

---

**SISTEMA DE CRÍTICAS DE CINEMA**  
**Trabalho final de Linguagem de Programação 2**

*Julia Klopffleisch Schaedler<sup>1</sup>; Miguel Gomes Menna Barreto<sup>2</sup>.*

## **INTRODUÇÃO**

O seguinte trabalho tem como objetivo colocar em prática tudo que foi visto na disciplina de Linguagem de Programação 2, tendo em vista os conteúdos de php, operações com o banco de dados, utilização de frameworks, etc. O tema escolhido foi um sistema de críticas de cinema, onde o usuário pode cadastrar-se no site e registrar uma crítica ou review de um determinado filme, cujo pode ser inserido pelo usuário no site também. Além disso, será utilizado, além de html, css e php; o banco de dados Sqlite e também o modelo MVC (Model, View e Controller) para organizar todos os componentes do projeto.

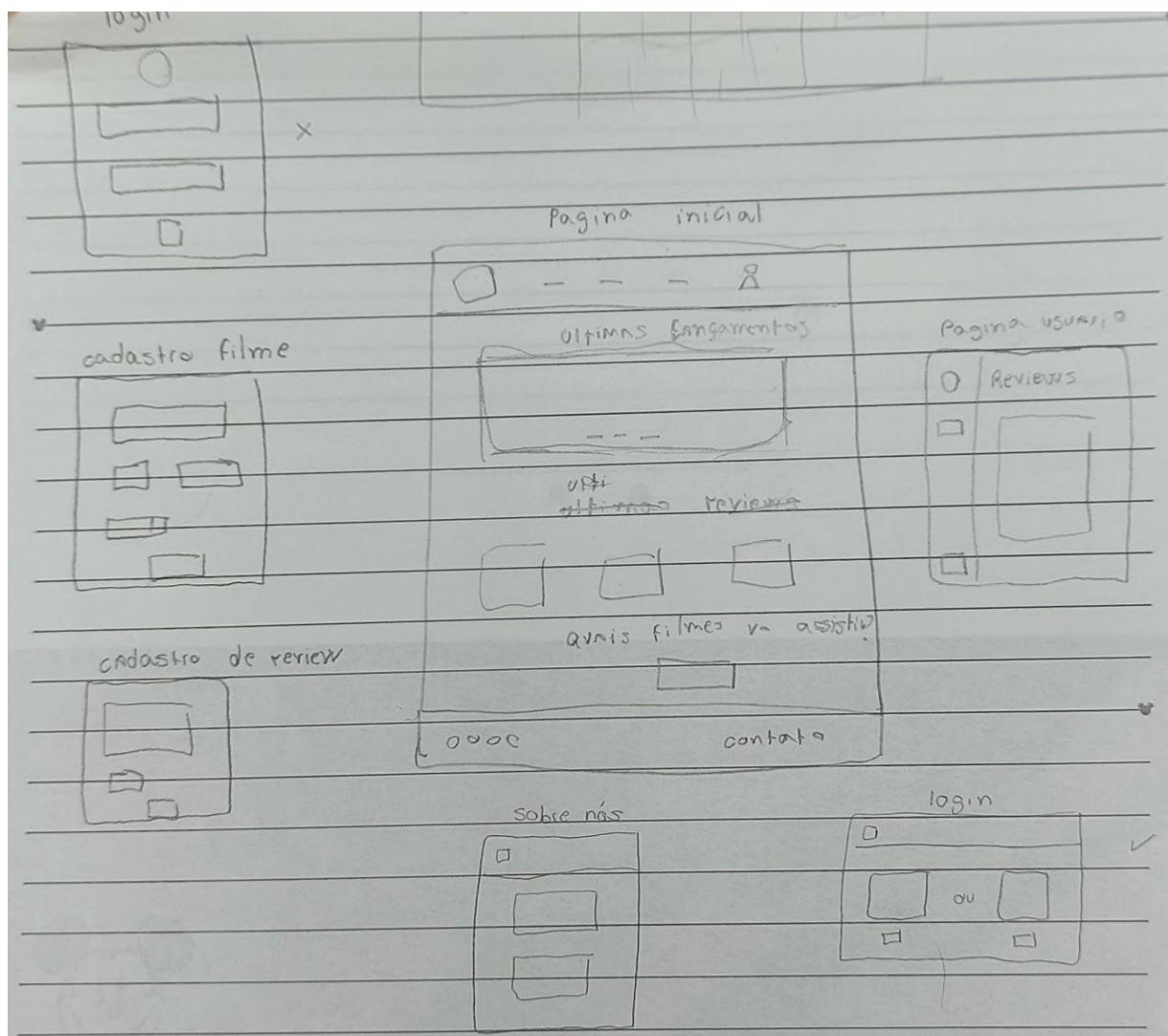
## **RESULTADOS E DISCUSSÃO**

A primeira coisa a ser feita foi a decisão de layout para o site. O objetivo inicial era que tivesse 6 páginas, sendo estas: a página inicial, de login, de perfil de usuário, de registro de crítica, registro de filme e uma informativa. Ao decorrer do desenvolvimento do trabalho, considerando o tempo, foi decidido diminuir este número de páginas para 4, além de simplificar o design geral, para facilitar a implementação. A seguir, na primeira imagem, estava a ideia inicial de separação e design das páginas, e, na segunda imagem, a ideia de implementação final destas.

Figura 1. Esboço do projeto

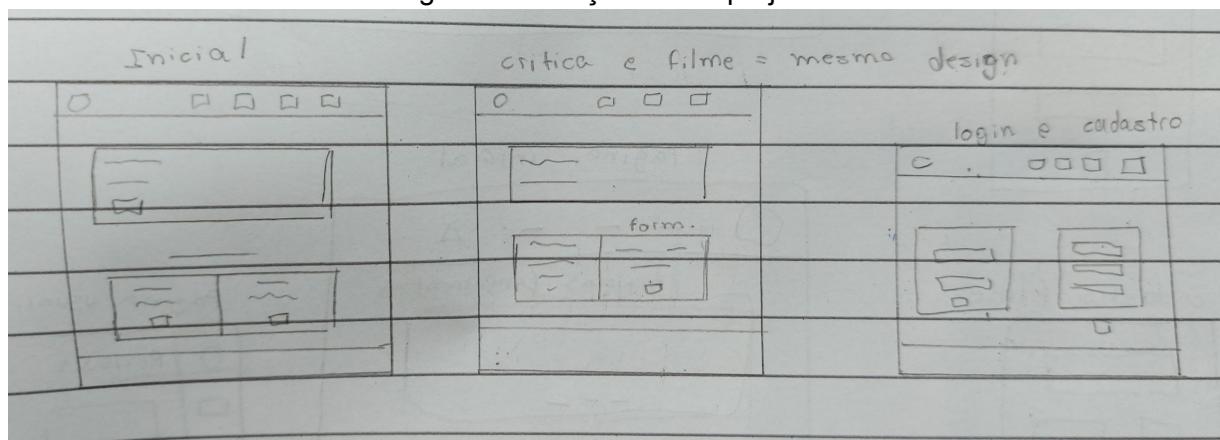
<sup>1</sup> Estudante de Graduação em Ciência da Computação, Instituto Federal Catarinense. E-mail: [juliaschaedler@gmail.com](mailto:juliaschaedler@gmail.com).

<sup>2</sup> Estudante de Graduação em Ciência da Computação, Instituto Federal Catarinense. E-mail: [miguel.menna2@gmail.com](mailto:miguel.menna2@gmail.com)



Esboço inicial das páginas do projeto.

Figura 2. Esboço final do projeto.



Esboço final da implementação das páginas, mais simples que o original.

O nome do site ficou como Moviewave, inspirado na estética escura dos sites de filmes e séries misturado com a estética retrô neon, tendo como símbolo um gatinho. O design em si, como mencionado anteriormente, ficou bem simples, inclusive

sendo um ponto interessante para considerar se este projeto for refeito no futuro. Na página inicial, temos o menu que leva para as próximas páginas; um banner que leva a tela de login e cadastro; dois banners que levam para as páginas de registro de crítica e filme, respectivamente, e, por fim, o rodapé, contendo algumas informações do site e de contato. Tanto na tela de crítica quanto de filme, o design é o mesmo, começando com o mesmo menu da página inicial, um banner simples, e a parte do formulário para preencher dados que posteriormente serão registrados no banco de dados. Um detalhe interessante que diferencia essas duas telas é que na página de crítica, há um espaço para por o id do filme, que, se estiver cadastrado no banco, puxará os dados correspondentes do filme. Por fim, a última tela é a de usuário onde há um espaço para login, caso o usuário em questão já tenha um cadastro, e a tela de cadastro, caso a informação anterior seja falsa. Para a implementação, foi utilizado html e css para a estrutura do site, php para realizar as funções de login e obtenção de dados na página e o banco de dados para armazenar estes dados, tudo isso dentro do padrão MVC. A parte mais complicada, até agora, foi inserir o php na estrutura já feita e realizar as conexões do banco. A seguir, será apresentando imagens do sistema web:

Figura 3. Página inicial



- Home
- Filmes
- Criticas
- Sobre nós
- Contato
- Log in

# MOVIEWAVE

O melhor site de críticas de filmes

[Cadastra-se](#)



**Registre:**

**Críticas**

Acabou de assistir um filme e quer deixar sua review? Escolha o filme em questão e registre sua nota!

[Registre sua críticas](#)



**Filmes**

Foi registrar sua crítica mas não tinha o filme em questão? Nos ajude a atualizar nosso site colocando novos filmes!

[Registre um filme](#)



**Sobre MovieWave**

Quisunque sollicitudin feugiat risus, eu posuere ex euismod eu. Quisque sollicitudin feugiat risus, eu posuere ex euismod eu.

[Facebook](#) [Twitter](#) [Instagram](#)

**Contato**

✉ +123, Main Street, Your City, New York USA 789456

📞 +123 4567 8900

✉ free@psdfreebies.com

🌐 www.psdfreebies.com

Copyright © 2022.

Figura 4. Página de registro de filme

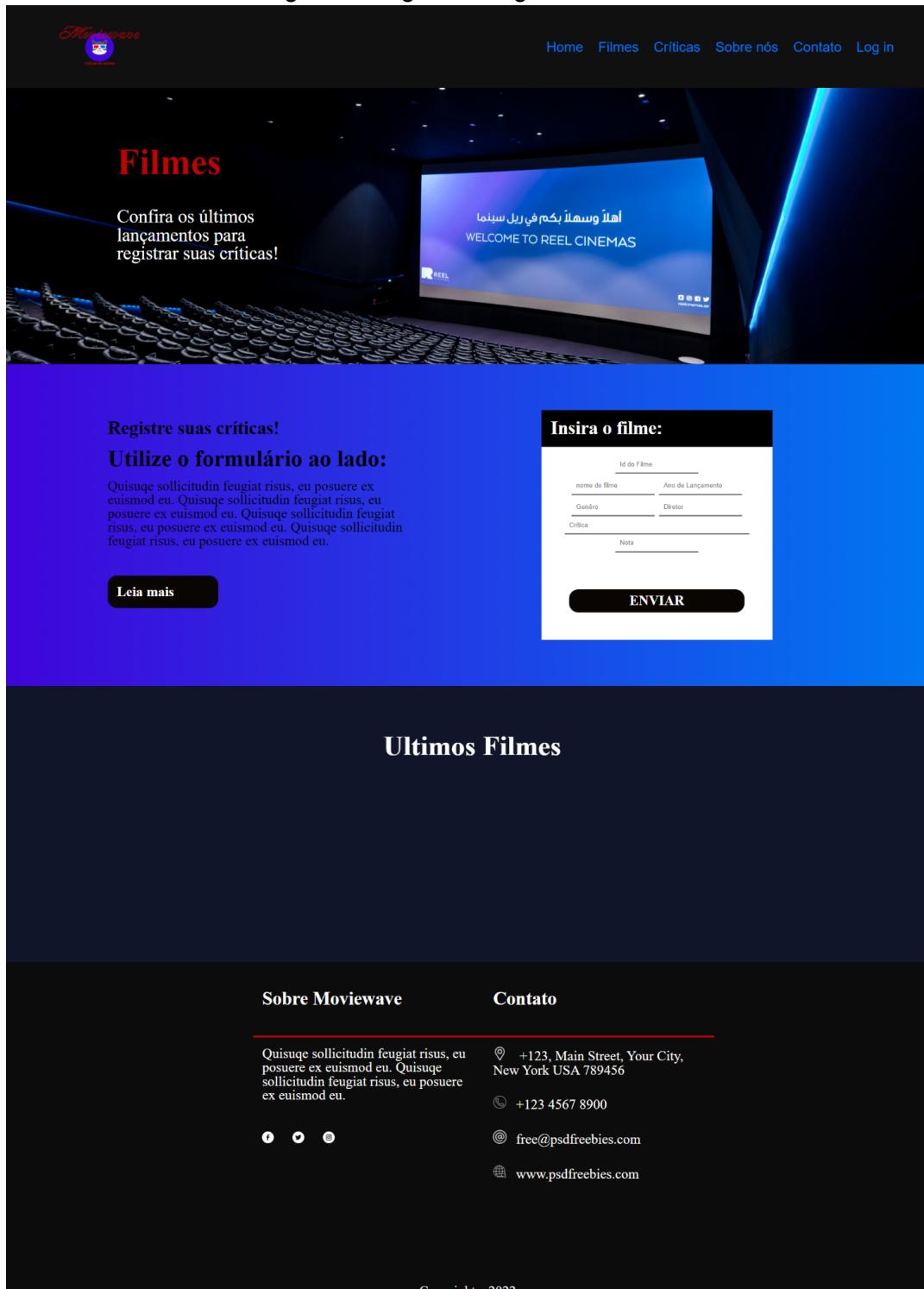


Figura 5. Página de registro de crítica

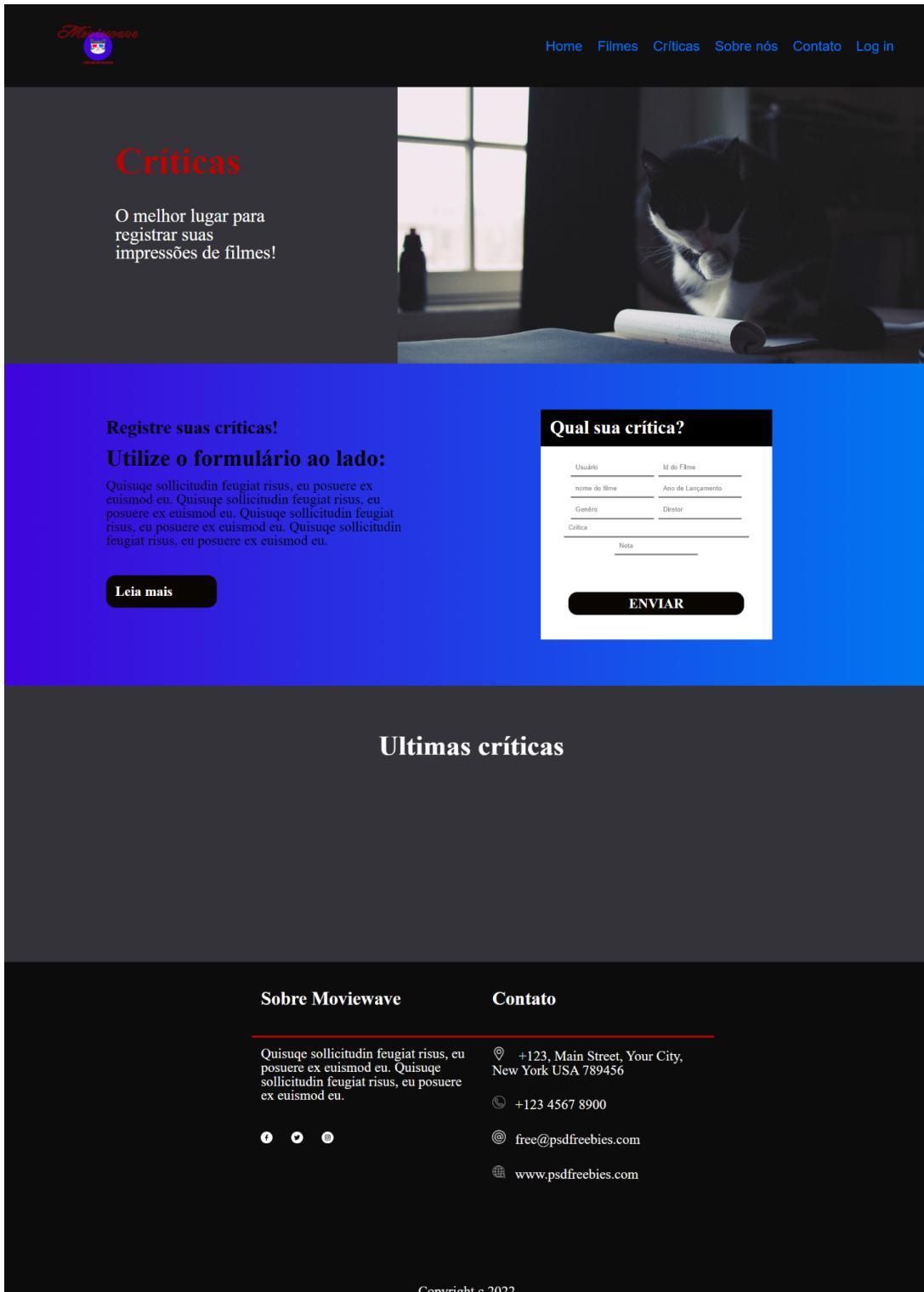
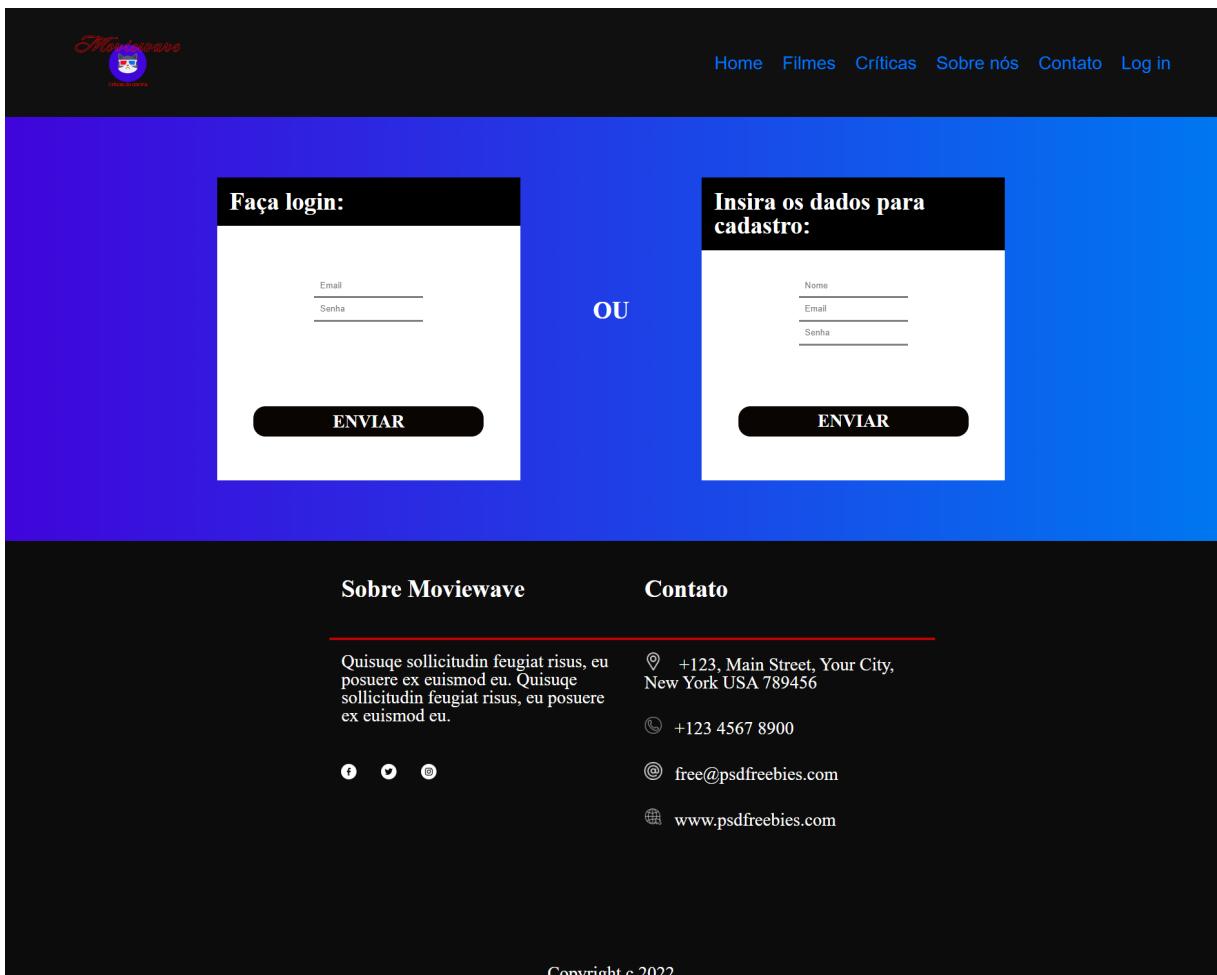


Figura 6. Página de login e cadastro



Há uma sessão “em branco” em ambas as páginas de filmes e críticas, o motivo para isso é que até o presente momento não foi implementada a função para preencher esse espaço. A intenção é que neste lugar apareçam algumas das últimas inserções do banco de dados de cada tema. Em relação ao banco de dados, o sistema está mais básico também, possuindo três tabelas, sendo estas: usuário, filmes e crítica, como demonstra a imagem a seguir. Em relação aos relacionamentos, temos 1:n entre filme e crítica (um filme pode ter várias críticas), 1:1 entre crítica e filme (uma crítica pertence ao único filme), 1:n entre usuário e crítica (um usuário pode fazer várias críticas) e 1:1 entre crítica e usuário (uma crítica pertence apenas a um usuário).

Figura 7. Tabelas do banco de dados.

```
▼ CREATE TABLE filmes (
    id INTEGER PRIMARY KEY,
    nome TEXT NOT NULL,
    anoLancamento INTEGER NOT NULL,
    genero TEXT NOT NULL,
    diretor TEXT NOT NULL
);

▼ CREATE TABLE criticas (
    id INTEGER PRIMARY KEY,
    idUser INTEGER NOT NULL,
    IdFilme INTEGER NOT NULL,
    comentario TEXT NOT NULL,
    nota FLOAT NOT NULL,
    FOREIGN KEY(IdFilme) REFERENCES filmes(id)
    FOREIGN KEY(idUser) REFERENCES usuarios(id)

);

▼ CREATE TABLE usuarios(
    id integer primary key not null,
    nome TEXT NOT NULL,
    email TEXT NOT NULL,
    senha TEXT NOT NULL
);
```

Essa tabela foi feita utilizando o Sqlite do Replit, assim como o restante do código, mas, por questões de funcionamento, no meio do trabalho decidimos continuar com o VS Code, pois o Sqlite não funciona como deveria no Replit. Antes disso, foi utilizado o MySql, mas por questões de permissões do sistema Windows foi optado, no fim, pelo VS Code. A imagem a seguir é a inserção de dados do banco MySql, mas que segue o mesmo esquema do Sqlite.

Figura 8. Inserção de dados no Banco.

### Inserção na tabela usuarios

	idusuarios	nome	email	senha
▶	1	Fulano	fulano@gmail.com	12345
	2	Miguel	miguel.menna2@gmail.com	hampter
*	NULL	NULL	NULL	NULL

### Inserção na tabela filmes

	idfilmes	nome	anoLancamento	genero	diretor
▶	1	Duna	2021	Ficção científica	Denis Villeneuve
	2	A viagem de Chihiro	2001	Animação	Hayao Miyazaki
*	NULL	NULL	NULL	NULL	NULL

### Inserção na tabela criticas

	idcriticas	idUser	idFilme	comentario	nota
	1	1	1	Filme espetacular, com certeza melhor que o an...	10
	2	1	2	Animação muito boa de assistir, história incrível,...	9
▶*	NULL	NULL	NULL		NULL

As últimas coisas a serem feitas foram o sistema de login, que deve comparar os dados inseridos pelo usuário e comparar com os do banco; e sistema de cadastro de filme, crítica e usuário, que pegaria os dados e colocaria no banco. Para fazer o login, o usuário coloca os dados e uma função, dentro da model usuário, irá fazer a comparação e no controller do login, irá retornar se o login do usuário foi realizado ou não. O código a seguir funciona exatamente do modo descrito.

Figura 9. Código para login.

```
public static function findByEmailAndPassword(string $email, string $senha){
    $conn = new Database();
    $comando = $conn->executeQuery('SELECT * FROM usuarios WHERE email = :EMAIL AND senha= :SENHA LIMIT 1', array(
        ':EMAIL' => $email,
        ':SENHA' => $senha
    ));

    return $comando->fetchAll(PDO::FETCH_ASSOC);
}
```

O código pega email e senha do usuário para depois comparar com os dados do banco.

Figura 10. Função logar.

```

public function logar(){
    $email = $_POST['email'];
    $senha = $_POST['senha'];

    $User = $this->model('Users');
    $data = $User::findByEmailAndPassword($email, $senha);
    // print_r($data);

    if (empty($data)) {
        $this->view('login/index');
        echo "Usuário ou senha inválido";
    } else {
        echo "Usuário logado com sucesso";
        $this->view('home/index');
    }
}

```

O código anterior é chamado nesta função para realizar a comparação e retornar o usuário para onde ele deve ir.

Para cadastro, foi utilizado o comando a seguir, seguindo a lógica do comando anterior, porém agora estamos fazendo uma inserção no banco.

Figura 11. Comando para cadastrar

```

public static function insertData(string $nome, string $email, string $senha){
    $conn = new Database();
    $comando = $conn -> executeQuery('INSERT INTO usuarios (nome, email, senha) VALUES (:NOME, :EMAIL, :SENHA)', array(
        ':NOME' => $nome,
        ':EMAIL' => $email,
        ':SENHA' => $senha
    ));

    return $comando->fetchAll(PDO::FETCH_ASSOC);
}

```

Comando insere os dados colocados pelo usuário no banco.

Para o cadastro de filmes e crítica, segue a mesma lógica. Entretanto, para crítica, foi pensado que seria interessante apresentar, de alguma forma, para o usuário uma tabela com os filmes já cadastrados e seus ids, para que quando ele fosse registrar a crítica, ele pudesse por o id do filme desejado e automaticamente puxar todas as informações daquele filme (gênero, ano de lançamento, diretor, etc).

Essa função infelizmente não foi implementada, assim como os teste finais, pois houve um erro com banco de dados, especificamente com aquela permissão do php.ini, pois, por algum motivo que ainda não foi descoberto, o sqlite não consegue ser habilitado, mesmo sendo colocado nas variáveis de ambiente para ter permissão.

Figura 12. Erro no driver do banco.



Fatal error: Uncaught PDOException: could not find driver in E:\TrabalhoFinalProg2\Moviewave\Application\core\Database.php:22  
Stack trace: #0 E:\TrabalhoFinalProg2\Moviewave\Application\core\Database.php(22): PDO->\_\_construct('sqlite:moviewav...') #1  
E:\TrabalhoFinalProg2\Moviewave\Application\models\Críticas.php(18): Application\core\Database->\_\_construct() #2  
E:\TrabalhoFinalProg2\Moviewave\Application\controllers\Crítica.php(14): Application\models\Críticas::findAll() #3 [internal function]: Crítica->index() #4 E:\TrabalhoFinalProg2\Moviewave\Application\core\App.php(26): call\_user\_func\_array(Array, Array)  
#5 E:\TrabalhoFinalProg2\Moviewave\public\index.php(38): Application\core\App->\_\_construct() #6 {main} thrown in  
E:\TrabalhoFinalProg2\Moviewave\Application\core\Database.php on line 22

## CONSIDERAÇÕES FINAIS

O trabalho, no geral, foi uma ótima oportunidade de colocar em prática o que foi visto na matéria. Apesar de algumas dificuldades enfrentadas durante o percurso, foi muito interessante realizar esse projeto pois assim podemos ter uma visão mais próxima do real desenvolvimento de um sistema web. Infelizmente, na reta final do trabalho, houve alguns problemas em relação ao banco de dados com permissões do sistema, que não estava acontecendo nos computadores do laboratório, onde parte do trabalho foi desenvolvido. Provavelmente foi algum erro de instalação do próprio PHP que não foi achado a tempo. Todavia, foi possível aprender muito com este projeto, e, para projetos futuros, o foco seria trazer mais funções da linguagem e também trabalhar com frameworks, além de melhorar a questão estética do site, para sim ter um produto final mais fluido e funcional.