



Curso: Ciência da Computação

Campus: Ribeirão Preto

**ATIVIDADES PRÁTICAS SUPERVISIONADAS - DESENVOLVIMENTO DE UM SISTEMA
COMPUTACIONAL PARA ANÁLISE E
CLASSIFICAÇÃO DE FORMAS.**

NOME: RAPHAEL BASSI ALVES DA SILVA	RA:F119556	CC2P18
NOME: THIAGO VIANA GOMES DOS SANTOS	RA:D868ig2	CC3P18
NOME: GABRIEL CONTI	RA:N414030	CC3Q18
NOME: SÁVIO CANGUSSU	RA: T870577	CC3P18

MAIO,2020

ÍNDICE:

1. Objetivo:	3
2. Introdução:.....	4 e 5
3. Referencial Teórico:.....	6 a 8
4. Plano de Desenvolvimento da Aplicação:	9 a 16
5. Projeto:.....	17 e 18
6. Bibliografia:.....	19
7. Fichas:.....	20 e 21

OBJETIVO:

Objetivo do trabalho: o objetivo do trabalho é o desenvolvimento de um sistema computacional para análise e classificação de formas, desenvolver um sistema de locadora de DVD com cadastro dos usuários, cadastros de cliente, cadastro de categoria dos DVDs e controle de locação de vídeo e vídeo disponíveis.

INTRODUÇÃO:

Com o alto avanço da internet e com a facilidade de pirataria na própria, as locadoras perderam seus espaços físicos e virtuais, dando lugar a pirataria e ao streaming de filmes, mas como um tema que abrange desde a front-end até a back-end e banco de dados, para criar um sistema de uma locadora que contenha a disponibilidade de filmes e seu estoque.

Para começar o projeto decidimos como faríamos a front-end, e assim decidimos fazer-lo em html e css, e para ajustar o layout, utilizamos o sistema de grid do css, grid é uma malha formada pela interseção de um conjunto de linhas horizontais com um conjunto de linhas verticais, a utilização do grid nos proporciona:

- Dimensões fixos ou flexíveis
- Posicionamento de itens: podendo assim posicionar itens precisamente na página usando nomes ou fazendo referência a determinada posição da página
- Criação de grid adicionais
- Alinhamento e controle de conteúdo sobreposto.

O sistema de banco de dados utilizamos o PostgreSQL que é um sistema de gerenciador de banco de dados desenvolvido em código aberto, foi desenvolvido na universidade de Berkeley na Califórnia, tendo esse sistema inúmeras características como por exemplo:

- Integridade transacional
- Chaves estrangeiras
- Consultas complexas
- Controle de concorrência multi-versão
- Suporte ao modelo híbrido objeto-relacional
- Visões
- Indexação por texto

- Gatilhos
- Facilidade de acesso
- Linguagem Procedural em várias linguagens
- Estrutura para guardar dados Georreferenciados

Com tudo definido precisaríamos nos comunicar e comunicar nossos arquivos uns com os outros, fizemos isso utilizando o github que é uma plataforma de hospedagem de código-fonte, utilizamos ele para separar os arquivos entre nós mesmos e organizar o que era front-end, back-end e dos arquivos que compunham o site, como por exemplos imagens.

REFERENCIAL TEÓRICO:

HTML:

O HTML é uma linguagem de marcação que é utilizada para se desenvolver sites, ela surgiu junto com o HTTP, e os dois foram responsáveis pela popularização da internet.

Ela foi criada em 1991 na suíça, por Tim Berners-Lee.

A linguagem HTML foi a primeira de nível mundial, apesar de existir várias outras, ele ainda é a que prevalece no topo.

Sendo o HTML uma linguagem de marcação, isso implica que a linguagem é constituída de códigos que delimitam conteúdos específicos segundo uma sintaxe própria.

No começo o HTML foi uma linguagem bem difícil de se aprender, por conta de que ele tinha vários comandos, porém conforme o tempo foi passando e a cada nova atualização eles o deixam mais fácil de se utilizar.

E o HTML serve para se criar páginas na web, com ele podemos definir o tipo de letra, cor, espaçamento e vários outros aspectos da estrutura de um site.

Uma vantagem do HTML é você poder fazer ele com qualquer editor de texto, até mesmo um bloco de notas, basta salvar o arquivo e salvar em formato .HTML e executar.

CSS:

O 'Cascading Style Sheets' ou somente CSS foi criado a princípio em 1994 pelo Hakon Lie, ele teve o intuito de cria-lo pra facilitar para as pessoas a criação de sites, que na época era bem complicado.

Já em 1995, a W3C que é um grupo de empresas, desenvolveu o CSS1 e ela cresceu bastante entre 1997 e 1999.

O CSS é voltado para a parte de layout de um programa, ou seja, "ele serve pra controlar as opções de margem, linhas,

cores, alturas, larguras, imagens e posicionamento, sem necessidade de programar em HTML” (Yuri Pacievitch).

Java:

A linguagem Java é uma gigante na área da programação, apesar de já ser um pouco antiga ela ainda é fortemente utilizada e tem um bom campo de mercado.

Seu surgimento foi em 1991 por um grupo de colaboradores da Sun Microsystems, porém naquela época seu nome era Projeto Green.

No início eles queriam criar programas portáteis para funcionarem em diversos tipos de dispositivos, porém eles viram que essa ideia traria muito problema, então tiveram a ideia de desenvolver um sistema operacional que permitiria a utilização de seus programas pelos mais diversos tipos de equipamentos. Essa nova linguagem que surgiu com a nova ideia acabou se chamando OAK e o sistema operacional que foi formado a partir chamou-se GreenOS.

Em 1994 o projeto mudou seu nome finalmente de OAK para JAVA, porém ele ainda não tinha uma aplicação bem definida, porque não tinham um navegador que suportasse as aplicações feitas em Java, mas naquele ano isso foi criado e foi batizado de Web Runner.

Durante os anos 2000 a SM continuou lançando atualizações de melhorias gratuitas e isso fez com que o Java fosse cada vez mais ganhando um público maior. Porém em abril de 2009 a empresa Oracle comprou a Sun Microsystems e com isso era a nova dona do Java, e continua sendo a atual dona até os dias atuais.

PostgreSQL:

O PostgreSQL é um sistema de gerenciamento de banco de dados objeto-relacional (ORDBMS). Ele foi criado na University of California no departamento de tecnologia.

O Postgre é um descendente do SQL padrão que foi criado antes, também no mesmo lugar, e ele trás muito do seu antecessor porém com várias facilitações mais modernas, como um upgrade como por exemplo: “chaves-estrangeiras, functions, triggers, views, integridades transacionais, data types, funções agregadas, operadores e etc” (DevMedia).

O comando mais utilizado dele é o psql ele consiste em permitir que um usuário execute consultas no sql e visualize os seus resultados.

O servidor tem uma barreira que não deixa nenhum cliente acessar as informações do banco, e todas essas informações são armazenadas através de um diretório.

PLANO DE DESENVOLVIMENTO:

FRONT:

Como essa atividade prática supervisionada é mais complexa, ela envolve várias áreas, desde HTML básico até o desenvolvimento de um banco de dados e manipulação de seus dados.

Com o programa Apache NetBeans utilizado pra fazer o front, sendo o HTML e CSS, o conceito do projeto era desenvolver um sistema de uma locadora com um banco de dados dos filmes, divididos por categoria e com as opções de idioma e legenda disponível para o filme, também criar uma interface diferente para cliente e para funcionário da locadora, começando com o cadastro do cliente no qual envolve campos padrões de cadastro, sendo alguns obrigatórios e outros não, todos eles contendo uma regra específica pra cada, como por exemplo não poder colocar letra no campo telefone.

Outro recurso utilizado na front-end foi o grid para criar uma organização melhor na front, já que o HTML não tem uma grande variedade de recursos para organizar sua página colocando uma coisa em cima da outra, utilizamos o grid css para o fazer, já que ele faz com que a página trabalhasse em duas linhas e duas colunas na questão da organização, grid-

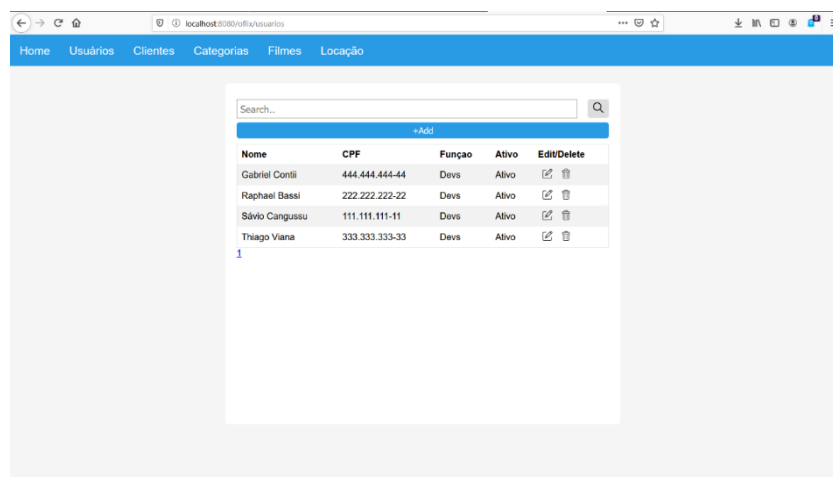
```
.container {  
  display: grid;  
  grid-template-rows: 7vh 93vh;  
  grid-template-areas: "h h"  
                      "m m"  
}
```

template-rows e grid-template-areas.

A unidade de tamanho usada é a vh, a medida do vh é igual a 1/100 de altura do viewport.

Como parte da configuração básica do site colocamos as opções para o administrador do site como deletar e editar usuários, também colocamos uma página na qual se vê detalhes desses

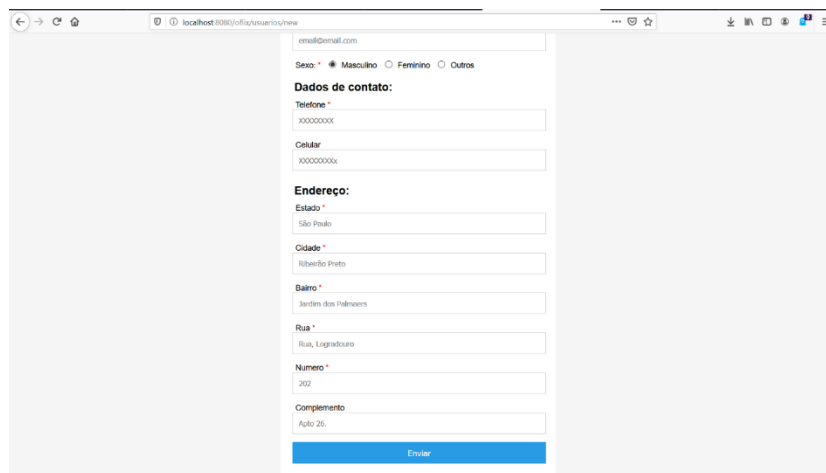
funcionários, também colocamos os botões personalizados



Na parte de cadastro do cliente, fizemos um formulário de cadastro básico, pedindo algumas informações, os campos com asterisco vermelho são obrigatórios, cada um com sua regra, como apenas números no campo de cadastro de telefone, somente letras no de nome, e assim por diante.

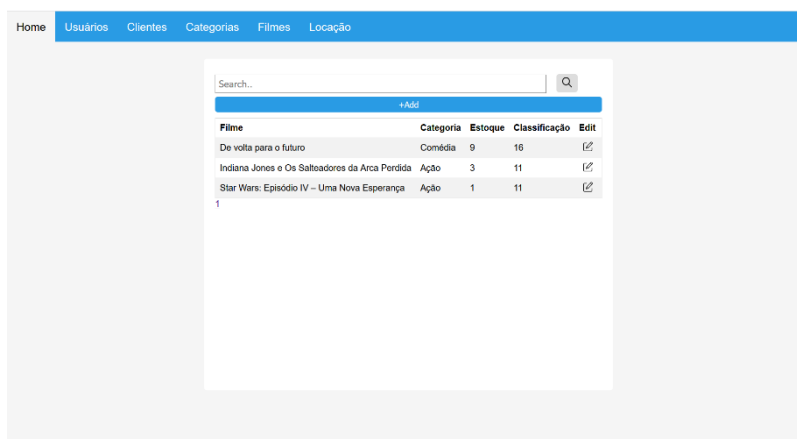
A screenshot of a web browser displaying a user registration form. The browser's address bar shows 'localhost:8080/usuarios/new'. The page has a blue navigation bar with links: Home, Usuários, Clientes, Categorias, Filmes, and Locação. The form is titled 'Dados pessoais:' and contains several input fields with red asterisks indicating required fields. The fields are: Nome (with a red asterisk), Função (with a red asterisk), CPF (with a red asterisk), Email (with a red asterisk), and Sexo (with a red asterisk). Below these are the 'Dados de contato:' fields: Telefone (with a red asterisk), Celular (with a red asterisk), and Endereço (with a red asterisk). The form also includes a 'Cidade' field with a red asterisk. The form is styled with a light gray background and a white border.

Na parte de filmes, classificamos os filmes com seus nomes, categorias, quantidade em estoque fazendo assim uma comunicação direta com o banco de dados, sua classificação indicativa, e o botão de Edit, que é disponibilizado apenas para usuários com certas permissões.



A screenshot of a web browser showing a user registration form. The browser's address bar displays 'localhost:8080/office/usuarios/new'. The form includes a 'Sexo' field with radio buttons for 'Masculino' (selected), 'Feminino', and 'Outros'. Below this is the 'Dados de contato' section with 'Telefone' and 'Celular' fields, each with a placeholder 'XXXXXXXX'. The 'Endereço' section contains 'Estado' (pre-filled with 'São Paulo'), 'Cidade' (pre-filled with 'Ribeirão Preto'), 'Bairro' (pre-filled with 'Jardim dos Palmeiras'), 'Rua' (pre-filled with 'Rua, Logradouro'), 'Número' (pre-filled with '202'), and 'Complemento' (pre-filled with 'Apto 25'). A blue 'Enviar' button is at the bottom.

Para cadastrar um filme fizemos uma página no qual solicita as informações do filme como nome, categoria, sinopse e etc. Todas essas informações são guardadas no banco de dados.



A screenshot of a web application interface for managing movies. The top navigation bar is blue and contains links: 'Home', 'Usuários', 'Clientes', 'Categorias', 'Filmes', and 'Locação'. Below the navigation bar is a search bar with the placeholder text 'Search..'. A blue button labeled '+Add' is positioned above a table. The table has five columns: 'Filme', 'Categoria', 'Estoque', 'Classificação', and 'Edit'. It contains three rows of movie data. Each row has an 'Edit' link represented by a pencil icon.

Filme	Categoria	Estoque	Classificação	Edit
De volta para o futuro	Comédia	9	16	✎
Indiana Jones e Os Saltadores da Arca Perdida	Ação	3	11	✎
Star Wars: Episódio IV – Uma Nova Esperança	Ação	1	11	✎

Os dados de locação seguem o mesmo estilo de formulário, pedindo assim o nome do cliente que está salvo no banco de dados e do filme também, e se foi devolvido ou não.

The screenshot shows a web application with a blue navigation bar containing links: Home, Usuários, Clientes, Categorias, Filmes, and Locação. The main content area is light gray. Centered is a white form titled 'Dados do filme:'. The form contains the following fields: 'Nome *' with a text input; 'Título' with a text input; 'Categoria *' with a dropdown menu; 'Sinopse' with a text area; 'Estoque *' with a text input; and 'Classificação *' with a text input. At the bottom of the form is a blue button labeled 'Enviar'.

E para adicionar novas categorias de filmes, criamos uma interface parecida com a de funcionários para editar ou ver os filmes pertencentes aquela categoria.

The screenshot shows a web application with a blue navigation bar containing links: Home, Usuários, Clientes, Categorias, Filmes, and Locação. The main content area is light gray. Centered is a white form titled 'Dados da locação:'. The form contains the following fields: 'Data de locação *' with a date input (format: dd / mm / aaaa); 'Cliente *' with a dropdown menu; 'Filmes *' with a dropdown menu; and 'Devolvido *' with radio buttons for 'Não' (selected) and 'Sim'. At the bottom of the form is a blue button labeled 'Enviar'.

The screenshot shows a web application with a blue navigation bar containing links: Home, Usuários, Clientes, Categorias, Filmes, and Locação. The main content area is light gray. Centered is a white form titled 'Categorias'. At the top of the form is a search bar with the placeholder 'Search...' and a magnifying glass icon. Below the search bar is a blue button labeled '+Add'. Below the button is a table with the following structure:

Nome da Categoria	Edit	View
Ação	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Comédia	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Drama	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Ficção científica	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Suspense	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

BACK-END:

Fora a classe DAO (responsável pela conexão com o banco de dados) back-end foi dividido em dois segmentos, *controller* e *model*;

O *controller* neste caso foi uma classe que herdou do *HttpServletRequest*, mandou pro front-end, dados úteis em forma de atributos e tratou a requisição, disponibilizando assim para suas subclasses; o *id*, a *action*, e possíveis *bad requests* que não são bem vindos nos url's.

O *id* é o que é chamado de identificação de uma propriedade do url, por exemplo, se temos no endereço, "*usuários/5*", "*5*" corresponde ao *id* de um usuário

A *action* é a ação feita para o url, podem haver actions estáticas, quanto dependentes de *id*, como por exemplo: *new*, *create* e *index* são estáticas, pois *index* é a tela inicial, onde será mostrados todos os usuários, logo, não há um *id* específico, e a *new* é responsável pela criação de um usuário ainda não criado e *create* é somente o envio do formulário com o método *POST*, logo não tem *id*, outras ações como: *edit*, *update* e *delete* necessitam de um usuário para ser deletado ou editado.

Por padrão, todas as requisições resultam em um *bad request*, logo, as subclasses devem substituir os métodos correspondentes às *actions* de acordo com a necessidade.

O *model* é uma classe que representa, em muito dos casos, uma tabela do banco, não houve a praticidade de herdar muitos métodos da superclasse *model*, por causa de atributos, logo, muitas funções que teriam que ser sobrescritas, não foram sequer escritas em primeiro lugar, mas as subclasses possuem um certo padrão de métodos, sendo eles;

- *getId()* – Função para pegar o id atual (caso não haja, retorna null)

- *update(map)* – Função que atualiza os dados do modelo a partir de um map, (usado para receber os parâmetros direto do formulário do front-end)
- *toMap()* – Função que retorna uma map para exibição do front-end nas telas de *edit*
- *valid()* – Função que retorna um booleano que diz se todos os atributos daquela classe são válidos para inserção no banco de dados através de um atributo *errors*.
- *save()* – Verifica se a classe tem seus dados validados através do método *valid()* e salva os dados no banco de dados, tanto para uma nova instância, tanto para atualizar
- *find(integer, params)* – Procura um modelo com o *id* passado por parâmetro no banco de dados, caso não encontra, retorna um novo
- *getResources(params)* – Procura no banco de dados todas as ocorrências com os parâmetros passados, alguns parâmetros são definidos automaticamente caso não sejam passados, como *page* e *limit*, que são responsáveis pela paginação dos dados no front-end através de *OFFSET* e *LIMIT* no banco de dados, outros parâmetros como *where* e *search* são opcionais, e podem ser utilizados para encontrar ocorrências mais específicas
- *validFields(String)* – esse conjunto de métodos é chamado no método *valid()* e validam os campos individuais chamando funções de validação de tipo com parâmetros na superclasse, sendo eles *validString*, *validInt* e *validSelect*, nessas funções se passam parâmetros em map para especificar o tipo de atributo que estamos lidando, por exemplo, para *validString* temos *regex*, *length*, *minimum* e *optional*, todos eles tem sua validação para uma única *String*, caso ele não cumpra algum dos atributos passados, é retornado uma chave específica no map *errors*.

Neste projeto em específico, são cinco controladores para quatro modelos, pois um dos modelos servem para mais de um

controlador (caso de usuários que podem ser clientes), os controladores, através dos modelos, fazem as verificações de erros e redirecionam o usuário para a página correta, passando também, os valores de atributos necessários para o funcionamento do front-end, como; avisar campos que devem ser marcados como errados, e valores que devem ser retornados para a página de HTML.

Não foram usadas bibliotecas externas fora drivers de conexão com o banco de dados, todas as bibliotecas são padrões do Java e até mesmo o front-end foi usado CSS e HTML puro, junto com *jsp*.

Todo o processo foi baseado no conhecimento sobre *Ruby on Rails*, por um dos desenvolvedores, muitas funções se parecem com as bibliotecas do *ActiveRecord*, principalmente de validação e atribuição no banco de dados.

BANCO DE DADOS - POSTGRESQL:

O nosso modelo de banco primeiro foi feito a partir de um DER e depois fizemos um diagrama de classes, e somente depois partimos para o código SQL em si.

Ele foi programado em 6 (seis) tabelas, sendo elas: *tb_funcao*; *tb_endereco*; *tb_pessoa*; *tb_categoria*; *tb_filme*; *tb_aluguel*.

Foi utilizado o comando SERIAL para fazer os id's de cada tabela, VARCHAR e INTEGER que são comandos padrões foram utilizados para os outros atributos, tendo a diferenciação de que alguns tiveram o comando NOT NULL para que o usuário tenha o compromisso de preencher aquele campo em seu cadastro.

A função CONSTRAINT foi utilizada para a criação das PRIMARY KEY e FOREIGN KEY das tabelas, sendo que a tabela função, endereço, e categoria tiveram apenas as chaves

primárias e as outras 3 tabelas tiveram suas chaves primárias e estrangeiras que referenciavam essas outras 3 tabelas.

PROJETO DO PROGRAMA:

Link para o código do programa no GitHub:

<https://github.com/saviu-u/oflix>

UML:

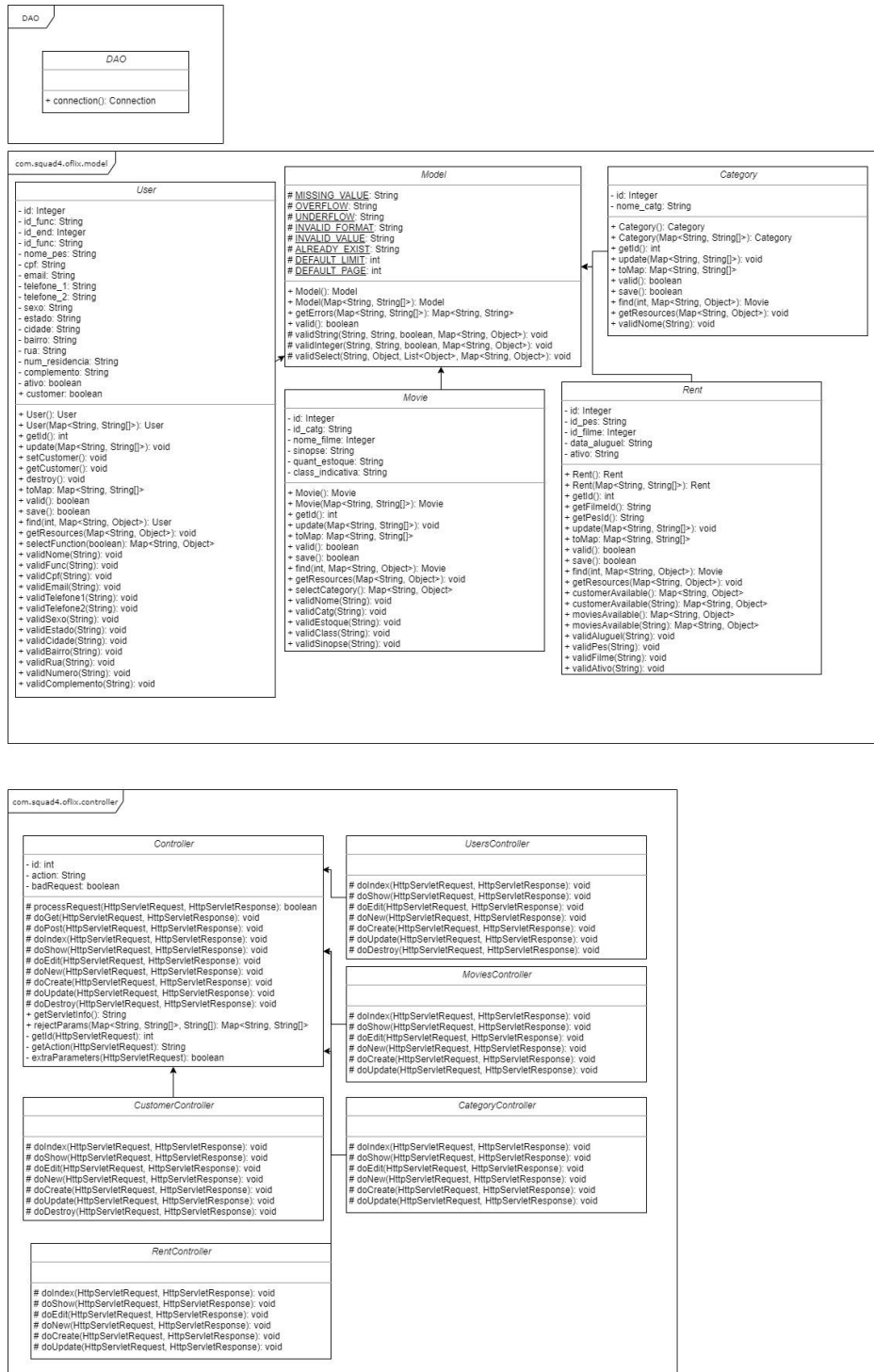
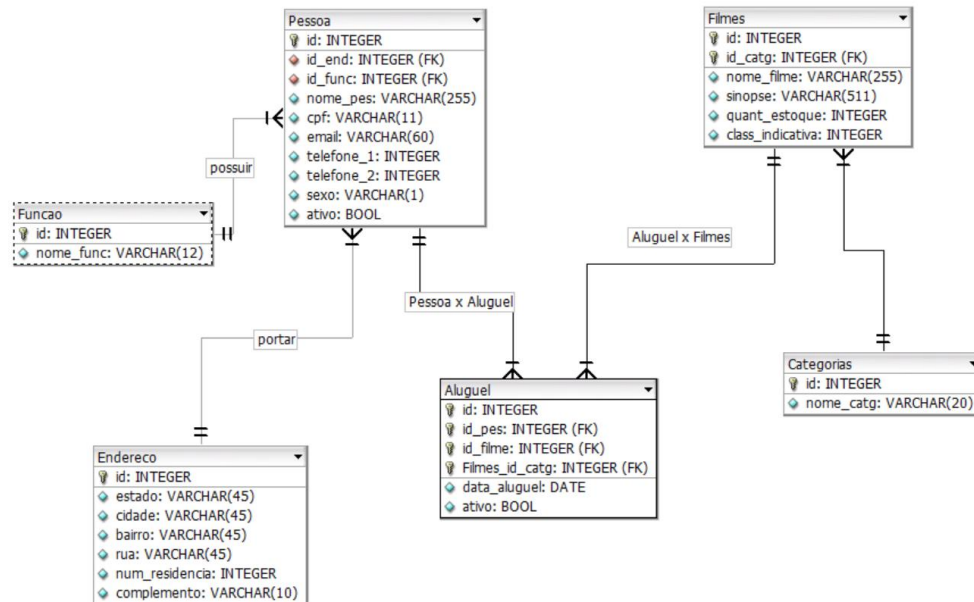
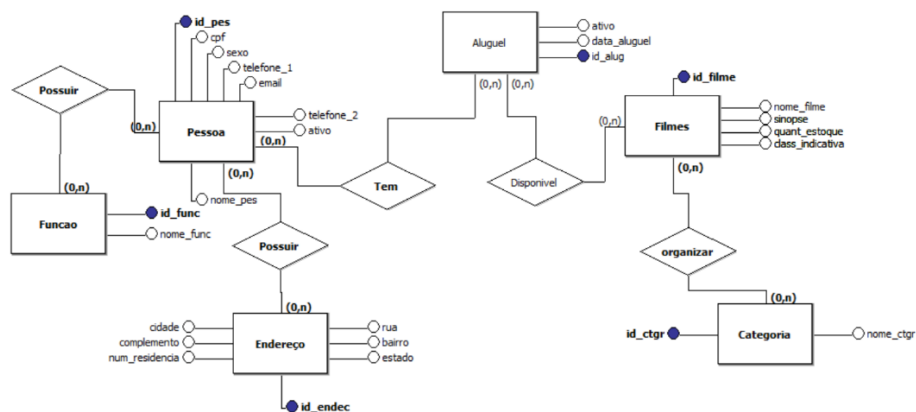


DIAGRAMA DE CLASSES:



MER LÓGICO:



BIBLIOGRAFIA:

1. <https://www.tutorialrepublic.com/snippets/preview.php?topic=bootstrap&file=crud-data-table-for-database-with-modal-form>
2. <https://www.devmedia.com.br/como-fazer-um-crud-a-partir-de-templates/33797>
3. <https://webdevacademy.com.br/ux/crud-com-bootstrap-3-parte1/>
4. <https://www.youtube.com/watch?v=RNvQzo4DoOA>
5. <https://www.w3schools.com/sql/>
6. <https://www.devmedia.com.br/servlet-tutorial/27841>
7. https://www.tutorialspoint.com/servlets/servlets_overview.htm
8. <https://pt.wikipedia.org/wiki/MVC>
9. <https://tableless.com.br/java-origem/>
10. <https://www.devmedia.com.br/java-historia-e-principais-conceitos/25178>
11. <https://www.devmedia.com.br/postgresql-tutorial/33025>

Ficha Thiago Viana:

Ficha Sávio Cangussu:

20

Ficha Raphael Bassi:

UNIP
UNIVERSIDADE PAULISTA

FICHA DAS ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

NOME: Raphael Bassi Nova da Silva TURMA: CC2P18 RA: F119556
CURSO: Engenharia de Computação CAMPUS: Ribeirão Preto SEMESTRE: 2º TURNO: Noite
CÓDIGO DA ATIVIDADE: 76B9 SEMESTRE: 23º ANO GRADE: 2020

DATA DA ATIVIDADE	DESCRIÇÃO DA ATIVIDADE	TOTAL DE HORAS	ASSINATURA DO ALUNO	HORAS ATRIBUÍDAS (1)	ASSINATURA DO PROFESSOR
09/04	Exercício sobre Java Baseado	1	Raphael Bassi	1	
09/04	Exercício JSP	1	Raphael Bassi	1	
11/04	Exercício SQL	1	Raphael Bassi	1	
14/04	Exercício de classe construída	1	Raphael Bassi	1	
16/04	Exercício de JSP	1	Raphael Bassi	1	
19/04	Exercício de manipulação de dados	1	Raphael Bassi	1	
25/04	Exercício de manipulação de dados	1	Raphael Bassi	1	
27/04	Exercício de manipulação de dados	1	Raphael Bassi	1	
03/05	Exercício de manipulação de dados	1	Raphael Bassi	1	
09/05	Exercício de manipulação de dados	1	Raphael Bassi	1	
16/05	Exercício de manipulação de dados	1	Raphael Bassi	1	
17/05	Exercício de manipulação de dados	1	Raphael Bassi	1	

(1) Horas atribuídas de acordo com o regulamento das Atividades Práticas Supervisionadas do curso.

TOTAL DE HORAS ATRIBUÍDAS: 94

AValiação: _____
Aprovado ou Reprovado: _____

NOTA: _____

DATA: 1/1/

CARIMBO E ASSINATURA DO COORDENADOR DO CURSO

Ficha Gabriel Conti:

UNIP
UNIVERSIDADE PAULISTA

FICHA DAS ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

NOME: Gabriel Tadeu Braga Conti TURMA: CC3P18 RA: N449030
CURSO: Engenharia de Computação CAMPUS: Ribeirão Preto SEMESTRE: 3º TURNO: Noite
CÓDIGO DA ATIVIDADE: 76B9 SEMESTRE: 3º ANO GRADE: 220

DATA DA ATIVIDADE	DESCRIÇÃO DA ATIVIDADE	TOTAL DE HORAS	ASSINATURA DO ALUNO	HORAS ATRIBUÍDAS (1)	ASSINATURA DO PROFESSOR
04/04	Exercício sobre Java Baseado	1,5	Gabriel Conti	1,5	
06/04	Exercício sobre Java Baseado	1,5	Gabriel Conti	1,5	
12/04	Exercício sobre Java Baseado	1,5	Gabriel Conti	1,5	
15/04	Exercício sobre Java Baseado	1,5	Gabriel Conti	1,5	
19/04	Exercício sobre Java Baseado	1,5	Gabriel Conti	1,5	
23/04	Exercício sobre Java Baseado	1,5	Gabriel Conti	1,5	
27/04	Exercício sobre Java Baseado	1,5	Gabriel Conti	1,5	
03/05	Exercício sobre Java Baseado	1,5	Gabriel Conti	1,5	
07/05	Exercício sobre Java Baseado	1,5	Gabriel Conti	1,5	

(1) Horas atribuídas de acordo com o regulamento das Atividades Práticas Supervisionadas do curso.

TOTAL DE HORAS ATRIBUÍDAS: 80

AValiação: _____
Aprovado ou Reprovado: _____

NOTA: _____

DATA: 1/1/

CARIMBO E ASSINATURA DO COORDENADOR DO CURSO