama

Descrição gerada automaticamente

Fonte: Os autores (2022).

# 2.3 PROGRAMAÇÃO II

Para a parte de Programação II foi realizado o desenvolvimento do projeto em linguagem Java utilizando o software Eclipse IDE para estruturação e desenvolvimento do código e utilizando de tecnologias como JavaScript e CSS para realizar a implementação de funções e estilos. Para implementação de métodos e otimização de tempo, foi utilizado o framework Spring Boot. Como banco de dados para armazenamento das informações, foi optado pelo PostgreSQL.

## **2.3.1 Estruturação do software**

Por tratar-se de um modelo de software apenas teórico, o Modelo Relacional foi simplificado para atender apenas à parte demonstrativa de funcionamento do fluxo de maior importância do projeto: gestão de filmes, salas, sessões e vendas de ingressos, conforme expressado na figura 14.

Figura 14: Modelo Relacional do banco de dados utilizado.

Diagrama

Descrição gerada automaticamente

Fonte: Os autores (2022).

A estruturação do desenvolvimento dentro do Eclipse IDE segue um padrão:

Dentro de model existem o mapeamento das tabelas de acordo com o Modelo Relacional, ou seja, durante a execução da aplicação as tabelas irão ser criadas no banco de dados conforme tal mapeamento.

Na estrutura de *repository* estão as interfaces que fazem *extends* com JpaRepository para herdar métodos que facilitarão o desenvolvimento do projeto e para criação de métodos individuais.

Dentro de *resources* estão os arquivos HTML que dão origem as nossas telas do sistema, ou seja, cada arquivo HTML representa uma tela, que serão apresentadas ao longo do trabalho. Ainda em resources, é encontrado a pasta “assents” que contém elementos de layout, imagens, estilos e demais componentes que estruturam visualmente o projeto.

A figura 15 contém a representação da estrutura utilizada.

Figura 15: Estruturação dentro do eclipse.

Texto

Descrição gerada automaticamente

Fonte: Os autores (2022)

## **2.3.2 Index.html**

O arquivo index.html é considerado o nosso arquivo raiz que será acessado ao executar o projeto. A estrutura do arquivo foi projetada pensando em uma página inicial com um menu de navegação que fosse possível acessar as demais telas, bem como possuísse uma interface visual atrativa que trouxesse identidade ao sistema.

Figura 16: index.html

Uma imagem contendo motor

Descrição gerada automaticamente

Fonte: Os autores (2022).

## **2.3.3 Filme.html**

O arquivo filme.html é a primeira tela que pode ser acessado através do botão “Filmes” na tela inicial. É nesta tela que se torna possível o cadastro de filmes dentro do sistema através do preenchimento do formulário. O campo ID é preenchido automaticamente pelo sistema no momento do cadastro. Abaixo do formulário possui 4 botões interativos que são utilizados em todas as páginas e seguem o mesmo padrão:

Salvar: salva os dados que estão preenchidos no formulário.

Limpar: limpa o formulário deixando todos os campos vazios.

Pesquisar: abre uma tela menor que permite pesquisar os filmes já cadastrados.

Excluir: exclui o registrado inserido na tela, para a que a exclusão ocorra, a informação precisa estar previamente cadastrada no banco de dados.

A figura 17 contém a representação da tela filme.html

Figura 17: filme.html

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Fonte: Os autores (2022).

## **2.3.4 Sala.html**

O arquivo sala.html é a segunda tela que pode ser acessada, utilizando o botão “Sala” na tela inicial. É nessa tela que são realizados os cadastros das salas de exibição existentes no cinema. A tela segue o mesmo padrão das demais, utilizando o mesmo layout e os mesmos botões. O campo ID é preenchido automaticamente pelo sistema no momento do cadastro.

A figura 18 contém a representação da tela filme.html

Figura 18: sala.html

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Fonte: Os autores (2022).

## **2.3.5 Sessao.html**

O arquivo sessao.html é a terceira tela que pode ser acessada, utilizando o botão “Sessão” na tela inicial. É nessa tela em que são juntadas as informações cadastradas nas telas anteriores do sistema, ou seja, para cadastrar uma sessão, é feito a união de um filme já cadastro e direcionado para uma sala também já cadastrada, informando para o sistema em qual sala será exibido determinado filme. O cadastro de sessão só pode ser realizado caso já existe um filme e uma sala cadastrados, pois os campos “ID do Filme” e “ID da Sala” não são digitáveis.

Diferentemente das outras telas, nessa tela existe dois botões a mais, porém ambos com a mesma função, ou seja, o botão “Buscar” servirá para buscar a informação a ser preenchida no campo indicado.

Figura 19: sessao.html

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Fonte: Os autores (2022).

## **2.3.6 Ingresso.html**

O arquivo ingresso.html é a quarta tela que pode ser acessada, utilizando o botão “Ingressos” na tela inicial. É nessa tela que são realizadas as vendas dos ingressos. Para que a venda aconteça, o sistema precisará que seja definido a Sessão e seu respectivo valor de ingresso. Como o cadastro de Sessão unifica o filme cadastrado com a sala cadastrada, basta utilizar a informação da sessão que podemos ter a informação dos outros dois cadastros também. A figura 20 representa a tela de ingressos.

Diferentemente das outras telas, aqui não há nenhum registro de “nome”, ou seja, em Pesquisar apenas é possível “listar todos” sem a opção de filtros. Ademais, ao clicar no botão salvar, além da informação ser enviada ao banco de dados, é aberta a tela de impressão do navegador, permitindo imprimir as informações do formulário, ou seja, imprimir o ingresso. Por tratar-se de um sistema teórico e não comercial, não foi trabalhado um layout para os ingressos impressos. A figura 21 representa a tela de impressão do ingresso.

Figura 20: ingresso.html

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Fonte: Os autores (2022).

Figura 21: Impressão do ingresso.

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

Fonte: Os autores (2022).

# 3 CONCLUSÃO

Com base nas informações apresentadas neste trabalho, conclui-se ser de suma importância o planejamento antecipado de como proceder antes e durante o desenvolvimento de um Software, sendo fulcral ter atenção em todas as etapas para garantirmos a melhor entrega possível do sistema sem problemas estruturais. É visto também que a documentação das informações, seja ela através de comentários ou do dicionário de dados é fundamental para um bom controle das informações a serem desenvolvidas, além de facilitar os processos de criação e manutenção do sistema em um todo, sendo a parte de banco de dados ou de código do sistema.

# APÊNDICE A (*Script* da criação dos índices)

-- Criação dos índices chaves estrangeiras

create index fun\_codcin\_sk

on funcionario(codcin);

create index fil\_idgen\_sk

on filme(idgen);

create index fil\_iddir\_sk

on filme(iddir);

create index ing\_id\_sal\_fil\_sk

on ingresso(id\_sal\_fil);

create index hor\_fun\_idfuncao\_sk

on horario\_funcionario(idfuncao);

create index hor\_fun\_idhor\_sk

on horario\_funcionario(idhor);

create index sal\_fil\_idfilm\_sk

on sala\_filme(idfilm);

create index sal\_fil\_idsala\_sk

on sala\_filme(idsala);

create index sal\_fil\_idhor\_sk

on sala\_filme(idhor);

create index fil\_pre\_idfilm\_sk

on filme\_premiacao(idfilm);

create index fil\_pre\_idpre\_sk

on filme\_premiacao(idpre);

create index tic\_idfilm\_sk

on ticket(idfilm);

create index tic\_idali\_sk

on ticket(idali);

-- Criação dos índices colunas mais usadas

create index fil\_nombr\_sk

on filme(nombr);

create index fil\_nomen\_sk

on filme(nomen);

create index ing\_valing\_sk

on ingresso(valing);

create index funcao\_nomfun\_sk

on funcao(nomfun);

create index funcionario\_nomfun\_sk

on funcionario(nomfun);

# APÊNDICE B - (*Script* de criação da trigger “filme\_after\_tg”)

-- Criação da Procedure:

create or replace function filme\_log\_func()

returns trigger

as

$$

Begin

insert into filme\_auditoria (id\_filme\_new, id\_filme\_old, dat\_alt, usu\_alt, dml)

values (new.idfilm, old.idfilm, current\_timestamp, current\_user, TG\_OP);

return new;

END

$$

Language plpgsql;

-- Criação da Trigger:

create trigger filme\_afeter\_tg

after insert or update or delete

on filme

for each row

execute procedure filme\_log\_func();

# APÊNDICE C - (Script de criação da trigger “disponibilidade\_before\_tg”)

create or replace function verifica\_disponibilidade()

returns trigger

as

$body$

declare

conta numeric;

capacidade numeric;

begin

select count(\*) into conta from ingresso i

inner join sala\_filme sf on i.id\_sal\_fil = sf.id\_sal\_fil

inner join sala s on sf.idsala = s.idsal

where i.id\_sal\_fil = new.id\_sal\_fil;

select capsal into capacidade from sala s

inner join sala\_filme sf on s.idsal = sf.idsala

inner join ingresso i on sf.id\_sal\_fil = i.id\_sal\_fil

where i.id\_sal\_fil = new.id\_sal\_fil

group by capsal;

if (conta >= capacidade) then

raise exception 'Não há mais assentos disponíveis para a sessão atual';

end if;

return new;

end

$body$

language plpgsql;

-- Criação da Trigger:

create trigger disponibilidade\_before\_tg

before insert on ingresso

for each row

execute procedure verifica\_disponibilidade();

# APÊNDICE D - (Script de criação da function “func\_bilheteria”)

-- Criação da Procedure:

create or replace function func\_bilheteria(id\_film integer)

returns SETOF record as

$body$

declare

begin

return query

select id\_film, count(\*), sum(valing) from ingresso i

inner join sala\_filme sf on i.id\_sal\_fil = sf.id\_sal\_fil

inner join filme f on sf.idfilm = f.idfilm

where f.idfilm = id\_film;

return;

end;

$body$

language plpgsql;

-- Chamada da procedure com o parâmetro (id\_fil).

select \* from func\_bilheteria(2) as (Id\_filme integer, ingressos\_vendidos int8, valor\_total numeric);

# APÊNDICE E - (Script de criação da function “func\_qtd\_pro”)

-- Criação da Procedure:

create or replace function func\_qtd\_pro(id\_alimento integer)

returns SETOF record as

$body$

declare

begin

return query

select desali, count(valali), sum(valali) from ticket t

inner join cardapio c ON t.idali = c.idali

where c.idali = id\_alimento

group by desali;

return;

end;

$body$

language plpgsql;

-- Chamada da Procedure:

select \* from func\_qtd\_pro(1) as (Descricao varchar, Quantidade int8, valor\_total int8);

# APÊNDICE F - (Script da criação dos privilégios de usuários)

-- Criação dos grupos.

create group TI;

create group Gerente;

create group Administrativo;

create group Atendente;

-- Criação dos usuários.

create user everto

with

superuser

password 'everto123';

create user gerente1

with password 'gerente123';

create user administrativo1

with password 'administrativo123';

create user atendente1

with password 'atendente123';

-- Adicionando os usuários aos grupos;

grant ti to everto;

grant gerente to gerente1;

grant administrativo to administrativo1;

grant atendente to atendente1;

-- Conceções de permissões ao grupo "TI".

-- ao grupo de "TI" é concedido total privilégio em todas as tabelas.

grant all privileges

on all tables in schema public

to ti;

-- Conceções de permissões ao grupo "Gerente".

-- Ao grupo de "gerente" são concedidos apenas as permissões permitindo-o "mexer" de forma básica no banco, mas em todas as tabelas.

grant select, insert, delete, update

on all tables in schema public

to gerente;

-- Conceções de permissões ao grupo "Administrativo".

-- Ao grupo de "Administrativo" são concedidos algumas permissões básicas, mas tabelas pertinentes apenas a sua devida função.

grant select, insert, delete, update

on filme, diretor, genero, sala, sala\_filme, premiacao, filme\_premiacao

to administrativo;

-- Conceções de permissões ao grupo "atendente".

-- Ao grupo de "atendente" são concedidos as mesmas permissões que o grupo "administrativo", mas apenas nas tabelas pertinentes apenas a sua devida função.

grant select, insert, delete, update

on filme, ticket, ingresso

to atendente;

# APÊNDICE G – (Políticas e configuração do backup e *restore*)

@ECHO off

TITLE DataCine

cls

color 0b

:MenuInicial

cls

ECHO.

ECHO ---------------------------------------

ECHO -- \* MENU BACKUP/RESTORE DATACINE \* --

ECHO -- \* ESCOLHA UMA OPCAO ABAIXO \* --

ECHO ---------------------------------------

ECHO.

ECHO [1] BACKUP

ECHO [2] RESTORE

ECHO [X] Sair

ECHO.

set /p MenuInicial=

if '%MenuInicial%'=='1' goto MenuInicial\_backup

if '%MenuInicial%'=='2' goto MenuInicial\_restore

if '%MenuInicial%'=='x' goto sair\_MenuInicial

if '%MenuInicial%'=='X' goto sair\_MenuInicial

if not %MenuInicial%=="1,2,3,X" (

ECHO Opcao [%MenuInicial%] invalida.

ECHO Verifique o menu acima.

goto MenuInicial

)

:MenuInicial\_backup

MKDIR "%USERPROFILE%"\Desktop\Backup\_DataCine

cd "%USERPROFILE%"\Desktop\Backup\_DataCine

ECHO %date:~6,4%.%date:~3,2%.%date:~0,2% - %time:~0,2%:%time:~-8,2%: BACKUP iniciado >> sgbd.log

ECHO Em execucao...

"c:\Program Files\PostgreSQL\14\bin\pg\_dump.exe" -Fc -U postgres -d datacine -C -Fp -f backup.bak >> sgbd.log

ren ""%USERPROFILE%"\Desktop\Backup\_DataCine\backup.bak" "Backup\_datacine\_%date:~6,4%.%date:~3,2%.%date:~0,2%\_%time:~0,2%%time:~-8,2%.bak" >> sgbd.log

ECHO %date:~6,4%.%date:~3,2%.%date:~0,2% - %time:~0,2%:%time:~-8,2%: BACKUP concluido >> sgbd.log

goto MenuInicial

:MenuInicial\_restore

cls

ECHO O nome da base restaurada deve ser "Backup.bak". >> sgbd.log

:Menu\_restore

ECHO.

color 4

ECHO --------------------------------------------------------

ECHO -- \* LEIA COM ATENCAO! \* --

timeout 4 > NUL

color 0B

ECHO -- \* O NOME DO ARQUIVO DEVE SER backup.bak \* --

ECHO -- \* O ARQUIVO DEVE ESTAR EM C:\Backup\_DataCine \* --

ECHO --------------------------------------------------------

ECHO.

ECHO [1] PROSSEGUIR

ECHO [X] SAIR

ECHO.

set /p Menu\_restore=

if '%Menu\_restore%'=='1' goto Restore\_prosseguir

if '%Menu\_restore%'=='x' goto MenuInicial

if '%Menu\_restore%'=='X' goto MenuInicial

:Restore\_prosseguir

cd C:\Backup\_DataCine

ECHO.

ECHO Em execucao o processo de restauracao...

ECHO.

ECHO %date:~6,4%.%date:~3,2%.%date:~0,2% - %time:~0,2%:%time:~-8,2%: RESTORE iniciado >> sgbd.log

ECHO Informe a senha para apagar a base de dados existente:

"c:\Program Files\PostgreSQL\14\bin\dropdb.exe" -U postgres datacine

ECHO.

ECHO Informe a senha para restaurar o backup...

"c:\Program Files\PostgreSQL\14\bin\psql.exe" -U postgres -d postgres < "backup.bak" >> sgbd.log

ECHO %date:~6,4%.%date:~3,2%.%date:~0,2% - %time:~0,2%:%time:~-8,2%: RESTORE concluido >> sgbd.log

ECHO %date:~6,4%.%date:~3,2%.%date:~0,2% - %time:~0,2%:%time:~-8,2%: Base restaurada >> sgbd.log

goto MenuInicial

# APÊNDICE H – (Relatórios)

-- 1

create view vw\_filme\_tempo as

select f.nomBR "Nome em português (PT-BR)", f.nomEN "Nome em Ingês (EN-US)", f.durfil "Tempo de duração", g.nomgen "Gênero do filme"

from filme f

inner join genero g on f.idgen = g.idgen

order by durfil desc;

select \* from vw\_filme\_tempo;

-- 2

create view vw\_filme\_sala as

select f.nomBR "Nome do filme (PT-BR)", f.nomEN "Nome do filme (EN-US)", s.nomsal "Sala de exibição"

from sala\_filme sf

inner join filme f on sf.idfilm = f.idfilm

inner join sala s on sf.idsala = s.idsal

where f.durfil > '1:30'

order by s.nomsal desc;

select \* from vw\_filme\_sala;

-- 3

create view vw\_filme\_ingressos as

select sf.idfilm "Código do filme", f.nomBR "Nome do filme (PT-BR)", f.nomen "Nome do filme (EN-US)", f.mesexi "Mês da bilheteria", extract(month from i.datven) "Mês da venda", extract(year from i.datven) "Ano da venda", count(i.id\_sal\_fil) "Quantidade de ingressos vendidos"

from sala\_filme sf

inner join filme f on f.idfilm = sf.idfilm

inner join ingresso i on sf.id\_sal\_fil = i.id\_sal\_fil

where extract(year from i.datven) between '2020' and '2021'

group by sf.idfilm, i.datven, f.nombr, f.mesexi, f.nomen

order by "Quantidade de ingressos vendidos" desc;

select \* from vw\_filme\_ingressos;

-- 4

create view vw\_filme\_valor as

select f.idfilm "Código do filme", f.nomBR "Nome do filme (PT-BR)", f.nomen "Nome do filme (EN-US)", count(i.id\_sal\_fil) "Quantidade de ingressos vendidos", sum(i.valing) "Valor total R$"

from filme f

inner join sala\_filme sf on f.idfilm = sf.idfilm

inner join ingresso i on sf.id\_sal\_fil = i.id\_sal\_fil

inner join genero g on f.idgen = g.idgen

where f.anolanfan >= 2018

and f.claind >= 12

and g.nomgen not in ('Drama')

group by f.idfilm, f.nomBR, f.nomen

order by "Valor total R$" desc;

select \* from vw\_filme\_valor;