TÉCNICAS DE PROGRAMAÇÃO 1
Trabalho 1

Maria Julia Dias Lima, 17/0151140
Emanoel Johannes Cardim Lazaro, 17/0140997

[1]Dep. Ciência da Computação – Universidade de Brasília (UnB)
CiC 117889 - Técnicas de Programação 1

majuhdl@gmail.com, emanoeljohannes@hotmail.com

1. Introdução

O seguinte trabalho tem como objetivo o desenvolvimento de um software, com descrição da modelação final, baseada em algum tipo de arquitetura de software. Foi escolhido pelo grupo o modelo Model-View-Controller, sobre o qual foi feito o seminário para a disciplina, sendo essa implementação um modo de uso da arquitetura.

Foi implementado também o uso de banco de dados para armazenamento e persitência dos dados obtidos e usados na interface com o usuário. O modelo de banco de dados escolhido foi o MySQL, a pedido do professor.

2. Implementação-Parte II

2.1. Apresentação das Partes do Projeto

Requisitos gerais: Um cinema pode ter muitas salas, sendo necessário, por tanto, registrar informações a respeito de cada uma, como sua capacidade, ou seja, o numero de assentos disponíveis. O cinema apresenta muitos filmes. Um filme tem informações, titulo e duração. Assim, sempre que um filme for ser apresentado, deve-se registrá-lo também. Um mesmo filme pode ser apresentado em diferentes salas e em horários diferentes. Cada apresentação em uma determinada sala e horário é chamada sessão. Um filme sendo apresentado em uma sessão tem um conjunto máximo de ingressos, determinado pela capacidade da sala. Os clientes do cinema podem comprar ou não ingressos para assistir a uma sessão. O funcionário deve intermediar a compra do ingresso. Além disso, um cliente só pode comprar ingressos para sessões ainda não encerradas.

2.2. Apresentação do Funcionamento

O programa desenvolvido foi baseado composto de classes e suas instanciações, de acordo com o desenvolvimento de programação Orientada a Objetos. Foi escolhida a linguagem javascript, pela facilidade de conexão com o banco de dados MySQL e da criação de uma interface gráfica funcional e de boa apresentação, para fácil uso por parte

do usuário. A facilidade de interação do usuário sem compreensão do que o programa faz além do uso da interface é também uma das premissas do MVC, modelo de arquitetura utilizado no trabalho.
O MVC divide o projeto em 3 partes: view, já explicado; model, que faz operações referentes ao banco de dados; e o controller, que faz a interação entre a model, e a viewer, coordenando as operações da model.

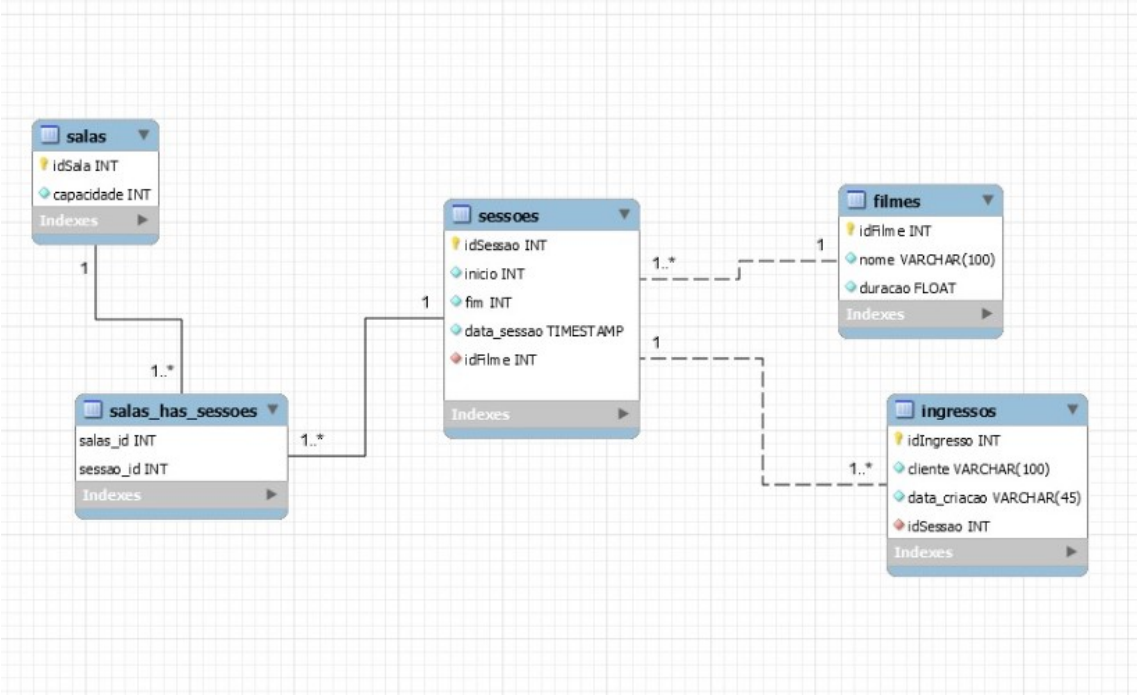

Figure 1. Diagrama de classes final do projeto escolhido,gerado pelo MySQL

O diagrama acima exibe as classes do programa, onde temos que as classes de salas e sessoes se associam de muitos para muitos. Diante disso, o banco de dados gerou uma classe intermediária, para fazer o intermédio de um sala poderem ter muitas sessoes, e que sessoes em um determinado horário poder ter várias salas. As sessoes tem uma chave externa para conexão com a classe de filmes, como forma de relacionar as dias, já que toda sessão deve ter um filme associado. O mesmo acontece entre ingressos e sessoes, o primeiro possuindo uma chave externa para uma sessao, pois deve ter uma sessao associada a ele obrigatóriamente.

3. Código e implementação

Nosso programa tem a perspectiva do funcionário, este que é possível adicionar e remover filmes, salas e sessões, e gerar ingressos paras as sessões existentes, e a cada ingresso gerado para uma sessão específica, a quantidade de ingressos disponíveis diminui, e essa quantia é baseada no número de assentos disponíveis na sala.

Ao remover uma sala, ou um filme, todas as sessões vinculadas são excluidas. Por exemplo, se há duas sessões na sala número 2, ao excluirmos esta sala, as sessões deixaram de existir também.

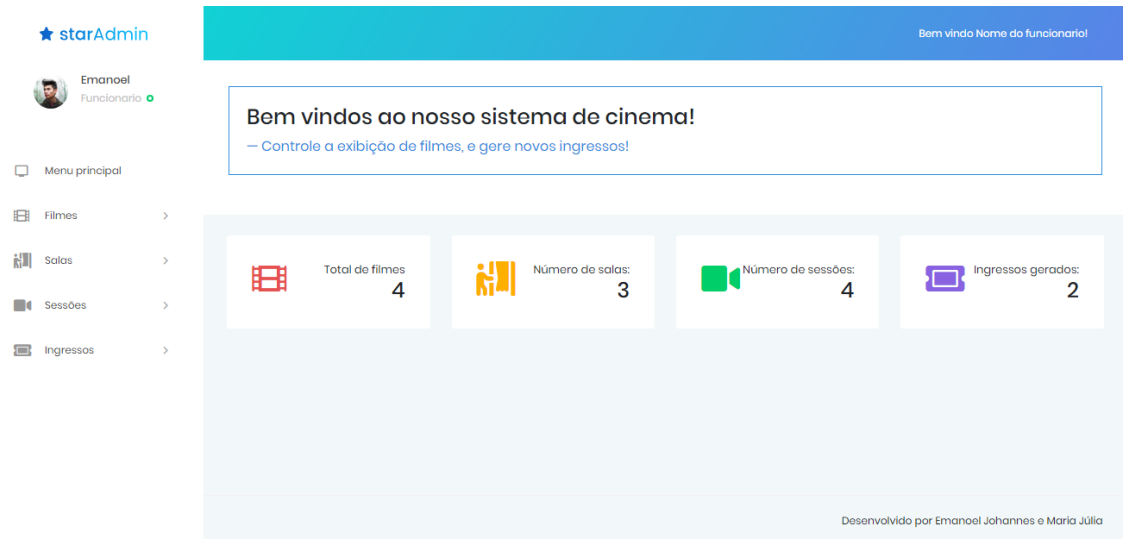

Figure 2. Página inicial da aplicação

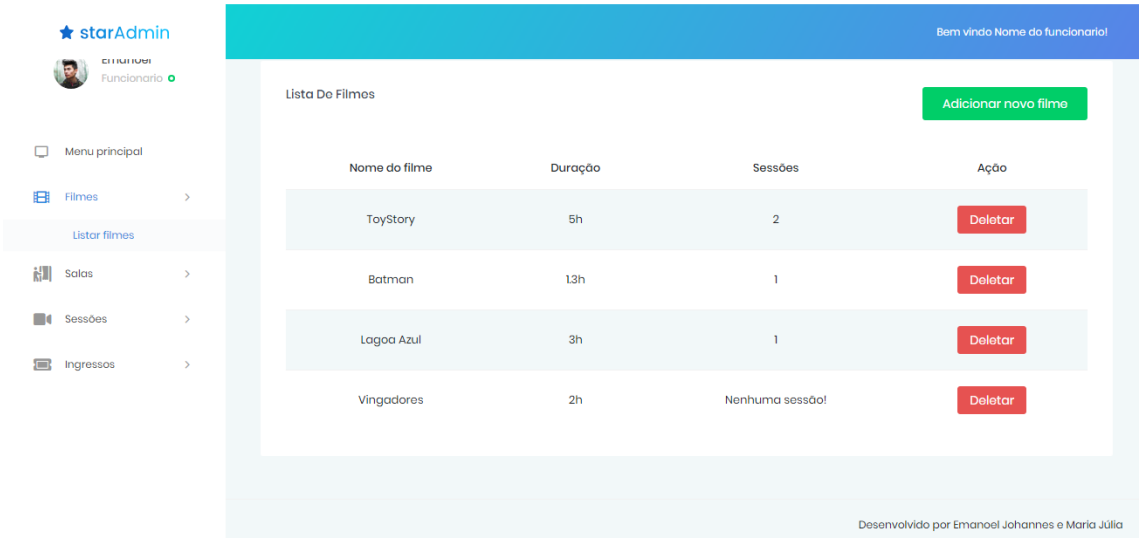

Figure 3. Listagem dos filmes, onde podemos adicionar, excluir e verificar quantas sessões temos para cada filme

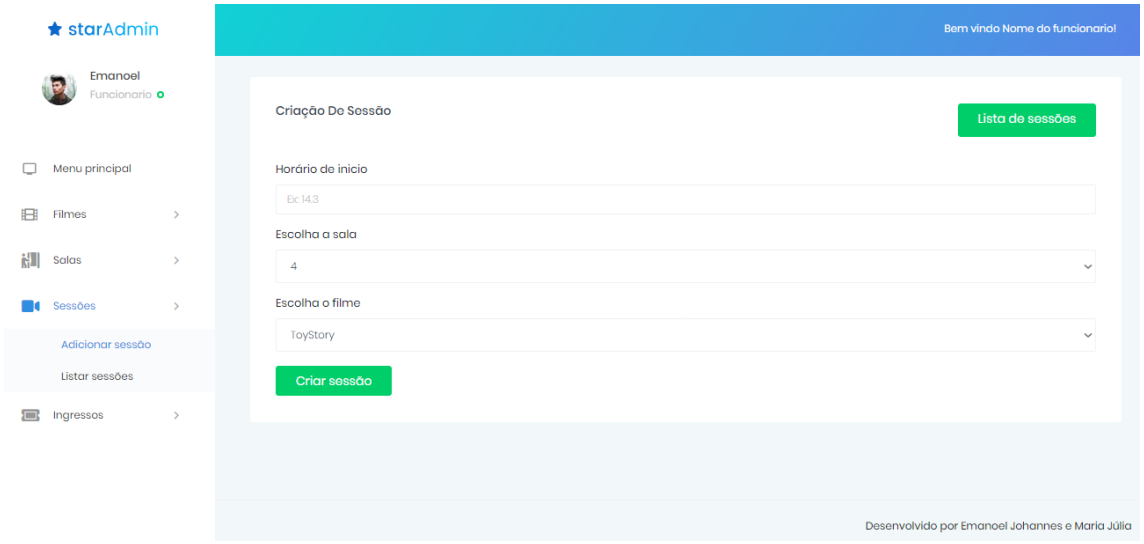

Figure 4. Tela exemplo de criação de sessões, onde definimos um horário de ínicio, a sala, e o filme a ser exibido, com todos os dados vindo do banco de dados

Para exemplificar a estrutura do código, deixarei abaixo a imagem do conjunto de pastas. Dentro de nossa pasta 'src', temos o código em sí, onde será dividido as models, views e controllers, além de um arquivo chamado server.js, que é responsável por fazer nossa aplicação ser ouvida pela porta 3333 de nosso navegador, além de unir todos os módulos. Temos também o arquivo de rotas 'routes.js', onde teremos tanto nossas rotas gets, quanto rotas post (que serão usadas para adição de novos filmes, salas ou sessões).

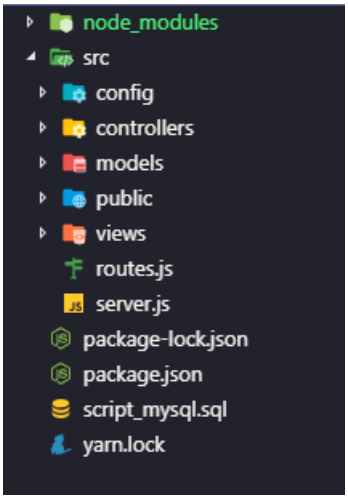

Figure 5. Conjunto de pastas de nossa aplicação

Na pasta node-modules, temos todos os módulos instalados que fazem nossa aplicação compilar, e outros módulos que nos ajudam, como por exemplo, o knex, que usamos para fazer as operações no banco de dados. Já em 'scriptmysql.sql' temos o script de nosso banco de dados, para que seja possível qualquer um ter o banco de dados correto para executar a aplicação.

4. Instruções de compilação

Para começar, temos que ter o NodeJs e o MySQL instalados em nosso computador, que podem ser baixados em seus sites oficiais gratuitamente, e para checar se o NodeJs está instalado, basta executar em seu terminal o comando: 'node -v', que deverá retornar a versão instalada. Uma vez que terminamos de instalar as dependências citadas, abrimos o prompt de comandos e navegamos até a pasta do projeto, e finalizamos executando: 'node scr/server.js', e o programa pode ser conferido em seu navegador, acessando http://localhost:3333. Não esquecer de compilar o script feito em 'scriptmysql.sql' em seu MySQL, este que irá instalar o banco de dados necessário para a aplicação.

5. Conclusão

Foi possível concluir o trabalho de acordo com o requisitado no relatório e nas aulas. Conseguimos implementar e aplicar na prática todos os conceitos mais importantes e demonstrar compreensão e entendimento de todos os conceitos envolvidos e da linguagem de desenvolvimento orientada a objetos. Conseguimos implementar a interface pedida, o tratamento de exceções necessário para tratar todo o código, como a linguagem permitia. Não foram implementados testes unitários, pois a própria interface já está bem organizada para teste do usuário, demonstrando o bom uso desse processo.