UNIVERSIDADE AUTÓNOMA DE LISBOA
LUÍS DE CAMÕES

DEPARTAMENTO DE CIÊNCIAS E TECNOLOGIAS

MESTRADO EM ENGENHARIA INFORMATICA E DE TELECOMUNICAÇÕES

Uso do estado da arte no desenvolvimento de aplicações web para a criação e teste de uma aplicação responsiva

Trabalho de Projeto para a obtenção do grau de Mestre em Engenharia Informática e de Telecomunicações

Autor: Ricardo Nuno Rafael Monteiro da Silva Pinto
Orientadora: Doutora Isabel Alvarez
Número do candidato: 30005649

Julho de 2021, Lisboa

1 Resumo

O aparecimento da Web 2.0 obrigou as empresas a mudar a sua estratégia de realização de negócios na internet [1]. A criação de aplicações e *sites* de negócio tornou-se crucial. Existem atualmente diversas ferramentas e tecnologias que auxiliam a programação de aplicações e *sites* de forma a oferecer sempre um melhor dinamismo [1]. Os programadores deparam-se com perguntas como: Quais serão as tecnologias com menor custo para chegar ao objetivo requerido? Quais serão as tecnologias que permitem a construção desta aplicação da maneira mais rápida e simples? Qual será o desempenho de cada uma destas tecnologias? Como será possível criar uma aplicação que funcione em vários tamanhos e tipos de ecrãs? Este processo da escolha ideal pode ser complexo devendo ser considerados diversos fatores para satisfazer os objetivos do projeto que se pretende desenvolver.

Este trabalho de projeto possui três principais objetivos. O primeiro foi o de investigar as tecnologias capazes de criar aplicações web. Esta investigação foi realizada tendo em mente o mercado deste tipo de aplicações, plataformas de desenvolvimento, linguagens de programação, plataformas de base de dados e servidores, segurança, *design* responsivo e princípios de *design* em aplicações web e seguiu uma metodologia básica estratégica, descritiva, indutiva e bibliográfica. O segundo objetivo concentrou-se na seleção de tecnologias para a criação de uma aplicação deste tipo. Esta seleção foi baseada no primeiro objetivo deste trabalho de projeto. O terceiro foi o desenvolvimento de uma aplicação web para comentar, listar, avaliar e partilhar opiniões sobre filmes, o qual foi composto por três fases. Na primeira fase trabalhou-se a análise de requisitos com um caso de estudo sobre o problema programático abordado, incluindo a descrição dos serviços disponíveis, os requisitos funcionais e o desenho da base de dados.

Na segunda fase foi feita a implementação da aplicação usando o *Microsoft Visual Studio*. Na última fase foram efetuados testes de usabilidade à aplicação desenvolvida com o objetivo de perceber qual a perspetiva dos utilizadores em relação à mesma. Utilizou-se a metodologia de *Design Science Research* para a criação da aplicação.

Palavras-chave: Web 2.0; Aplicações web; *Responsive design;* Bootstrap;

2 Abstract

The emergence of Web 2.0 forced companies to change their strategy for doing business on the internet [1]. The creation of business applications and websites has become crucial. There are currently several tools and technologies that help the programming of applications and websites in order to always offer a better dynamism [1]. Programmers are faced with questions such as: What will be the technologies with the lowest cost to reach the required goal? What technologies will allow this application to be built in the quickest and easiest way? How will each of these technologies perform? How will it be possible to create an application that works on different sizes and types of screens? This process of the ideal choice can be complex, and several factors must be considered to satisfy the objectives of the project to be developed.

This project work has three main objectives. The first one was to investigate the technologies capable of creating web applications. This investigation was carried out keeping in mind the market for this type of applications, development platforms, programming languages, database and server platforms, security, responsive design and design principles in web applications and followed a basic strategic, descriptive, inductive and bibliographical methodology. The second objective focused on the selection of technologies for creating an application of this type. This selection was based on the first objective of this project work. The third was the development of a web application to comment, list, rate and share opinions about movies, which consisted of three phases. In the first phase, the requirements analysis was worked with a case study on the programmatic problem addressed, including the description of the available services, the functional requirements and the database design.

In the second phase, the application was implemented using Microsoft Visual Studio. In the last phase, usability tests were carried out on the application developed in order to understand the perspective of users in relation to it. A Design Science Research methodology was used to create the application.

3 Índice

4 Lista de Ilustrações

5 Lista de Siglas

API- *Application Program Interface*

CSS- *Cascading Style Sheets*

DSR- *Design Science Research*

HTML- *Hypertext Markup Language*

HTTP- *Hypertext Transfer Protocol*

IU- Interface do Utilizador

MVC- *Model View Controller*

PDF- *Portable Document Format*

Rest- *Representational State Transfer*

SEO- *Search Engine Optimization*

SQL- *Structured Query Language*

URL- *Uniform Resource Locator*

Web- *World Wide Web*

WCAG- *Web Content Accessibility Guidelines*

XML- *Extensible Markup Language*

6 Introdução

Este capítulo tem o objetivo de descrever a contextualização, motivação, objetivos, perguntas de investigação, e a organização do documento deste trabalho de projeto.

6.1 Contextualização

Web 2.0 é um conceito que se refere a um conjunto de tendências económicas, sociais e tecnológicas que servem de base para a criação de uma internet mais madura e distinta caraterizada pela interação do utilizador, abertura e efeitos de rede [1]. O aparecimento da Web 2.0 obrigou as empresas a mudar a sua estratégia de realização de negócios [1]. Esta mudança levou a que aplicações com contato direto com o cliente se tornassem mais dinâmicas e acessíveis [1]. Atualmente existe um grande número de opções que permitem criar todo este dinamismo e acessibilidade e que facilitam e auxiliam os programadores destes serviços. O grande número de opções aumenta a dificuldade de iniciar um novo projeto nesta área, devido à difícil tarefa de selecionar quais as melhores tecnologias para o objetivo final [1].

Atualmente existem diversas ferramentas e tecnologias que auxiliam a programação de aplicações e *sites* de forma a oferecer sempre um melhor dinamismo [1]. Esta é uma área que tem visto um grande crescimento o que pode levar a uma maior indecisão no processo de seleção das tecnologias a utilizar.

Quando se inicia um novo projeto de criação de uma aplicação é necessário responder a perguntas como: Quais serão as tecnologias com menor custo para chegar ao objetivo requerido? Quais serão as tecnologias que permitem a construção desta aplicação da maneira mais rápida e simples? Qual será o desempenho destas tecnologias? Como será possível criar uma aplicação que funcione em vários tamanhos e tipos de ecrãs? Uma escolha incorreta de tecnologias pode levar a custos bastantes elevados, ineficiência e problemas de desempenho entre outros.

6.2 Motivação

Este projeto surge devido à rápida evolução de tecnologias para criação de aplicações web e ao rápido aumento de utilização da internet por parte de utilizadores de dispositivos móveis. Em 2021 mais de metade, 56 porcento, de todo o tráfego *online* foi realizado por dispositivos

móveis [2]. Esta alta percentagem torna-se ainda mais relevante quando reconhecemos que o número médio de segundos gastos em *websites* por utilizadores de dispositivos móveis continua a descer ano após ano enquanto a média de segundos em *desktops* continua mais ou menos fixa [2].

Tendo isto em mente, o investigador teve interesse em realizar uma investigação sobre as tecnologias que permitem criar aplicações web capazes de funcionar em qualquer ecrã. Além disto, criou-se também uma aplicação com o objetivo de aplicar toda a investigação realizada neste trabalho de projeto.

6.3 Objetivos

A investigação realizada foi orientada para a concretização de vários objetivos gerais e específicos. O cumprimento destas metas irá facilitar na procura de uma resposta à pergunta de investigação referenciada no subcapítulo 6.4. Estes objetivos são os seguintes:

- Identificar e caracterizar as tecnologias atualmente existentes para a criação de uma aplicação web com as características definidas;
- Avaliar os pontos fortes e fracos destas tecnologias;
- Selecionar as tecnologias mais adequadas para a aplicação em questão e providenciar uma justificação para esta escolha;
- Construção e teste de uma aplicação exemplo para aplicar os resultados da investigação de forma prática.

6.4 Pergunta de investigação

Segundo Quivy e Campenhoudt uma questão de investigação define um projeto de investigação e só é útil se for corretamente formulada [3]. É também necessário que a pergunta seja clara, efetiva, unívoca e pertinente [3]. Tendo isto em mente a pergunta desta investigação é a seguinte:

Que tecnologias permitem a criação de uma aplicação web para vários utilizadores com ecrãs de formato diferente?

6.5 Organização do documento

Este documento está organizado da seguinte forma:

- Capítulo 7 (Revisão Literária): apresenta toda a investigação realizada.
- Capítulo 8 (Metodologia): descrição da toda a metodologia utilizada neste trabalho de projeto.
- Capítulo 9 (Análise do Sistema a Desenvolver): descrição de uma análise à aplicação desenvolvida.
- Capítulo 10 (Descrição do Sistema): apresenta a descrição da aplicação desenvolvida.
- Capítulo 11 (Desenvolvimento): processo de desenvolvimento da aplicação.
- Capítulo 12 (Testes): são descritos os testes de usabilidade realizados para testar a aplicação web desenvolvida.
- Capítulo 13 (Conclusão): são apresentadas as conclusões do projeto.
- Capítulo 14 (Trabalho futuro): são apresentados possíveis trabalhos futuros a realizar.

7 Revisão Literária

Neste capítulo será abordado a investigação que serviu de guia para a escolha das tecnologias de desenvolvimento web. Os tópicos aqui expostos irão definir e caracterizar os conceitos que foram mais importantes na realização da escolha.

7.1 Web 2.0 em relação a 1.0

O aparecimento da Web 2.0 revolucionou a maneira como a web é utilizada [4]. O'Reilly definiu a Web 2.0 através das seguintes regras [5] [6]:

- ***Beta State***- não tratar *software* como um artefacto, mas como um processo de comprometimento com os seus utilizadores [5].
- **Pequenas peças unidas**- abrir dados e serviços para que estes sejam reutilizados por outros. Reutilizar dados e serviços de outros sempre que possível [5].
- ***Software* acima do nível de um único dispositivo** - Desenvolver aplicações que se encontram no espaço entre o cliente e o servidor [5].
- **Lei da conservação de lucros**- em um ambiente de rede, API´s abertas vencem, mas isso não significa que a competitividade tenha de desaparecer [5].
- **Dados são o novo *Intel inside***- No futuro a informação será cada vez mais importante dentro de um mercado competitivo [5].

A principal diferença entre web 2.0 e web 1.0 é a sua estrutura. "A web fragmenta-se em inúmeras permutações com diferentes aspetos, comportamentos e usos dentro dos diferentes *hardwares*" [7] [8]. Esta nova web é tratada como um mecanismo de transporte em vez de ecrãs com texto e grafismo, por outras palavras "o éter através do qual a interatividade acontece" [7]. Outra grande diferença é o aparecimento de páginas não estáticas. Estas páginas permitem maior interatividade, maior comunicação entre utilizadores e maior número de utilizadores não passivos da internet [8]. Exemplos disto são redes sociais, *sites* de partilhas de vídeo, blogs, *wikis* entre outros.

7.2 Tipos de Aplicações

Nos últimos anos uma parte significativa da sociedade passou a depender de equipamentos móveis para o seu quotidiano [8]. Esta dependência existe de vários modos. O individuo

depende do seu dispositivo móvel para aceder a redes sociais, para comunicar com outros a quilómetros de distância, para se entreter nos seus tempos de lazer e para se informar em relação ao mundo que o rodeia [8]. As empresas dependem de aplicações para gerir, publicitar, comunicar e até vender os seus produtos. Assim as empresas que fabricam dispositivos móveis aumentaram a sua capacidade a um ritmo acelerado. Nasceram três formas de criar aplicações para dispositivos móveis [9]:

1. **As aplicações nativas**: são concebidas para correr num sistema operativo de dispositivos móveis específicos [9]. Estas aplicações são programadas com o uso de linguagens de programação específicas e são suportadas por quadros de desenvolvimento desenvolvidos pelos fornecedores de sistemas operativos [9]. Na maioria dos casos são instaladas no dispositivo móvel com os dados de utilizador guardados no próprio dispositivo ou em nuvem [9].

2. **As aplicações web**: são desenvolvidas para serem utilizadas através de um navegador da Internet [9]. Podem ser executadas num servidor HTTP ou localmente no dispositivo do utilizador. Estas aplicações fazem uso de servidores Web. Estes servem para receber e responder a pedidos do cliente. Assim o *browser* é capaz de solicitar recursos ao servidor pelo utilizador. Estes recursos podem variar desde páginas html e documentos PDF, entre outros [10]. O servidor em si contém recursos, mas sozinho não iria permitir páginas dinâmicas ou base de dados [10].

3. **As aplicações híbridas**: são capazes de serem utilizadas em vários sistemas operativos [11]. Têm de ser instaladas no dispositivo móvel, mas permitem o uso de várias plataformas e sistemas operativos [11]. Estas aplicações são limitadas em relação às suas funcionalidades porque estas só conseguem funcionar em primeiro plano [11].

As aplicações móveis nativas e híbridas são distribuídas através de plataformas de distribuição de aplicações móveis que funcionam como mercado destas aplicações [12] e têm tendência a serem mais rápidas devido a não dependerem da internet para o seu funcionamento [13]. Ambas estas vantagens existem devido à instalação da aplicação no dispositivo e à personalização do código para o sistema operativo, isto no caso das nativas [13] . No entanto, estas aplicações são mais caras e complicadas de criar, têm de passar um longo processo de

revisão antes de serem adicionadas a lojas de distribuição e necessitam de ser instaladas ocupando espaço no dispositivo em questão [9]. Por outro lado, as aplicações web não necessitam de utilizar as lojas de aplicações e são mais simples e baratas de produzir [9]. Também têm a vantagem de serem independentes de plataformas de dispositivos e quando são projetadas e desenvolvidas seguirem abordagens específicas [9]. Estas abordagens podem ser *adaptive* ou *responsive* [9].

Para este trabalho de projeto decidiu-se focar a investigação e a construção do exemplo no tipo web. Esta seleção foi tomada com base nos vários fatores abaixo discriminados:

- A não necessidade da utilização de plataformas de distribuição.
- O custo reduzido e o menor grau de dificuldade ao criar a aplicação.
- A aplicação em questão apenas teria significado e sentido quando ligada à internet. A única ação que seria possível sem internet poderia ser a procura de um filme o que representa menos de um terço das funcionalidades da aplicação. E mesmo esta ação iria necessitar de acesso prévio à internet e um elevado número de espaço de memória. Isto porque seria necessário guardar no dispositivo uma lista de todos os filmes.

Depois desta decisão tornou-se necessário selecionar qual a plataforma mais adequada para a construção de uma aplicação web.

7.3 Plataformas para criação de aplicações web

Existem diversas plataformas de desenvolvimento para aplicações web. Neste trabalho de projeto consideramos a investigação de três plataformas, essencialmente devido ao seu alto nível de popularidade e quantidade de funcionalidades únicas [14]:

- O Appery - é uma plataforma de desenvolvimento de aplicações capaz de produzir todos os tipos de aplicações, que não necessita de instalação e faz uso de um editor visual. Possui opções de arraste e soltar (o que é benéfico para indivíduos não programadores ou *designers*), e permite escolher modelos e componentes para adicionar à sua aplicação. Possui também funções de desenvolvimento avançado como serviços de *back-end* em base de dados em nuvem, notificações *push*, APIs REST, etc. Faz também

uso de plugins para media, mailing, social, e até suporta tecnologias como jQuery, Angular.js, Bootstrap integrado [15].

- O Swing2App- é uma plataforma em nuvem desenhada para criação de aplicações, onde um utilizador pode verificar a IU (*Interface* do Utilizador) da aplicação. Esta verificação funciona em tempo real e faz uso de uma máquina virtual. A plataforma funciona para os sistemas operativos Android e iOS, em qualquer dispositivo, hora ou local. Fornece também ferramentas de administração e de gestão disponíveis 24 horas por dia. A principal vantagem desta plataforma é a não necessidade de codificação e a rapidez de criação de uma aplicação. Isto é possível devido ao uso de modelos predefinidos, fazendo também uso de um painel de administração para gerir a aplicação.

- O Visual Studio Code- é um editor de código, lançado em 2015 pela Microsoft, com o objetivo de facilitar o desenvolvimento de aplicações web. Foi anunciado no evento Build e trata-se de uma ferramenta multiplataforma que funciona para uma gama de projetos. O editor deste programa possui suporte à sintaxe de várias linguagens [16]. O ponto mais importante ao olhar para esta plataforma será a utilização de Net. Esta tecnologia é uma das tecnologias mais fortes na criação de aplicações web [17].

7.4 C # e ASP.Net

C# é uma linguagem orientada a objetos desenvolvida pela Microsoft e faz parte da plataforma .Net. Embora a linguagem tenha sido criada do zero foi baseada no C++ [18].

A plataforma .Net, inicialmente, criou as suas bibliotecas fazendo uso de *Simple Managed* C, porém, isto limitou muito o trabalho na plataforma. No início de 1999 uma equipa, liderada por Anders Hejlsberg, criou uma linguagem para a plataforma .Net. Essa linguagem tinha como objetivo tornar o .Net mais independente de outras linguagens. Assim a criação da linguagem C# ajudou no desenvolvimento do .Net porque a plataforma não teve de se ajustar a linguagens já existentes. C# foi criado especificamente para uso de .Net [18].

"O visual Studio.Net é uma plataforma integrada de desenvolvimento que permite criação de vários tipos de projetos e soluções usando Visual Basic .Net, C# e Visual C++.Net" [19]

ASP.NET é uma tecnologia de *script* orientada a servidores. Esta tecnologia permite colocar web *scripts* numa aplicação para execução pelo servidor. Todas as aplicações .Net são compiladas em vez de interpretadas e é impossível correr código C# não compilado [19].

As páginas ASP.Net são uma parte principal das aplicações ASP.Net, isto porque, estas estão responsáveis pelo output que é enviado para o *browser* depois de o cliente realizar um *request*. Uma página ASP.Net é equivalente a uma página HTML e pode conter HTML, XML e scripts. Estes scripts são interpretados no servidor [19].

Nas aplicações web assim que uma página é renderizada, toda a informação e objetos são descartados. Este modelo melhora a escalabilidade e o tráfego das aplicações. Uma forma de resolver este problema é utilizando a função *View State* que coloca variáveis num campo escondido. Estas variáveis mantêm-se iguais quando a página faz o seu *reset* [19].

ASP.Net utiliza também o modelo de eventos, ou seja, são selecionados eventos e são programadas respostas a esses eventos. Cada evento possui um gestor de evento ou e*vent handler* que servirá para programar e gerir os eventos separadamente, sendo possível criar ligação entre estes. Alguns exemplos destes *event handlers* são o Page.Init (gere o evento de abertura de página), Page.Load (gere o evento de *load* da página), TextBox.TextChanged (gere o evento de mudança de texto numa caixa de texto), Button.Click (gere o evento de clicar num botão) entre outros [18]. Além disto o .Net possui um serviço de ligação a um servidor com base de dados simples e seguro [18].

7.5 MVC

MVC consiste num estilo de arquitetura de *software* que contribui para a otimização de velocidade entre os vários pedidos realizados por comandos de utilizadores. Este tipo de arquitetura possui três camadas: *Models*, *Views* e *Controllers* [20].

A camada de *Models* é orientada a objetos e gere todas as tarefas relacionadas com dados. Estas tarefas podem ser de validação, estado e controlo de sessão e estrutura de base de dados. Esta camada reduz a complexidade do código [20]. Podemos encontrar nesta camada métodos de acesso à base de dados. Embora esta camada seja construída com dados abstratos em mente, possui classes fixas que definem o domínio de interesse [20]. Os objetos criados nesta camada muitas vezes encapsulam dados da base de dados, mas também possuem códigos para controlar e gerir estes dados [20]. Esta camada deve ser mantida o mais simples possível e com informação apenas relacionada com os objetos que cria. Quando esta camada consegue ser

simples ela torna os seus modelos e classes altamente reutilizáveis e a maioria do trabalho dos programadores passa pelos *controllers* [20]. Os modelos devem saber o mínimo possível sobre a conexão ao mundo exterior por razões de segurança [21].

A camada de *Views* é responsável pela parte gráfica e pela *interface* para uso de utilizador [20]. Ao separar a lógica da aplicação da parte gráfica reduz o número de erros que os programadores podem cometer [20]. As várias *views* controlam a maneira como os dados são revelados ao utilizador e permitem a interação entre o programa e o utilizador. Atualmente a maioria dos programadores usam *templates* para criar *views* de maneira mais fácil e eficiente [20]. As *views* têm de ser capazes de reconhecer a existência dos modelos e a sua natureza. Sem este reconhecimento as *views* seriam incapazes de controlar e expor certos tipos de dados [21].

A camada de *Controllers* gere eventos [20]. Estes eventos podem acontecer quando existe interação com a *interface* ou podem ser requisitados pelo sistema. Um *controller* recebe pedidos e responde com a informação requisitada [20]. Esta camada interage com a camada de *Models* para receber a informação requisitada e de seguida envia essa informação para *views* onde vai ser exposta para os utilizadores [20]. Quando um pedido chega ao servidor, a *framework* MVC atribui um *controller* a esse pedido baseado no URL. Esta camada serve de comunicação entre as outras duas e também está encarregue de resolver potenciais erros [20].

Este tipo de arquitetura possui as seguintes vantagens [22]:

- Desenvolvimento da aplicação mais rápido.
- É mais fácil vários programadores trabalharam no mesmo projeto.
- É mais fácil realizar o processo de *debug* devido à existência de três camadas separadas.
- É mais fácil de realizar o processo de *update*.
- O *controller* possui a função de filtro e é capaz de impedir que qualquer dado incorreto chegue à camada de modelos.
- É mais fácil realizar mudanças ao código.

Este tipo de arquitetura possui as seguintes desvantagens [22]:

- A compreensão desta *framework* pode ser difícil.
- Necessidade de regras rígidas para os métodos.

7.6 Linguagens de suporte

O .Net permite o uso de várias linguagens e *frameworks* de suporte. Neste capítulo realizou-se uma análise detalhada de tecnologias de suporte para .Net.

7.6.1 HTML e CSS

O HTML é uma linguagem de marcação. Estas linguagens são constituídas por códigos que delimitam conteúdos específicos. A divisão da página é possível através do uso de *tags*. "Inicialmente era muito complicado aprender HTML, pois eram muitos comandos para fazer algo simples. A cada nova versão, o HTML fica mais fácil de utilizar, e adquire mais funções. Atualmente qualquer pessoa pode aceder à internet e aprender a construir um *site* básico em questão de horas, seguindo os passos de tutoriais e aprendendo as funções de cada código" [23].

O HTML permite também o uso de linguagens externas como o CSS e o Javascript [23].A versão mais recente de HTML é o HTML5 [24]. O HTML5 traz novidades como:

- A inclusão do elemento *canvas* para desenho [24].
- A inclusão de elementos de vídeo e áudio para reprodução multimédia [24].
- Melhor suporte para armazenamento local [24].
- Inclusão de novas *tags* [24].
- Inclusão de novos controlos para formulário e total suporte CSS3 [24].

O CSS é uma linguagem de estilos, ou seja, determina a aparência ou *layout* de páginas web [25]. Este programa permite ao utilizador criar páginas web mais facilmente com a utilização de certos códigos. Consegue controlar opções como margem, linhas, cores, alturas, larguras, imagens e posicionamento [25]. Utiliza códigos prontos para economizar tempo. Consegue controlar várias páginas HTML com uma só página CSS, permite facilitar *layouts* e é fácil de aprender [25]. O CSS é um elemento indispensável para a implementação de *responsive design* que será abordado mais à frente, no capítulo 7.7.

7.6.2 Bootstrap

Bootstrap é uma *open source framework* desenhada para a construção de *interfaces* web que utilizem *design* responsivo [26]. O Bootstrap utiliza JavaScript (incluindo jQuery), CSS e HTML e inclui suporte para CSS3 e HTML5.

Bootstrap é a ferramenta mais popular para desenvolver aplicações web responsivas. É uma *framework* de *front-end* que facilita o desenvolvimento web [27]. Bootstrap faz uso de HTML e modelos de *design* baseados em CSS para tipografia, formulários, botões, tabelas, navegação, carrosséis de imagens e muitos outros [27]. Também é possível utilizar *plug-ins* de JavaScript.

Bootstrap é suportado por todos os *browsers* populares (Chrome, Firefox, Microsoft Edge etc.). É fácil de utilizar se o programador possuir conhecimento de CSS e HTML [27]. Bootstrap é um método eficaz para criar *layouts* consistentes e eficazes para web *design* [27]. Esta tecnologia utiliza o sistema de *fluid grids* que escala adequadamente até doze colunas conforme o tamanho do dispositivo ou janela de visualização [27]. Além disso, utiliza classes predefinidas para uma mais fácil gestão de l*ayout* [27].

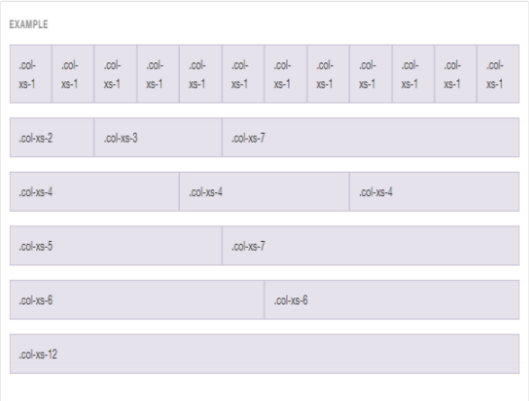

Figura 1 Bootstrap Grid ***Fonte****: [27]*

Bootstrap possui quatro categorias de recursos de *design*: *scaffolding*, componentes, *plug-ins* de JavaScript e CSS básico [26]. *Scaffoldings* são constituídos por *grids* responsivas de 12 colunas, *layouts* e componentes. Os componentes incluem estilos básicos para recursos comuns da *interface* do utilizador [26]. Os *plug-ins* de JavaScript seguem os componentes, mas usam o JavaScript em vez de CSS para os elementos [26]. O CSS básico inclui estilos para elementos HTML básicos [26].

7.6.3 Linguagem e plataforma para suporte à base de dados

Se uma aplicação necessitar de uma base de dados, esta terá de ser pensada e estruturada para satisfazer as necessidades da aplicação. Existem dois tipos de base de dados: relacional e não relacional [28]:

- Uma base de dados relacional é baseada no modelo de dados relacionais. Estas bases de dados oferecem aos utilizadores a linguagem SQL (*Structured Query Language*) que irá gerir e consultar a base de dados. Uma base de dados relacional é fundamentalmente um conjunto de dados que possuem relacionamentos predefinidos entre eles através do uso de chaves [28]. Os dados são todos armazenados em tabelas e linhas. Uma tabela

armazena um conjunto de dados que representam várias características de um certo aspeto [28]. Uma coluna de uma tabela representa todos os valores de uma característica específica. As linhas da tabela são representações de várias características de um objeto. Uma linha é identificada pela sua chave primária ou conjunto de chaves primária e estrangeira. Este tipo de base de dados apresenta como principais características uma alta precisão de procura, alta segurança, normalização de dados e um modelo simplista [28].

- Uma base de dados não relacional não utiliza o formato de tabela, linha e coluna. Estas bases de dados funcionam através de várias estruturas de modelagem e armazenamento de dados. Possuem maior disponibilidade e escalabilidade [28]. A popularidade destas bases de dados aumentou devido à sua capacidade de armazenamento de grandes quantidades de dados de vários tipos e formatos diferentes. Estas bases de dados são usadas em tecnologias de Big Data. Além disto, este tipo de base de dados é capaz de guardar qualquer tipo de informação ou ficheiro [28].

7.6.3.1 SQL

O SQL é uma linguagem de programação que serve para comunicação e manipulação de base de dados. É utilizado por empresas para manipular e criar dados que são armazenados em servidores. Permite também retirar dados específicos de tabelas e é uma linguagem simples e poderosa [29].

O SQL utiliza um conjunto de comandos para manipular dados. Além disto também utiliza *constraints* e tipos de variáveis para regular o tipo de informação inserida na base de dados [29]. As tabelas relacionam-se através do uso de chaves [29]. “Nos dias de hoje, a linguagem de SQL é considerada um *standard* dos sistemas Gestores de Base de Dados Relacionais, por isso todos os fabricantes a integram nos seus produtos” [30]. Existem vários tipos de plataformas e maneiras de aplicar SQL.

7.6.3.2 Microsoft SQL Management Studio

O *Microsoft Management Studio* é um programa que serve para gerir, configurar e administrar a infraestrutura do SQL Server. Este contém uma *interface* de utilizador e um grupo

de ferramentas com editores *script*. O *Microsoft SQL Server* consegue gerir bases de dados relacionais. Algumas das funções deste programa são [31]:

- *Triggers*: permitem correr código em relação a eventos.
- *Stored Procedures*: permitem criar métodos definidos pelo utilizador para uso e repetição.
- SQL *User Functions*: funções de SQL definidas pelo utilizador.

7.6.3.3 Segurança em *Microsoft SQL Management Studio* e MD5

Utilizando o programa *Microsoft SQL Management Studio*, o acesso à base de dados é apenas permitido a utilizadores que possuam as credenciais corretas. O programa permite o uso de vários modos de autentificação. Na aplicação exemplo descrita no capítulo de desenvolvimento selecionou-se o modo de *Windows Authentication*. Este método de autentificação utiliza as credenciais Windows para realizar o *login* [32]. As credenciais do Windows são *usernames* e *passwords* usadas para fazer *login* em comunicação de rede baseadas no Windows com acesso a serviços de Web que utilizem *Windows Authentication* e com conexões remotas de *desktop*/servidor de terminal [32].

Qualquer aplicação que possua um sistema de *logins* tem a necessidade de guardar as credenciais numa base de dados, requerendo maior segurança no campo de *passwords*. Uma ferramenta que permite um maior nível de segurança neste campo será o MD5.

O MD5 é um algoritmo de encriptação irreversível, ou seja, o valor calculado pelo MD5 não irá devolver o valor inicial em caso de algoritmo invertido [33]. Este método evita que o administrador do sistema obtenha as *passwords* e aumenta a dificuldade de descobrir *passwords* em caso de falha do sistema de *login* Windows [33].

7.7 Tipo de *Design*

É também necessário decidir qual o tipo de *design* que deverá ser utilizado. Tendo em conta o tipo de aplicação escolhida, existem duas hipóteses:

***Adaptive Design*:** utiliza o *server* para detetar o dispositivo móvel onde a aplicação web está a ser executada e utiliza vários *templates* dependendo do dispositivo em causa. Esta abordagem utiliza HTML5 e CSS próprio para cada aparelho diferente [9]. A criação de código próprio por tipo de aparelho leva à necessidade de consumo de mais recursos. Além disso o código fica

mais extenso e a criação de novos tipos de ecrãs irá implicar criação de novas especificações HTML e CSS [9].

Responsive Design: Este tipo de *design* apareceu em 2010 e a sua adoção tem sido cada vez mais rápida devido à sua escalabilidade e adaptação superior. Este *design* permite a criação de um código único para todos os ecrãs e plataformas existentes em vez de um código específico para cada tipo de ecrã [34]. Este tipo de *design* utiliza CSS específico para ser aplicado num ficheiro de HTML, ou seja, existe um ficheiro de HTML que se adapta fluidamente ao ecrã em questão [9]. "Para entender esta abordagem, o conteúdo pode ser comparado à água porque preenche o espaço do navegador da mesma forma" [35].

Figura 2 Content is like Water ***Fonte****: [35]*

Adaptive design permite que uma aplicação móvel mude dependendo do ecrã em questão. Este tipo de *design* implica a criação de vários conjuntos de código HTML e CSS para utilização dependendo do dispositivo em questão [9]. Por outro lado, *responsive design* permite a utilização de um ficheiro HTML que se irá adaptar a qualquer ecrã, fazendo uso de CSS [9]. A maior desvantagem do *design* responsivo é o desempenho [36]. Este tipo de *design* tende a criar aplicações mais lentas. A utilização de CSS e Javascript especializado criam maior lentidão ao correr o código. A utilização de ecrãs de baixa resolução para mostrar imagens de alta resolução aumenta os problemas de velocidade [37]. Embora isto seja verdade também devemos ter em conta que construir um *site* único web para múltiplos dispositivos é vantajoso para a experiência do utilizador e para o *search engine optimization* (SEO) [36]. Isto porque apenas um URL é observado e desta forma toda a informação é transmitida [36].

Responsive design é formado por três componentes: *fluid grid*, *flexible images* e *media queries* [35].

1. ***Fluid grid*** consiste em mudar o tamanho de elementos através de percentagens em vez de pixéis ou pontos [35]. Isto é possível devido ao uso de comandos que atribuem uma percentagem a elementos de código HTML [35]. O cálculo destas percentagens é feito como visto na seguinte imagem:

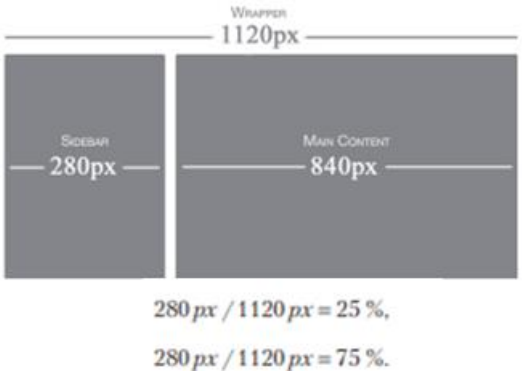

Figura 3 Lógica para Fluid Grid ***Fonte****: [35]*

2. ***Flexible images*** consiste no redimensionamento de imagens com o objetivo de as manter dentro do seu elemento de HTML e manter a sua plenitude. Isto é feito através de código de CSS e com uso de percentagens [35]. Por exemplo o código seguinte faz com que uma imagem seja tão grande como o elemento onde está inserida.

```
img {
max-width: 100 %;
}
```

Figura 4 Código para flexible Images ***Fonte****: [35]*

3. ***Media queries*** consistem em especificações de CSS3 que permitem a construção de vários *layouts* no mesmo documento HTML [38]. O CSS3 é capaz de identificar tipos de ecrã e características físicas do dispositivo e do navegador, por exemplo, a largura do navegador [38]. Folhas de estilo são utilizadas seletivamente com base nas características do dispositivo e do navegador. A característica de ecrã mais referenciada é a largura [38]. A figura 4 mostra um exemplo de uma CSS script que irá variar consoante a largura de ecrã. A utilização de largura máxima e mínima serve para englobar ecrãs em grupos de tamanho.

```
@media screen and (min-width: 800px) {
/* ...large-screen styles here... */
}

/* mobile styles */
@media screen and (min-width: 500px) {
/* tablet styles */
}
@media screen and (min-width: 800px) {
/* desktop styles */
}
```

Figura 5 Codigo CSS para media queries ***Fonte****: [35]*

7.7.1 *Responsive Design* em utilizadores Cegos

Aplicações, *websites* e outros serviços web assumem atualmente elevada importância na comunicação e no quotidiano dos cidadãos. A web revela-se igualmente importante para a integração social e ocupacional de pessoas com necessidades especiais, nomeadamente para indivíduos com deficiência visual. No entanto, indivíduos com necessidades especiais podem, muitas vezes, ter dificuldades em utilizar estes serviços web, dificuldades estas que resultam essencialmente do incumprimento de diretrizes de acessibilidade [39]. Apesar das diretrizes de acessibilidade serem um requisito legal na maioria dos países, muitos serviços web têm barreiras que impossibilitam essa utilização [39]. Para além do incumprimento das diretrizes de acessibilidade existe também o facto de estas diretrizes, muitas vezes, não garantirem a acessibilidade a todos os utilizadores possíveis. De referir igualmente, que em alguns casos, existem serviços aos quais as pessoas com necessidades especiais podem aceder, mas são ineficientes o que pode levar a experiências de utilizador negativas [39]. *Responsive design*, como visto anteriormente, é cada vez mais utilizado por empresas e programadores para criar serviços web. No entanto, é necessário perceber, através do estudo e da observação, como este novo estilo de *design* pode afetar as pessoas com necessidades especiais. Para esse efeito, Tiago C. Nogueira em 2017 decidiu fazer um estudo do impacto de *responsive design* em utilizadores cegos [39].

O estudo consistiu em selecionar utilizadores cegos e submetê-los a serviços com *responsive design* e *non-responsive design* e observar as variações das emoções dos utilizadores [39]. Foi efetuada a separação em emoções positivas e negativas e com os resultados criou-se um valor médio dessas emoções. Os resultados em forma de gráfico foram estes:

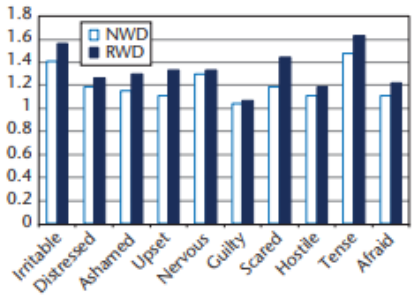

Figura 6 Comparação entre responsive e non-responsive design em emoções negativas. ***Fonte****: [39]*

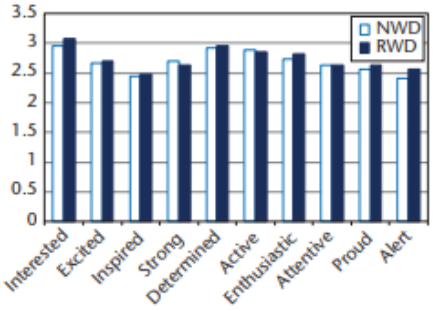

Figura 7 Comparação entre responsive e non-responsive design em emoções positivas. ***Fonte*** *[39]*

Com base no estudo efetuado [39] foi observado o seguinte:

- As emoções dos utilizadores foram mais intensas em *responsive design*. Verificou-se, durante a interação com o serviço, médias mais altas de reações negativas e positivas.

- Os problemas de utilizador estão muitas vezes relacionados com eficiência e eficácia nos serviços web.

- As barreiras encontradas, em alguns casos estavam diretamente relacionadas com as emoções dos utilizadores. Por exemplo à medida que o número de barreiras de navegação aumentava o utilizador tornava-se mais angustiado, envergonhado, perturbado, nervoso, tenso, assustado e zangado. Os *designs* responsivos estudados neste estudo revelaram barreiras emocionais intensas que diminuem a aceitação destes serviços para utilizadores cegos.

É importante também referenciar que estes serviços web seguem o WCAG 2.0 e que mesmo assim apresentam problemas para utilizadores com necessidades especiais.

Assim podemos chegar à conclusão que o *design* responsivo terá de continuar a ser desenvolvido e a evoluir antes de se tornar num *design* global [39].

7.7.2 Princípios de *Design* aplicados ao *responsive design*

Ao longo dos anos, diversos autores analisaram e descreveram, o que seria para eles, os princípios de *design* para *interfaces* informáticas [40]. Das várias abordagens efetuadas, existem três autores que se destacam. Estes são Nielsen em 1990, Galitz em 2002 e Preece em 1994 [40]. Em 2015 realizou-se um estudo para descobrir se o *responsive design* estaria em conformidade com os princípios definidos por estes indivíduos [40]. Primeiramente o estudo decidiu olhar para a perspetiva destes três autores e unificá-la através do uso de palavras-chave comuns. Esta unificação pode ser observada na seguinte tabela.

Scholars	Principles Suggested	Common keywords
10 general principles of user interface design for heuristics evaluation (Nielsen, 1990)	1. Visibility 2. Familiarity 3. User constraint and freedom 4. Consistency 5. Error prevention 6. Recognition 7. Flexibility 8. Aesthetically pleasing 9. Efficiency Feedback 10. User Task	Consistency Familiarity Flexibility Efficiency Feedback Aesthetically pleasing
The basic principles of interface design (Galitz, 2002)	1. Aesthetically pleasing 2. Clarity 3. Compatibility 4. Efficiency Feedback 5. Consistency 6. Familiarity 7. Flexibility	
General principles for interface design (Preece, 1994)	1. Simplicity 2. Visibility 3. Efficiency Feedback 4. Consistency 5. Affordances 6. Tolerance	

Figura 8 Princípios de Design unificados ***Fonte****: [40]*

De acordo com o estudo efetuado, chegou-se a cinco princípios comuns de *design*: 1) Consistência- consiste em evitar que os utilizadores possam pensar que palavras, situações ou ações diferentes significam a mesma coisa; 2) Familiaridade- como princípio diz-nos que será importante usar elementos de *design* que são parecidos com outros elementos, de outras *interfaces* ou da mesma, se estes realizarem a mesma função; 3) Flexibilidade- consiste em fornecer aceleradores que são invisíveis para novos utilizadores, mas que permitem utilizadores mais experientes realizar tarefas mais rapidamente; 4) *Feedback* eficiente- significa que o utilizador recebe *feedback* eficiente quando realiza uma ação na *interface*, ou seja, o utilizador deve sempre perceber o que acontece na *interface* em que está a realizar tarefas; 5)

Esteticamente agradável -significa que a *interface* deve ser agradável ao olho e não deve ter elementos confusos.

De seguida foram selecionadas seis *interfaces* que utilizavam *design* responsivo e que pertenciam a empresas com algum sucesso. O estudo selecionou a *interface* das empresas Zalora, Ebay, Tesco, Shopkissan, Rakuten e Superbuy [40]. Os resultados do estudo podem ser observados na seguinte tabela:

Bil	Website	Desktop	Mobile Device	Principles	
1.	zalora.com			1.Consistency	✓
				2. Familiarity	✓
				3. Flexibility	✓
				4. Feedback	✓
				5. Aesthetic	✓
2.	ebay			1.Consistency	✓
				2. Familiarity	✓
				3. Flexibility	
				4. Feedback	
				5. Aesthetic	✓
3.	tesco			1.Consistency	✓
				2. Familiarity	
				3. Flexibility	✓
				4. Feedback	
				5. Aesthetic	✓
4.	shopkissan dteff. com			1.Consistency	✓
				2. Familiarity	✓
				3. Flexibility	
				4. Feedback	
				5. Aesthetic	✓
5.	rakuten			1.Consistency	✓
				2. Familiarity	✓
				3. Flexibility	✓
				4. Feedback	✓
				5. Aesthetic	✓
6.	superbuy			1.Consistency	✓
				2. Familiarity	✓
				3. Flexibility	✓
				4. Feedback	
				5. Aesthetic	✓

Figura 9 Estudo de Principios de Design ***Fonte****: [40]*

Analisando este estudo, podemos concluir que é possível o uso de *design* responsivo para satisfazer os princípios de *design* (caso cinco e um). Podemos verificar também que os elementos mais difíceis de cumprir são o *feedback* e a flexibilidade. Isto porque o *feedback* falhou quatro das seis *interfaces* e a flexibilidade falhou em duas das seis *interfaces*. Houve também uma *interface* que falhou em familiaridade, mas este pode apenas ser um *outlier*. Em

conclusão, é possível utilizar *responsive design* e seguir os princípios de *design* de *interfaces*, mas é importante ter atenção em relação ao nível de flexibilidade e *feedback*.

7.8 Casos de estudo

No interesse de criar um conhecimento total deste tipo de aplicações decidiu-se analisar casos reais de aplicações web de sucesso com a função de comentar, listar, avaliar e partilhar opiniões sobre filmes. Esta investigação teve como objetivos:

- Identificar quais as semelhanças e diferenças entre as aplicações.
- Analisar estas semelhanças e diferenças para identificar práticas de sucesso.
- Perceber quais as necessidades destas aplicações.

No processo de seleção de aplicações de sucesso foram utilizadas plataformas de distribuição de aplicações. Alguns exemplos destas são a Apple Store e a Play Store. Dentro destas plataformas foram selecionadas as aplicações com melhor classificação que possuam mais de um milhão de *downloads*. No processo de análise das aplicações foi usado o *site* AppGrove [41]. Este *site* possui análises mais complexas de opiniões de utilizadores. Seguindo estes critérios, analisou-se as seguintes cinco aplicações:

- **IMDbd : Your guide to movies, TV shows, celebrities.** Esta aplicação foi criada pela empresa IMDbd. Em julho 2021, possui uma classificação de 4.5 e possui mais de 100 milhões de *downloads*. Esta aplicação possui também a possibilidade de direcionar o utilizador para um programa ou plataforma para ver filmes. Segundo o AppGrove, as melhores características desta aplicação são o suporte a várias linguagens e a informação representada sobre filmes. Os utilizadores expressaram também que as piores características são a informação sobre os atores e a funcionalidade de ver *trailers* de filmes. Uma justificação possível para a existência destas piores características é o facto de a informação sobre os atores ser incompleta e a existência de grande quantidade de *bugs* e avarias na funcionalidade de *trailer*.

- **Movies by Flixster, with Rotten Tomatoes.** Esta aplicação foi criada pela Flixster com a ajuda da empresa Rotten Tomatoes. Em julho 2021, possui uma classificação de 4.1 e mais de 10 milhões de *downloads*. Esta aplicação tem também a possibilidade de direcionar o utilizador para um programa ou plataforma para ver filmes e comprar bilhetes de cinema. Segundo o AppGrove, os utilizadores expressaram que as melhores características desta aplicação, são a informação sobre atores e as funcionalidades de comunidade como comentários e *reviews* de utilizadores. Os utilizadores expressaram também que as piores características são a loja de compra e venda e as recomendações de filmes. Estas piores características estão ligadas a falhas e *bugs* na loja e a um sistema de recomendação que não é representativo do utilizador.

- **Taste: Movie & TV Reviews**. Esta aplicação foi criada pela Taste Labs. Em julho de 2021, possui uma classificação de 4.6 com mais de cem milhões de *downloads*. Esta aplicação não possui uma análise complexa no *site* AppGrove. Ao analisar *reviews* de utilizadores é possível observar que a sua característica única é o uso de uma série de perguntas para reconhecer os tipos de filmes que os utilizadores iriam gostar de ver.

- **Letterboxd.** Esta aplicação foi criada pela LetterBoxd Inc. Em julho de 2021, possui uma classificação de 4.3 com mais de sessenta e três mil de *downloads*. Esta aplicação não possui uma análise complexa no *site* AppGrove. Ao analisar as suas características e *reviews* não foi identificada nenhuma característica única.

- **Fandango Movie Tickets & Times.** Esta aplicação foi criada pela Fadango. Em julho 2021, possui uma classificação de 4.8 com mais de dez milhões de *downloads*. Esta aplicação possui também a possibilidade de direcionar o utilizador para um programa ou plataforma para ver filmes e comprar bilhetes de cinema. Segundo o AppGrove, as suas melhores características são a informação de atores, suporte a várias linguagens e as opiniões de críticos. Os utilizadores expressaram também que as piores características são a loja de compra e venda de filmes e bilhetes e a funcionalidade de ver *trailers* de filmes. Estas piores características estão ligadas a falhas e *bugs* na loja e na funcionalidade de ver *trailers*.

Todas estas aplicações têm as seguintes semelhanças: utilizam um *design* onde inicialmente é apresentado ao utilizador uma lista de filmes e onde o utilizador pode de seguida clicar em

filmes específicos para saber mais informação; todas necessitam de uma ligação à internet para funcionar; todas possuem interações entre utilizadores; todas possuem uma maneira de pesquizar filmes específicos; todas têm páginas de gestão de conta e de observação de outras contas; todas tentam ser informativas sobre os filmes em questão e todas possuem zonas de comentários ou *reviews*. Estas semelhanças parecem ser os requisitos mínimos para criar uma aplicação deste tipo.

Cada uma destas cinco aplicações foi criada com um plano em mente. Movies by Flixster, with Rotten Tomatoes apresenta um maior foco na secção social, a IMBD tenta ser mais informativa, a Taste: Movie & TV Reviews tenta conhecer os gostos do utilizador para recomendar filmes, a LetterBox não possui características únicas, mas apresenta um *design* simples que permite uma secção social e informativa acessível e a Fandango Movie Tickets & Times é mais focalizada em expressar opiniões de críticos. Além disto três das cinco aplicações criaram uma maneira de poder ver filmes a partir da sua *interface*. Este tipo de *feature* requer licenças e compra de direitos de filmes.

8 Metodologia

Este trabalho de projeto seguiu duas metodologias. Para a investigação inicialmente feita optou-se por uma metodologia básica estratégica, descritiva, indutiva e bibliográfica. Cada um destes termos pode ser definido da seguinte forma [42] [43] [44]:

- Básica estratégica- refere-se à finalidade da investigação. Esta investigação não parte de um caso específico, mas tem como objetivo desenvolver conhecimentos que possam ser soluções de problemas conhecidos.
- Descritiva- refere-se ao objetivo da investigação. O objetivo será expor e ligar as características dos objetos de estudo.
- Bibliográfica- refere-se ao procedimento da investigação. Este tipo de procedimento faz uso de livros, artigos e outros textos de caráter científico para reunir informação.
- Indutiva- refere-se ao método da investigação. Neste tipo de método o autor utiliza observações específicas para obter como conclusão uma premissa geral.

Para a criação da aplicação exemplo foi seguida a metodologia *Design Science Research* (DSR). Esta técnica tem como objetivo estudar, pesquisar e investigar artefactos como *software*, métodos, modelos e conceitos [45] [46].

Esta abordagem tem como objetivo criar artefactos que permitem criar soluções para problemas ou limitações. A metodologia DSR apresenta um foco em pesquisa literária que permite aos seus utilizadores fazer decisões informadas [45] [46].

O DSR define um ciclo de etapas para o desenvolvimento da solução abaixo descrita [46]:

- **Consciencialização ou investigação do problema:** Nesta etapa é realizada a compreensão do problema e que limitações e necessidades existem para o projeto.

- **Sugestão ou definição dos resultados esperados:** Considera-se o conjunto de possíveis artefactos e sugere-se a escolha de um, ou mais, a serem usados. No caso deste projeto decidiu-se utilizar as plataformas e tecnologias Microsoft Visual Studio para criar uma aplicação Web.

- **Desenvolvimento da solução:** Procede-se à construção do artefacto. Neste projeto este processo será explicado detalhadamente no capítulo de desenvolvimento.

- **Demonstração:** apresenta-se o funcionamento do artefacto. A demonstração da aplicação será feita no capítulo de desenvolvimento.

- **Validação e Conclusão:** Testa-se e avalia-se se o artefacto resolve o problema inicial.

9 Análise do sistema a desenvolver

Neste capítulo é realizada uma análise de requisitos sobre os serviços que a aplicação a ser desenvolvida deverá garantir. São definidos os requisitos funcionais e não funcionais, bem como os casos de uso e desenho da base de dados. Por último, são também definidas as estratégias selecionadas de desenvolvimento e a sua justificação para a criação do exemplo prático.

9.1 Requisitos funcionais e não funcionais

Neste subcapítulo serão demonstrados os requisitos funcionais e não funcionais da aplicação desenvolvida. Estes requisitos são necessários para a compreensão das funcionalidades e serviços que o sistema a desenvolver deve oferecer.

9.1.1 Requisitos funcionais

Este tipo de requisito define as funções do sistema e os seus componentes. Neste caso são os seguintes:

- ***Login*- Permitir ao utilizador realizar o processo de *login* na aplicação.
- **Registo-** Permitir ao utilizador realizar o processo de registo na aplicação.
- **Revelar o painel de filmes mais bem cotados-** Permitir ao utilizador ver e interagir com o painel de filmes mais bem cotados.
- **Revelar o painel de Perfil próprio-** Permitir ao utilizador ver e interagir com informação relacionada com as suas atividades na aplicação.
- **Procurar um filme-** Permitir ao utilizador pesquisar um filme dado um conjunto de filtros de pesquisa.
- **Revelar o painel de filme-** Permitir ao utilizador ver e interagir com informação relacionada ao filme selecionado.
- **Revelar o painel de não Perfil próprio-** Permitir ao utilizador ver e interagir com informação relacionada com atividades de outros utilizadores na aplicação.
- **Comentar-** Permitir que o utilizador exponha a sua opinião, em forma de texto, sobre um filme.
- **Classificar-** Permitir que o utilizador exponha a sua opinião, em forma de classificação numérica, sobre um filme.
- ***Like* e *dislike*-** Permitir que o utilizador exponha a sua opinião, em forma de *like* ou *dislike*, sobre um comentário de outro ou do próprio utilizador.

9.1.2 Requisitos não funcionais

Requisitos não funcionais servem para expor quais os critérios usados para avaliar as operações de um sistema. Estes requisitos consistem em:

- **Usabilidade-** A *interface* deverá ser intuitiva e deverá ser fácil aprender a utilizar as várias funcionalidades da aplicação.
- **Fiabilidade-** Esta aplicação deverá ter o menor número possível de falhas na sua utilização.
- **Portabilidade-** A aplicação deve ser acessível em qualquer dispositivo.
- **Segurança-** A base de dados desta informação deverá possuir algum nível de segurança.

9.2 Casos de Uso

Antes de iniciar o processo de criação da aplicação foi necessário reconhecer as necessidades e requisitos da mesma. Depois deste reconhecimento, foram selecionadas as funcionalidades e tecnologias da aplicação. Para uma melhor compreensão desta aplicação foram criados os seguintes casos de uso:

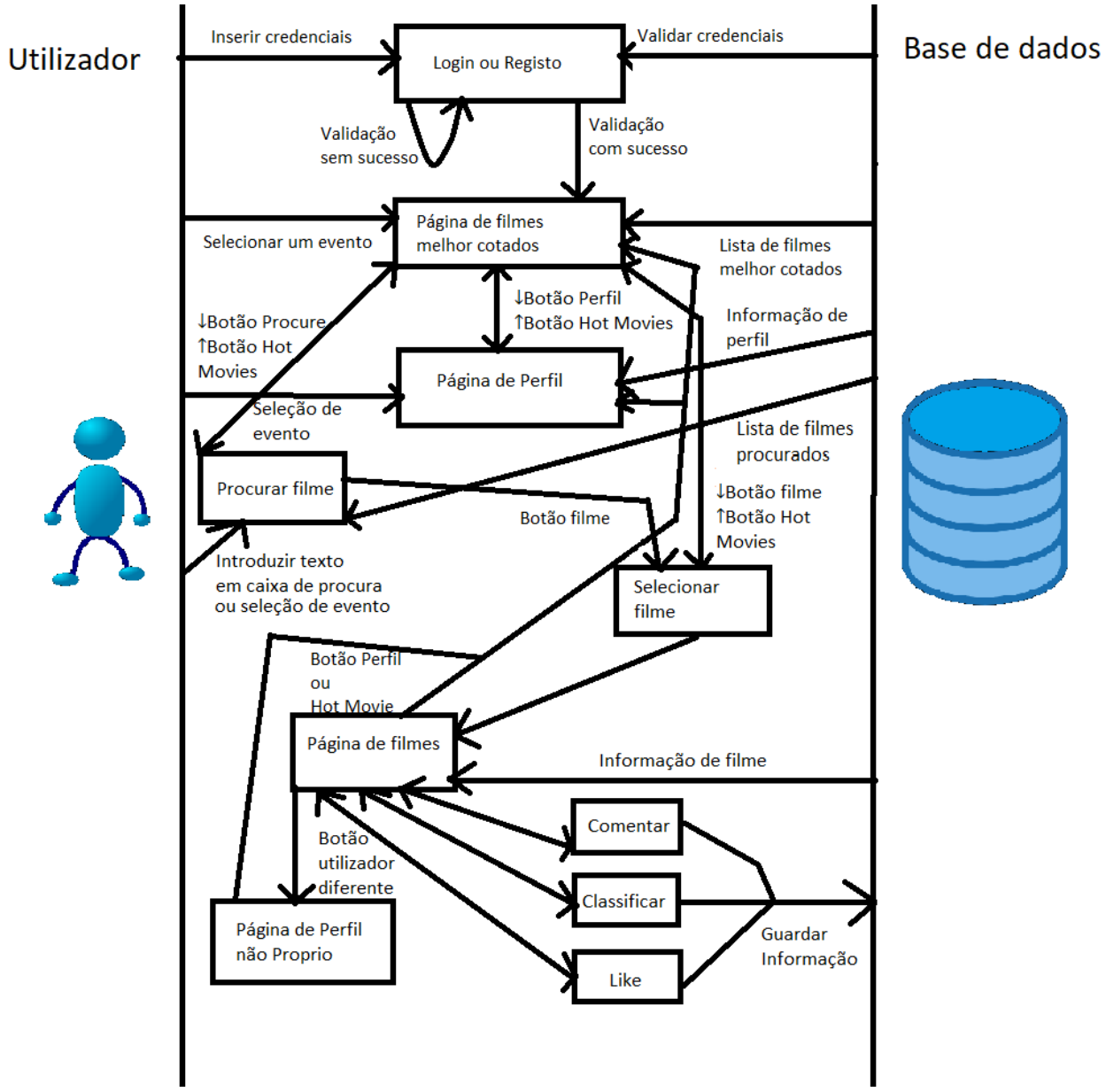

Figura 10 Use Case ***Fonte: autor***

A imagem acima representada pela Figura 10, pode ser descrita pelas seguintes tabelas:

Descrição: *Login* ou Registo
Pré-condições: Nenhuma
Fluxo de ações:
1- Aceder à aplicação. 2- Introduzir credenciais de *login* ou registo. 3- Clicar no botão de *login* ou registo.

Resultado Previsto:	Após o passo dois do fluxo de ações, o utilizador ganha acesso à aplicação.
Estado Erro:	Este processo só dará erro em caso de introdução incorreta de credenciais.

Figura 11 Caso de uso Login ou Registo ***Fonte: autor***

Descrição: Página de Filmes mais bem cotados	
Pré-condições: *Login* ou registo realizado	
Fluxo de ações:	
1- Aceder à aplicação. 2- Introduzir credenciais de *login* ou registo. 3- Clicar no botão de login ou registo ou clicar no botão *Hot Movies*. 4- Recolher a informação de base de dados, relativa a este painel.	
Resultado Previsto:	Após o passo quatro do fluxo de ações, o utilizador ganha acesso ao painel de filmes mais bem cotados.
Estado Erro:	Não aplicável.

Figura 12 Caso de uso Página de Filmes mais bem cotados ***Fonte: autor***

Descrição: Página de Perfil
Pré-condições: *Login* ou registo realizado
Fluxo de ações:

1- Aceder à aplicação. 2- Introduzir credenciais de *login* ou registo. 3- Clicar no botão de *login* ou registo. 4- Clicar no botão Perfil. 5- Recolher a informação de base de dados, relativa a este painel.	
Resultado Previsto:	Após o passo cinco do fluxo de ações, o utilizador ganha acesso ao painel de perfil.
Estado Erro:	Não aplicável.

Figura 13 Caso de uso Página de Perfil ***Fonte: autor***

Descrição: Procurar Filme	
Pré-condições: *Login* ou registo realizado	
Fluxo de ações:	
1- Aceder à aplicação. 2- Introduzir credenciais de *login* ou registo. 3- Clicar no botão de *login* ou registo. 4- Introduzir valores nos campos de procura. 5- Recolher a informação de base de dados, relativa à procura do utilizador. 6- Clicar no botão Procure.	
Resultado Previsto:	Após o passo seis do fluxo de ações, o utilizador consegue observar o resultado da sua procura.
Estado Erro:	Este processo só dará erro em caso de introdução incorreta de valores nos campos de procura.

Figura 14 Caso de uso Procurar Filme ***Fonte: autor***

Descrição: Página de filmes
Pré-condições: *Login* ou registo realizado.

Fluxo de ações:	
1- Aceder à aplicação. 2- Introduzir credenciais de *login* ou registo. 3- Clicar no botão de *login* ou registo. 4- Clicar no nome do filme desejado.	
Resultado Previsto:	Após o passo quatro do fluxo de ações, o utilizador ganha acesso ao painel de filme.
Estado Erro:	Não aplicável.

Figura 15 Caso de uso Página de films ***Fonte: autor***

Descrição: Comentar	
Pré-condições: *Login* ou registo realizado e selecionar um filme.	
Fluxo de ações:	
1- Aceder à aplicação. 2- Introduzir credenciais de *login* ou registo. 3- Clicar no botão de *login* ou registo. 4- Clicar no nome do filme desejado. 5- Recolher a informação de base de dados, relativa ao filme e questão. 6- Escrever um comentário na caixa de comentários. 7- Clicar no botão de comentar. 8- Guardar comentário na base de dados.	
Resultado Previsto:	Após o passo oito do fluxo de ações, o utilizador comenta um filme.
Estado Erro:	Mensagem de erro se o campo estiver vazio.

Figura 16 Caso de uso Comentário ***Fonte: autor***

Descrição: Classificar	
Pré-condições: *Login* ou registo realizado e selecionar um filme.	
Fluxo de ações:	
1- Aceder à aplicação. 2- Introduzir credenciais de *login* ou registo. 3- Clicar no botão de *login* ou registo. 4- Clicar no nome do filme desejado. 5- Recolher a informação de base de dados, relativa ao filme e questão. 6- Selecionar um valor de classificação. 7- Clicar no botão *submit*. 8- Guardar valor de classificação na base de dados.	
Resultado Previsto:	Após o passo oito do fluxo de ações, o utilizador classifica um filme.
Estado Erro:	Não aplicável.

Figura 17 Caso de uso Classificar ***Fonte: autor***

Descrição: *Like* ou *Dislike*	
Pré-condições: *Login* ou registo realizado e selecionar um filme.	
Fluxo de ações:	
1- Aceder à aplicação. 2- Introduzir credenciais de *login* ou registo. 3- Clicar no botão de *login* ou registo. 4- Clicar no nome do filme desejado. 5- Recolher a informação de base de dados, relativa ao filme em questão. 6- Clicar no botão de *like* ou *dislike*.	
Resultado Previsto:	Após o passo seis do fluxo de ações, o utilizador atribui um *like* ou *dislike* a um comentário.
Estado Erro:	Não aplicável.

Figura 18 Caso de uso Like ou dislke ***Fonte: autor***

Descrição: Página de Perfil não próprio	
Pré-condições: *Login* ou registo realizado e selecionar um filme.	
Fluxo de ações:	
1- Aceder à aplicação. 2- Introduzir credenciais de *login* ou registo. 3- Clicar no botão de *login* ou registo. 4- Clicar no nome do filme desejado. 5- Recolher a informação de base de dados, relativa ao filme em questão. 6- Clicar no nome do utilizador pretendido.	
Resultado Previsto:	Após o passo seis do fluxo de ações, o utilizador ganha acesso ao painel de perfil não próprio.
Estado Erro:	Não aplicável.

Figura 19 Caso de uso Página de Perfil não próprio ***Fonte: autor***

Como é possível observar nos casos de uso, ao abrir a aplicação os utilizadores terão de realizar o *login* ou registo. Depois de realizar um destes processos com sucesso, os utilizadores irão encontrar a página ou painel de filmes mais bem cotados ou *Hot Movies*. Este painel recebe uma lista de filmes da base de dados e possui três eventos acionáveis pelos utilizadores. O primeiro evento depende do botão Perfil e permite entrar no painel de Perfil. Este painel receberá informação sobre o utilizador e irá expor essa informação. A partir deste painel é também possível aceder ao painel *Hot Movies* através do botão *Hot Movies*. O segundo evento depende do botão Procure e consiste em procurar e revelar filmes com as características indicadas pelo utilizador. Por fim existe o evento selecionar filme. Este evento é acionado quando utilizadores clicam num nome de um filme. Ao acionar este evento é revelado o painel de filmes. Este painel irá receber informação específica desse filme da base de dados. Dentro deste painel é possível aceder ao painel de Perfil, ao painel *Hot Movies* e é possível acionar quatro eventos. Os primeiros três eventos servem apenas para guardar informações como comentários, classificações e *likes* do utilizador. O último evento revela ao utilizador o painel de Perfil não próprio. Neste painel é possível observar informações sobre outros utilizadores e atribuir *likes* e *dislikes* a comentários desse utilizador. Este painel possui também acesso ao painel de perfil e ao painel de *Hot Movies*.

Falta apenas referenciar que esta aplicação terá um painel de administrador que será capaz de eliminar contas e eliminar e adicionar filmes.

9.3 Desenho da Base de Dados

Para a aplicação desenvolvida foi usada uma base de dados relacional dado que irá fazer uso de pesquisas de alta precisão e beneficiar de um modelo de dados normalizado.

O modelo de classes da base de dados é apresentado na figura em baixo:

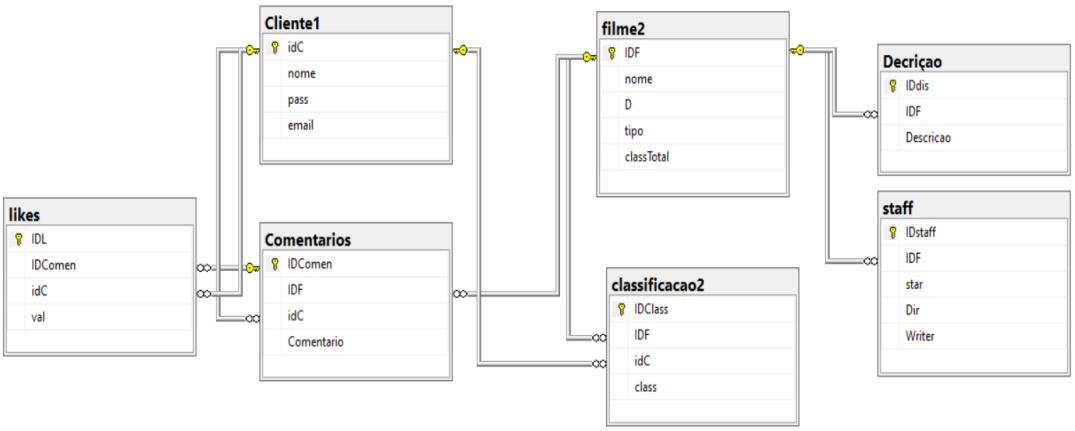

Figura 20 Base de dados ***Fonte: autor***

9.4 Decisão técnica sobre as estratégias de desenvolvimento

Este capítulo demonstra as tecnologias que foram selecionadas na criação da aplicação desenvolvida no capítulo de desenvolvimento. No capítulo 7.2 estabeleceu-se que esta aplicação seria uma aplicação web. De seguida tornou-se necessário decidir qual a plataforma que se deveria utilizar.

Foram analisadas as plataformas Appery, Swing2App e o Visual Studio. Decidiu-se escolher o Visual Studio. Esta plataforma foi selecionada devido aos seguintes fatores:

- Visto que o criador da plataforma será um programador o Visual Studio é a opção com maior liberdade, isto porque é livre de modelos e sistemas de arraste.
- Das opções verificadas é a que possui um custo monetário inferior, embora a versão Pro desta plataforma tenha um custo.
- O fator mais importante na seleção foi o uso de tecnologias .Net.

O tipo de *design* selecionado foi o responsivo devido à sua escalabilidade e custos reduzidos. Este tipo de *design* vai criar uma aplicação que irá funcionar em todos os ecrãs com menos esforço. "O *design* responsivo é uma abordagem que cria verdadeiro valor de negócio: economiza tempo e esforço para desenvolvedores, possuí maiores taxas de adoção entre os

utilizadores" [47]. Foi também necessário selecionar quais as linguagens e *frameworks* de suporte à aplicação. O *front-end* foi construído fazendo uso de HTML, CSS e Bootstrap. Estas tecnologias foram selecionadas porque permitem o *design* responsivo. O *back-end* foi construído fazendo uso de .Net. A *framework MVC* seria apropriada para desenvolver este tipo de aplicação, no entanto, não foi selecionada devido à sua maior complexidade de compreensão e à necessidade de regras mais rígidas para métodos. Mesmo assim, transformar a aplicação descrita no desenvolvimento numa aplicação em MVC é perfeitamente possível. Seria necessário dividir o ficheiro de HTML por várias *views*, criar modelos de classes para chamar dados da base de dados e inserir os métodos do *code behind* em controladores.

A aplicação em questão deverá possuir uma base de dados relacional. Isto porque esta aplicação irá fazer uso de pesquisas de alta precisão e irá beneficiar de um modelo de dados normalizado (tipos de dados a guardar nunca se irão alterar). Assim foi, então, necessário selecionar qual a melhor plataforma para criar um servidor com uma base de dados. Foi escolhida a plataforma *Microsoft Management Studio*, em virtude de esta possuir uma ligação simples *built-in* com a plataforma *Visual Studio*. A aplicação que se pretende desenvolver possui um sistema de *logins*, ou seja, a entrada na aplicação irá necessitar de uma combinação de *username* e *password*. Para a aplicação conseguir verificar as credenciais do utilizador, esta irá precisar de guardar *usernames* e *passwords* na base de dados. Assim torna-se necessário tornar a *password* secreta e mais segura. Para este efeito decidiu-se utilizar MD5 para encriptar as *passwords* na base de dados.

10 Descrição do Sistema

Neste capítulo será apresentado a arquitetura do sistema, a estrutura do sistema e o desenvolvimento e conexão da base de dados.

10.1 Arquitetura do sistema

Este sistema possui três intervenientes. O primeiro é o utilizador. Um utilizador interage diretamente com a *interface*. Esta interação acontece através de eventos como, por exemplo, clicar num botão. O segundo é a aplicação. A aplicação possui um *front-end* e um *back-end*. O front-end expõe informação ao utilizador e permite interação com a *interface*. O *back-end* irá

extrair e guardar informação na base de dados e mudar a *interface* consoante os pedidos do utilizador. O último interveniente é a base de dados. Este possui informação que deverá ser usada na aplicação. Para além disto também possui métodos de procura de informação específica (em forma de funções de SQL).

10.2 Estrutura do sistema

O *Microsoft Visual Studio* permite uma gestão fácil e eficaz da estrutura da aplicação. Esta estrutura pode ser observada no *Solution Explorer* do programa. As diretorias e ficheiros que merecem maior destaque em relação à estrutura da aplicação, são os seguintes:

- CSS- Esta diretoria possui ficheiros responsáveis pelo estilo da aplicação. Entre estes podemos encontrar variados recursos visuais.
- Pasta *Content*- Esta pasta possui as especificações de Bootstrap e imagens que foram utilizadas na aplicação.
- Ficheiro *Default*- Aqui podemos encontrar o ficheiro HTML da *interface* e o *back-end* em c#.Net. Este ficheiro está dividido em três partes. O *Default.aspx* possui o ficheiro de HTML, o *Default.aspx.cs* possui o código c#.Net e o *Default.aspx.designer.cs* declara os elementos dinâmicos da *interface* e é construído automaticamente.

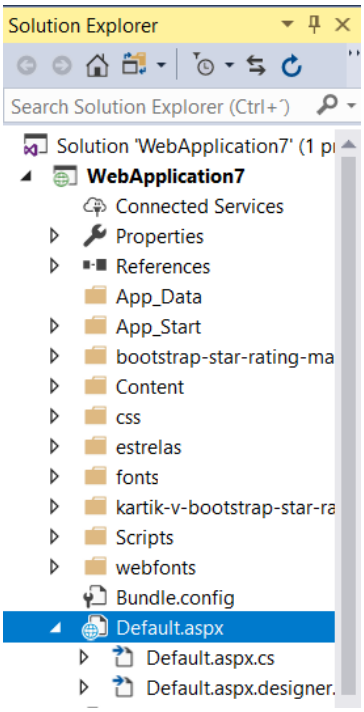

Figura 21 Solution Explorer ***Fonte: autor***

10.2.1 Desenvolvimento da base de dados

A base de dados possui sete tabelas:

- A tabela cliente1 serve para guardar as informações de utilizadores. A tabela possui uma chave primária (idC), uma coluna para guardar todos os *usernames* (nome), uma coluna para guardar *passwords* encriptadas (pass) e uma coluna para guardar emails (email).
- A tabela filme2 serve para guardar informação sobre filmes. A tabela possui uma chave primária (IDF), uma coluna para guardar o nome do filme (nome), uma coluna para guardar a data de lançamento do filme (D), uma coluna com um valor numérico para guardar o tipo filme (tipo) e uma coluna que serve para guardar o cálculo da média de todas as classificações de um certo filme (classTotal).
- A tabela descrição serve para guardar a descrição de cada filme. A tabela possui uma chave primária (IDdis), uma chave estrangeira que referencia a tabela filme2 (IDF) e uma coluna para guardar a descrição em formato de texto (descrição).
- A tabela staff serve para guardar o nome dos indivíduos envolvidos na criação do filme. A tabela possui uma chave primária (IDstaff), uma chave estrangeira que referencia a tabela filme2 (IDF), uma coluna de texto para guardar o nome das estrelas do filme (star), uma coluna de texto para guardar o nome dos diretores do filme (Dir) e uma coluna de texto para guardar o nome dos escritores do filme (Writer).
- A tabela classificação2 serve para guardar as classificações dadas pelos utilizadores a cada filme. A tabela possui uma chave primária (IDClass), uma chave estrangeira que referencia a tabela filme2 (IDF), uma chave estrangeira que referencia a tabela cliente1 (idC) e uma coluna com valor numérico que guarda o valor da classificação dada pelo utilizador (class).
- A tabela comentários serve para guardar comentários feitos por utilizadores sobre filmes. A tabela possui uma chave primária (IDComen), uma chave estrangeira que referencia a tabela filme2 (IDF), uma chave estrangeira que referencia a tabela cliente1 (idC) e uma coluna de texto para guardar os comentários dos utilizadores (comentário).
- A tabela de *likes* serve para guardar os *likes* e os *dislikes* de utilizadores associados a comentários. A tabela possui uma chave primária (IDL), uma chave estrangeira que referencia a tabela cliente1 (idC), uma chave estrangeira que referencia a tabela

comentários (IDComen) e uma coluna com valor 1 (like) ou -1(*dislike*) que guarda o valor do *like* do utilizador (val).

Foi também criada uma *view* chamada MainPage. Esta *view* serve para ordenar todos os filmes por ordem decrescente de média de classificação de utilizadores e criar um índice. Isto é possível com o seguinte código:

```
CREATE view [dbo].[MainPage]
as
SELECT
a.IDF,
a.nome,
avg(b.class) as class, ROW_NUMBER() OVER(ORDER BY avg(b.class)  desc) as Number
FROM [MovieFlame].[dbo].[filme2] a join classificacao2 b on b.IDF =a.IDF
group by a.IDF,a.nome
order by Number
```

Assim é possível selecionar qualquer posição de filme melhor classificado. Um exemplo do uso desta *view* seria selecionar os seis filmes mais bem cotados. Para este efeito seria apenas necessário selecionar todos os valores na *view MainPage* onde a coluna *Number* fosse igual ou inferior a seis. O resultado deste *select* seria:

	IDF	nome	class	Number
1	6	The Shawshank Redemption	5	1
2	7	The Godfather	5	2
3	1004	Shrek 2	4	3
4	9	The Dark Knight	4	4
5	10	12 Angry Men	4	5
6	11	Schindlers List	3	6

Figura 22 Resultado de select na view Main Page ***Fonte****:autor*

10.2.2 Ligação à base de dados

Para criar a ligação segue-se os seguintes passos [48]:

- Clicar no botão *Tools* na barra de ferramentas do programa *Visual Studio*.
- Selecionar a opção *connect to Database*.
- Selecionar a opção *Microsoft Sql Server Database File*.
- Selecionar o nome da base de dados e do servidor.

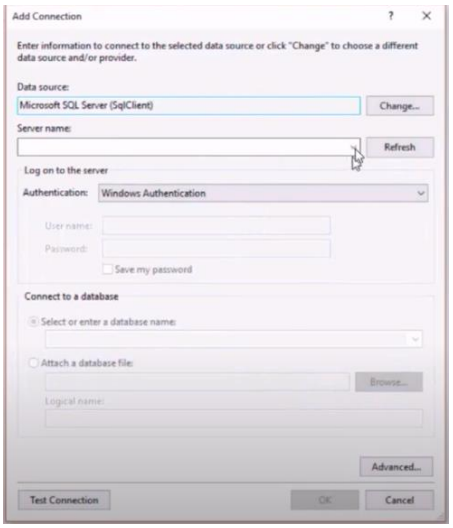

Figura 23 Conexão à base de dados ***Fonte****:autor*

- Criar uma variável que represente a ligação

```
SqlConnection conn = new SqlConnection(@"Data Source = LAPTOP-KECI0SP4\TRABALHOFINAL; Initial Catalog = MovieFlame; Integrated Security = True");
```

- Utilizar o comando *conn.Open*(); e *conn.Close*(); para abrir e fechar a conexão à base de dados.

Por questões de segurança, é importante fechar a conexão no fim de todos os eventos que utilizem a base de dados.

11 Desenvolvimento

Nesta secção de desenvolvimento descreve-se o processo de criação de todos os elementos da aplicação web. Para este efeito, a aplicação foi dividida por painéis e a cada painel foi atribuído um capítulo e dois subcapítulos. O primeiro subcapítulo explica o Html, CSS e Bootstrap enquanto o segundo irá explicar o c# e o SQL utlizado na realização de cada painel.

Foi necessário dividir o CSS por tamanho de ecrã. Para este efeito foram utilizados os tamanhos *default* do Microsoft Visual Studio da seguinte forma:

- Maior ou igual a 1200px de largura.
- Entre 992px e 1199px de largura.
- Entre 768px e 991px de largura.
- Menor ou igual a 767px de largura.

Tamanhos muito superiores a 1200px ou muito inferiores a 767px podem distorcer o *design* da aplicação. Se a aplicação for um dia utilizada com estes valores de largura problemáticos será apenas necessário criar mais divisões de CSS. Isto é possível adicionando à página de estilos o seguinte:

```
@media (min-width: x px) and (max-width: y px) {
    }
```

Os valores de x e y devem ser substituídos pelos valores do ecrã em questão e todas as especificações de elementos da aplicação, para esse tamanho, devem ser adicionados entre parêntesis.

11.1 Login e Register

11.1.1 Html, CSS e Bootstrap

O primeiro elemento que será visível para o utilizador será a caixa de *login* e registo. Estas terão as seguintes formas:

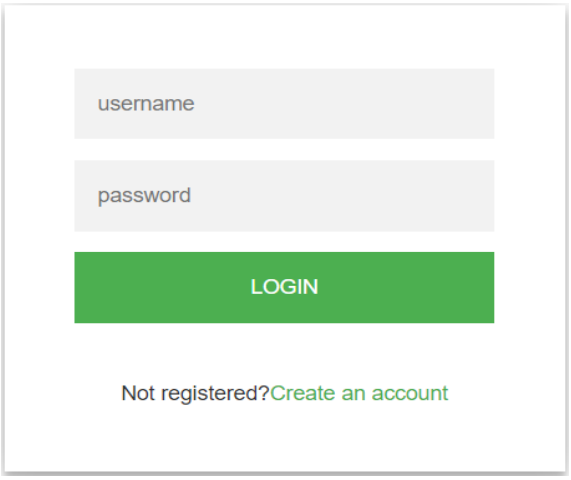

Figura 24 Painel de Login ***Fonte****:autor*

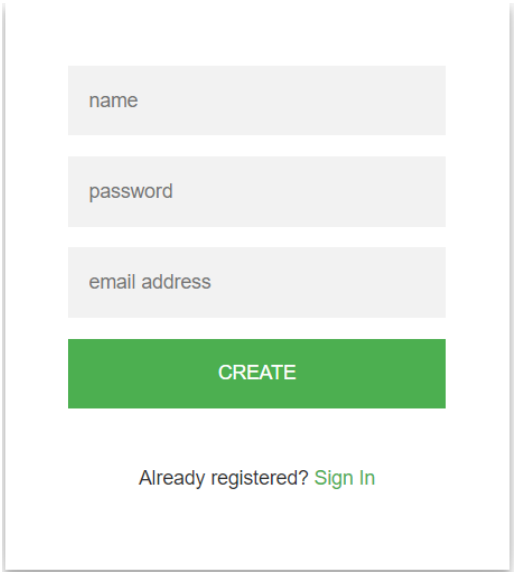

Figura 25 Painel de Registo ***Fonte****:autor*

A primeira imagem servirá para um utilizador realizar o *login* e a segunda servirá para novos utilizadores se registarem. O primeiro passo na criação desta caixa de *login* foi a importação de uma fonte *online*. Esta fonte *online* pode ser encontrada e utilizada no URL https://fonts.googleapis.com/css?family=Roboto:300 e para fazer uso desta é necessário adicionar à página de CSS a seguinte linha:

```
@import url(https://fonts.googleapis.com/css?family=Roboto:300);
```

Este painel de *login* e registo tem o seguinte código Html:

```
<asp:Panel ID="LoginFO" runat="server">
<div class="login-s">
 <asp:Panel ID="LabelR1" Text="Label" runat="server" >
 <div class="register-form" >
<asp:textbox type="text" Cssclass="TXTL" ID="reglog" placeholder="name"  runat="server"
/>
<asp:textbox type="password" Cssclass="TXTL" ID="regpass"  placeholder="password"
runat="server"/>
<asp:textbox type="email" Cssclass="TXTL" ID="regemail"  placeholder="email address"
runat="server"/>
<asp:button Text="create"  OnClick="Register"  Cssclass="LBT"
runat="server"></asp:button><hr>
<asp:Label  ID="erroreg"  Text=""  CssClass="erro"  runat="server"></asp:Label>
```

```
<p class="message">Already registered? <asp:LinkButton ID="LinkButton2" runat="server"
OnClick="LOGc" >Sign In</asp:LinkButton></p>
</div>
</asp:Panel>
<asp:Panel ID="LabelL1" Text="Label" runat="server">
<div class="login-form">
<asp:textbox type="text" Cssclass="TXTL" ID="Userlog" placeholder="username"
runat="server"/>
<asp:textbox type="password" Cssclass="TXTL" ID="passlog" placeholder="password"
runat="server"/>
<asp:button Text ="Login" Cssclass="LBT"  OnClick="Login"
runat="server"></asp:button><hr>
<asp:Label  ID="errolog"  Text="Label" Visible="false" CssClass="erro"
runat="server">Credenciais Incorrectas</asp:Label>
<div class="message">Not registered?<asp:LinkButton ID="LinkButton3"  runat="server"
OnClick="RegC">Create an account</asp:LinkButton> </div>
</div></asp:Panel> </div> </asp:Panel>
```

O painel de registos com o id R1 começa escondido. O painel de *logins* irá ser escondido quando mudamos para a janela de registos. Todo este código existe dentro de um painel que irá ser escondido quando o utilizador realizar o *login* ou registo. O *design* responsivo é garantido através de controlo de margem. Esta técnica é uma das técnicas de *design* responsivo que foram utilizadas neste projeto. Outras técnicas irão ser explicadas quando forem utilizadas. A técnica de controlo de margens consiste em utilizar *tags* de CSS como *margin*, *padding*, *position* e outras para controlar o tamanho de margens do elemento. Parte do código CSS que controla este aspeto está escrito da seguinte forma:

```
.login-s {
    width: 360px;
    padding: 8% 0 0;
    margin: auto;
    position: relative;
    z-index: 1;
    max-width: 300px;
    margin: 0 auto;
    position: relative;
```

```
    z-index: 1;
    background: #FFFFFF;
    max-width: 360px;
    margin: 0 auto 100px;
    text-align: center;
}
```

Como é possível observar o *padding* tem um valor percentual o que permite ao código mudar consoante o tamanho de ecrã. Faz-se uso também de *tags* para centrar o texto, tornar a posição da caixa relativa entre outros. Foram utilizadas mais *tags* de CSS, mas estas apenas controlam aspetos visuais, animações e cores.

11.1.2 C# e SQL

Devido à programação ser orientada a eventos torna-se necessário criar uma função para cada evento único que exista na aplicação. Os primeiros dois eventos que foram criados, foram os eventos que tratam de alternar entre as janelas de *login* e registo. Estes eventos acontecem quando o utilizador clica no texto *Create an account* ou *Sign in*. Ao clicar em qualquer um deles altera-se o valor da propriedade *Visibility* para revelar um painel e esconder outro. Ao clicar em *Create an account* o painel de *login* é escondido e o de registo é revelado. Ao clicar em *Sign in* o painel de registo é escondido e o de *login* é revelado. A propriedade *Visibility* será crucial visto que a aplicação tem uma só uma página de html que irá revelar e esconder elementos desejados.

O evento de *login* é acionado quando o utilizador clica no botão de *login*. Quando um utilizador tenta realizar um *login* é necessário verificar se a combinação de utilizador e *password* existem na base de dados. Para isto é necessário primeiro adicionar a biblioteca de SQL. Para fazer isto é apenas necessário adicionar *using System.Data.SqlClient*; no início do código c#. De seguida é necessário criar um leitor que verifique se esta combinação está correta. Decidiu-se resolver esta necessidade com o seguinte código:

```
int sair = 0;
string sql = "select nome, pass from Cliente1";
SqlCommand login = new SqlCommand(sql, conn);
SqlDataReader reader = login.ExecuteReader();
string Paseg = Encrypt(passlog.Text.Trim());
```

```
while (reader.Read() && sair == 0){
 if (Userlog.Text.Trim().Equals(reader["nome"].ToString().Trim()) &&
Paseg.Trim().Equals(reader["pass"].ToString().Trim()))
{
 sair = 1;
  Efeitos de sucesso…
  }
  else{
  Efeitos de erro…
  }}
```

Como se pode observar foi utilizado a função Encrypt. Esta função pertence à biblioteca de System.Security.Cryptography e serve para encriptar a *password* com MD5. Isto é necessário porque as *passwords* na base de dados estão encriptadas em MD5. Em caso de sucesso a variável sair será igual a 1. Quando a variável sair ganha o valor 1, o leitor irá parar e o utilizador poderá desfrutar das funcionalidades da aplicação. Por fim, foi criada uma variável global *Username* que irá guardar o *username* do utilizador. Esta variável mantém o seu valor fazendo uso de *viewstate* tanto na função de Page Load(if (ViewState["Username"] != null)Username=(string)ViewState["Username"];) como na função de Page_Prerender(ViewState["Username"] = Username;).

O evento de registo é acionado ao clicar no botão de *Create*. Neste evento é necessário garantir que o *username* e o *email* escolhido é único na aplicação, que todos os campos de registo foram preenchidos, que o *email* escolhido tem um formato correto e que o *username* e *password* possuem pelo menos cinco caracteres. Em termos de formato é possível mudar o tipo de caixa de texto para o tipo de formato pretendido. É possível também verificar o tamanho do *username* e *password* facilmente através da função *Count*. Para verificar se o *email* e o *username* são únicos será necessário verificar na base de dados se existem entradas iguais. No caso do *username* criou-se o seguinte código:

```
string userinput = reglog.Text;
string sql1 = "select nome from cliente1";
SqlCommand register = new SqlCommand(sql1, conn);
SqlDataReader reader1 = register.ExecuteReader();
while (reader1.Read() && sair1 == 0){
if (userinput.Equals(reader1["nome"].ToString().Trim())){
x = userinput;
```

```
sair1 = 1;
} };
if (userinput.Equals("") || userinput.Count().Equals(4) || userinput.Count().Equals(3) ||
userinput.Count().Equals(2) || userinput.Count().Equals(1) || x == userinput)  {
if (userinput.Equals("")){
Mensagem de erro em caso de campo vazio.
}
else if (x == userinput){
Mensagem de erro em caso de utilizador não único.
}
else{
Mensagem erro em caso de não escrever mínimo de 5 caracteres.
}}
else{
Sucesso em Registo
}
```

Utilizando esta lógica é possível criar as condições de *password* e email. Em caso de sucesso é necessário igualar *o username* à variável global *Username* e guardar o novo utilizador na base de dados. Para guardar o novo utilizador criou-se a *stored procedure* register da seguinte forma:

```
Create proc [dbo].[register] @username char(30) ,@password char(30), @email char(100)
as
begin
insert into cliente1 (nome,pass,email) values (@username,@password,@email)
end
```

Esta *stored procedure* é depois corrida no código da seguinte forma:

```
SqlCommand regist = new SqlCommand("register", conn) {
CommandType = System.Data.CommandType.StoredProcedure,
Connection = conn
 };
string PassC = Encrypt(regpass.Text);
regist.Parameters.AddWithValue("@username", reglog.Text.Trim());
regist.Parameters.AddWithValue("@password", PassC);
regist.Parameters.AddWithValue("@email", emailinput.ToString().Trim());
regist.ExecuteNonQuery();
```

Quando o utilizador acabar de completar o *login* ou o registo, passa a ser visível o painel *Hot Movies*. Assim torna-se necessário, selecionar a informação dos filmes que vão ser demonstrados, esconder a caixa de *login* e registo e revelar o painel de *Hot Movies*. Além disto é necessário ter em mente que os filmes serão mostrados em conjuntos de seis. Assim, inicialmente criou-se a variável Botao que será igualada a zero. De seguida criou-se a seguinte *stored procedure*:

```
Create proc [dbo].[MainPagefun] @var int
as
SELECT nome,class
FROM [MovieFlame].[dbo].[MainPage] where   Number = @var
```

Esta *stored procedure* irá recolher a informação necessária do filme com o Number pretendido. Os resultados são depois demonstrados utilizando a propriedade .Text ou guardados numa variável para uso futuro. Este processo é codificado da seguinte forma:

```
SqlCommand topM = new SqlCommand("MainPagefun", conn){
CommandType = System.Data.CommandType.StoredProcedure,
Connection = conn
};
topM.Parameters.AddWithValue("@var", Botao+1);
string temp = "";
string temp2 = "";
SqlDataReader reador = topM.ExecuteReader();
while (reador.Read()){
temp += reador["nome"].ToString();
temp2 += reador["class"].ToString();
}
L1.Text = temp;
int t1 = Int16.Parse(temp2);
```

A variável t1 vai depois ser usada para definir a classificação do filme em estrelas. Adicionando valores à variável Botao é possível selecionar qualquer posição de filme por classificação, isto porque a *view MainPage* está ordenada por classificação. Também é possível selecionar a informação de filmes em conjuntos de seis e guardar numa matriz. Este método será utilizado mais à frente noutro caso. Utilizando esta *stored procedure* recolhemos toda a informação para revelar o painel de filmes.

11.2 Painel de Filmes

11.2.1 HTML, CSS e Bootstrap

Este painel apresenta as seguintes formas:

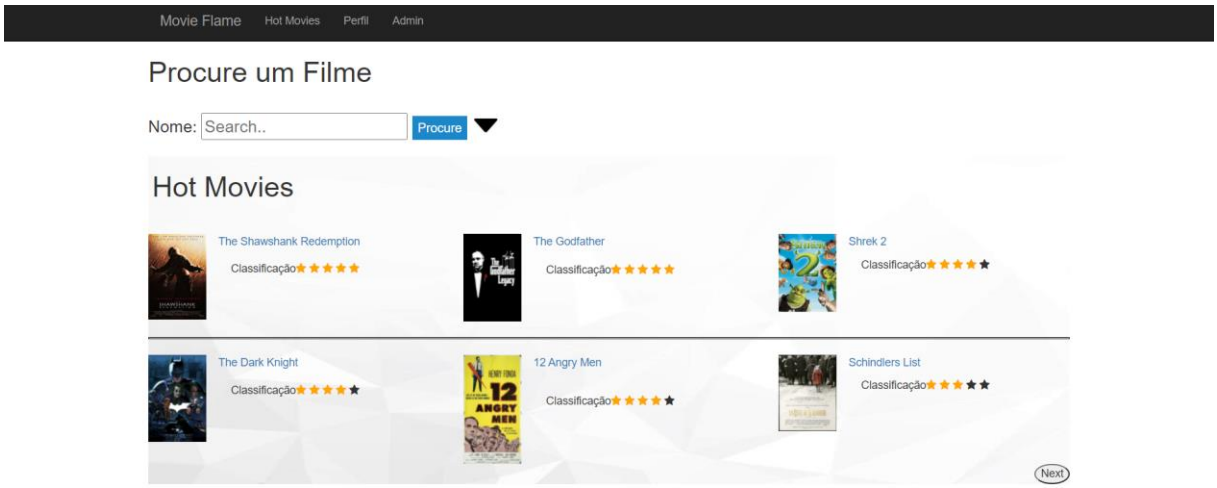

Figura 26 Painel de Filmes com largura de ecrã superior a 1999px ***Fonte****:autor*

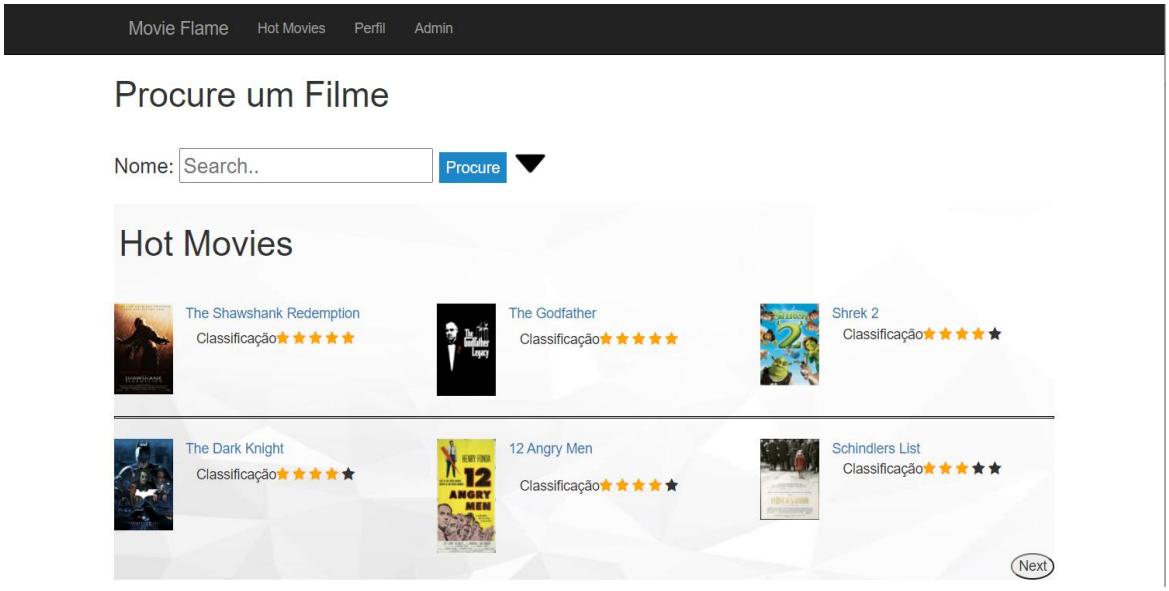

Figura 27 Painel de Filmes com largura de ecrã entre 992px e 1199px ***Fonte****:autor*

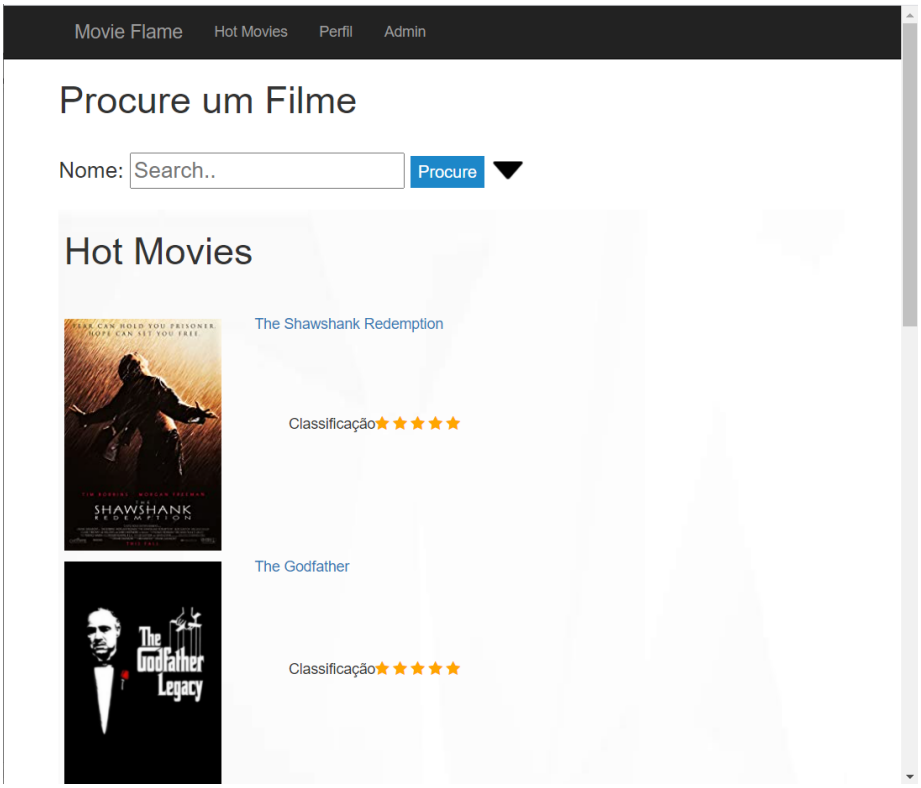

Figura 28 Painel de Filmes com largura de ecrã entre 768px e 991px ***Fonte****:autor*

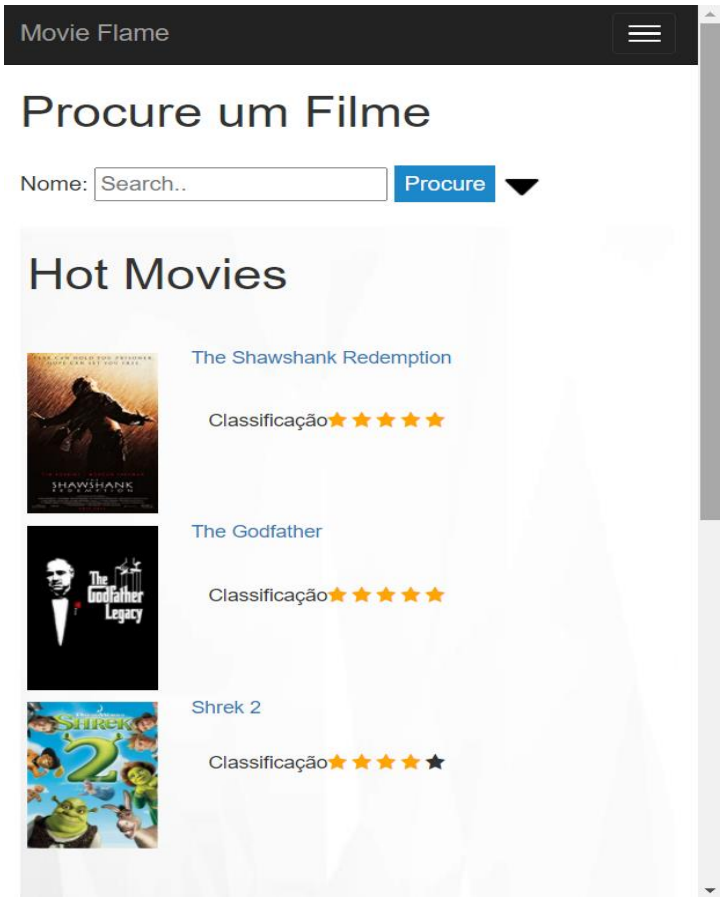

Figura 29 Painel de Fimes com largura de ecrã inferior a 768px ***Fonte****:autor*

Nos quatro casos é possível observar três grandes grupos de elementos. O primeiro destes é a barra de navegação (zona preta no topo do ecrã). Este grupo foi criado automaticamente pelo *Visual Studio* e foi apenas necessário mudar o texto e funções associados aos botões. O segundo grupo é o grupo de procura (zona com o título Procure um Filme). Esta zona contém dois botões. Um botão azul e um botão em forma de seta. Ao clicar no botão em forma de seta serão reveladas zonas de *input* para uma procura mais avançada. Todos os elementos desta zona foram desenhados de forma responsiva. Isto foi possível através da manipulação da largura e altura de elementos desta zona. Por exemplo, ao especificar a largura do botão em forma de seta para ecrãs inferiores a 768px utiliza-se o código:

```
@media (max-width: 767px) {
.Seta {
width: 8%;}}
```

Utilizando a mesma lógica demonstrada em cima é possível manipular todos os elementos desta zona. A terceira e última zona será a zona de filmes. Esta zona foi construída em cima de uma *fluid grid*. Esta grelha tem a seguinte forma:

Figura 30 Fuild Grid da zona de Filmes ***Fonte****:autor*

Como é possível observar esta grelha tem duas linhas cada uma com três colunas. Para criar uma linha é necessário criar uma divisão com a classe de CSS row. Para criar uma coluna é necessário criar uma divisão com classe de CSS col-md-x. O x representa o tamanho da coluna em questão. Todos os valores de x numa linha, quando somados, tem de ser igual a doze. Esta *fluid grid* é criada da seguinte forma:

```
<div id="row1" class="row">
<div class="col-md-4"></div>
<div class="col-md-4"></div>
<div class="col-md-4"></div>
</div>
```

```
<div id="row2" class="row">
<div class="col-md-4"></div>
<div class="col-md-4"></div>
<div class="col-md-4"></div>
</div>
```

Em casos onde é necessário aplicar CSS a uma divisão será impossível utilizar uma classe de CSS, isto porque todas estas divisões já possuem uma classe. Uma forma de resolver este problema é atribuir um id à divisão e aplicar o CSS pelo id em vez de classe. Isto é possível escrevendo um # e de seguida o id da divisão em questão. Foi necessário também a utilização do conceito de *flexible image*. Cada filme está associado a uma imagem promocional e a uma imagem em forma de estrelas. O tamanho destas imagens irá variar dependendo do tamanho do ecrã. Esta variação foi conseguida através da manipulação de altura e largura (mesmo processo utilizado no botão em forma de seta explicado em cima). As estrelas observadas nesta zona foram retiradas do *website* Font Awesome [49]. Estas estrelas têm dois estados: cheias ou vazias. Dependendo das classificações dos utilizadores guardadas na base de dados, a aplicação irá apresentar mais ou menos estrelas no estado cheio. Nas figuras 26 e 27 é possível observar uma linha preta horizontal que divide as duas linhas da *Fluid Grid*. Quando corremos esta aplicação num ecrã inferior a 991px será necessário retirar esta linha e o espaço que ela cria. Esta linha existe devido ao uso da *tag* de Html <hr id="Divider">. Através de CSS atribuímos uma linha com o tamanho de 1px de altura:

```
.divider {
  height: 1px;
  margin: 9px 0;
  overflow: hidden;
  background-color: #e5e5e5;
}
```

Para esconder este hr será apenas necessário, na zona de tamanhos inferiores a 991px, atribuir o valor zero às propriedades *margin* e *height*. Este processo permite criar espaços que podem desaparecer da aplicação dependendo do tamanho de ecrã.

11.2.2 C# e SQL

No capítulo anterior criou-se três divisões para este painel. Na primeira divisão (barra de navegação) existem três botões. Cada um deles utiliza a propriedade *Visibility* para esconder e revelar os painéis entendidos. Além disso cada um dos botões precisa de retirar da base de dados a informação que irá ser demonstrada nos painéis. Para este efeito o botão *Hot Movies* irá utilizar a *stored procedure* MainPageFun (capítulo 11.1.2), o botão Perfil irá utilizar uma *stored procedure* que irá retirar a informação de utilizador e o botão Admin irá utilizar uma *stored procedure* que irá retirar a informação de administrador.

Na segunda divisão (zona com o título Procure um Filme) podemos encontrar um botão azul com o texto Procure. Esta aplicação tem dois tipos de procura: normal e avançada. Para aceder à procura avançada é necessário clicar no botão em forma de seta. Para fechar a procura avançada basta apenas clicar no botão em forma de seta novamente. A procura avançada tem esta forma:

Figura 31 Procura avançada ***Fonte****:autor*

Foram pensadas duas formas para a aplicação reconhecer se estamos em modo normal ou avançado. A primeira será verificar qual o estado do botão em forma de seta (cima ou baixo), a segunda será criar uma variável global (mantêm o valor através da função *View State*) que alterna entre 1 e 0 representando os dois modos. Foi selecionada a segunda opção porque esta não envolveria verificar a fonte da imagem o que requer mais processamento.

Ao clicar no botão Procure a aplicação irá procurar, na base de dados, filmes com as especificações dadas pelo utilizador. Em modo normal o utilizador apenas escreve o nome do filme que procura. Neste modo o botão irá utilizar duas *stored procedure*. A primeira serve para

contar o número de resultados e a segunda serve selecionar os primeiros seis resultados e demonstrá-los no painel. A primeira *stored procedure* tem a seguinte forma:

```
Create proc [dbo].[Maxcount] @var varchar
as
SELECT Count ([IDF]) as cont
FROM [MovieFlame].[dbo].[MainPage]
where   nome like '%'+@var+'%'
```

Depois de correr esta *stored procedure* no código é necessário igualar o seu resultado a uma variável. Isto é possível através do uso de um leitor (demonstrado no capítulo 11.1.2).

A segunda *stored procedure* é escrita da seguinte forma:

```
Create proc [dbo].[Searchbutton1] @var varchar
as
SELECT [nome], class
FROM [MovieFlame].[dbo].[MainPage]
where   nome like '%'+@var+'%'
```

Esta *stored procedure* devolve todos os resultados da procura. Depois de correr este procedimento no código, torna-se necessário guardar os resultados em duas listas. Isto é possível da seguinte forma:

```
SqlCommand C = new SqlCommand("Searchbutton1", conn)
{
CommandType = System.Data.CommandType.StoredProcedure,
Connection = conn
};
C.Parameters.AddWithValue("@var",SearchBar.Text);
List<string> numb = new List<string>();
List<string> numb1 = new List<string>();
SqlDataReader readorCounter = C.ExecuteReader();
while (readorCounter.Read()){
numb.Add(readorCounter[0].ToString());
numb1.Add(readorCounter[1].ToString());
}
```

A lista numb recebe todos os nomes de filmes (coluna 0) e a lista numb1 (coluna 1) recebe todas as classificações. De seguida é apenas necessário demonstrar os resultados ao utilizador.

Por fim a terceira zona (filmes) contem dois pares de botões e seis *Linkbuttons*. Cada par de botões contém um botão *next* e um *previus*. O botão *next* serve para mostrar os próximos seis filmes e o botão *previus* serve para voltar a mostrar os seis filmes já vistos na página anterior. O primeiro par de botões irá permitir ao utilizador navegar por todos os filmes na base de dados por ordem decrescente de classificação. O segundo par de botões servirá para navegar por todos os filmes que apareçam na sua procura. O segundo par de botões só é visível quando o utilizador clica no botão procure e o primeiro par de botões só é visível quando o utilizador está a navegar pelos filmes sem utilizar a barra de procura. Eles encontram-se na mesma posição o que causa a ilusão da existência de apenas um par de botões. O primeiro par de botões incrementa e reduz a variável Botao (declarada no capítulo 11.1.2) para selecionar que filmes serão visionados. Neste caso, o botão *next* incrementa a variável Botao por seis e o botão previus diminui o seu valor por seis. Além disto os dois botões são capazes de reconhecer quando chegam ao início ou ao fim da lista. Esta funcionalidade permite a não existência de valores vazios ou erros críticos no programa. Os botões são capazes disto comparando o valor máximo de resultados com o valor da variável Botao. O segundo par de botões utiliza exatamente a mesma lógica, mas faz uso de uma segunda variável global, Botao1, e o valor máximo de resultados será calculado com a pesquiza do utilizador em vez do número total de filmes. É necessário o uso de duas variáveis para evitar erros críticos e valores vazios. É também necessário igualar a variável Botao a zero quando entramos na secção de *Hot Movies* e é necessário igualar a variável Botao1 a zero quando clicamos no botão de Procure.

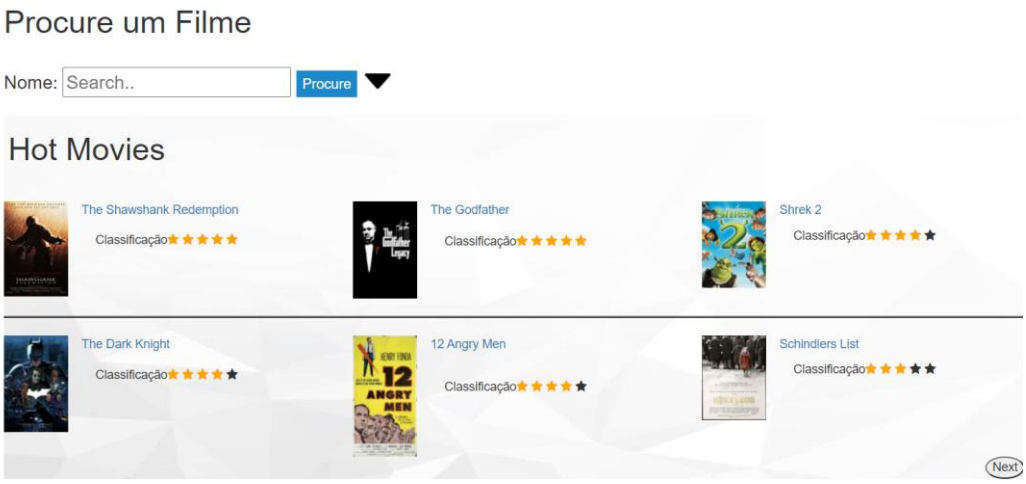

Figura 32 Inico de lista fotal de filmes ***Fonte****:autor*

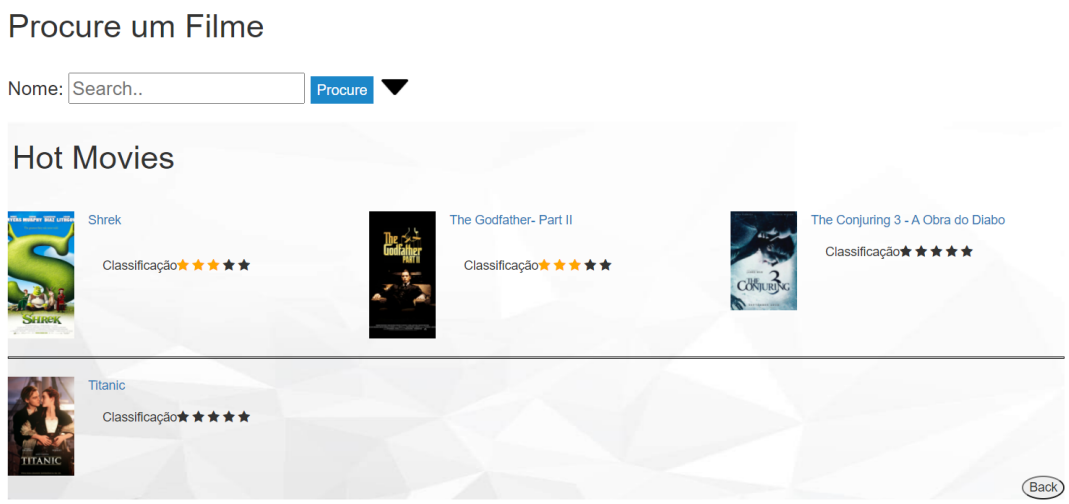

Figura 33 Fim da lista Total de Filmes ***Fonte****:autor*

Este primeiro par de imagens mostra o início e fim da lista total de filmes (primeiro par de botões).

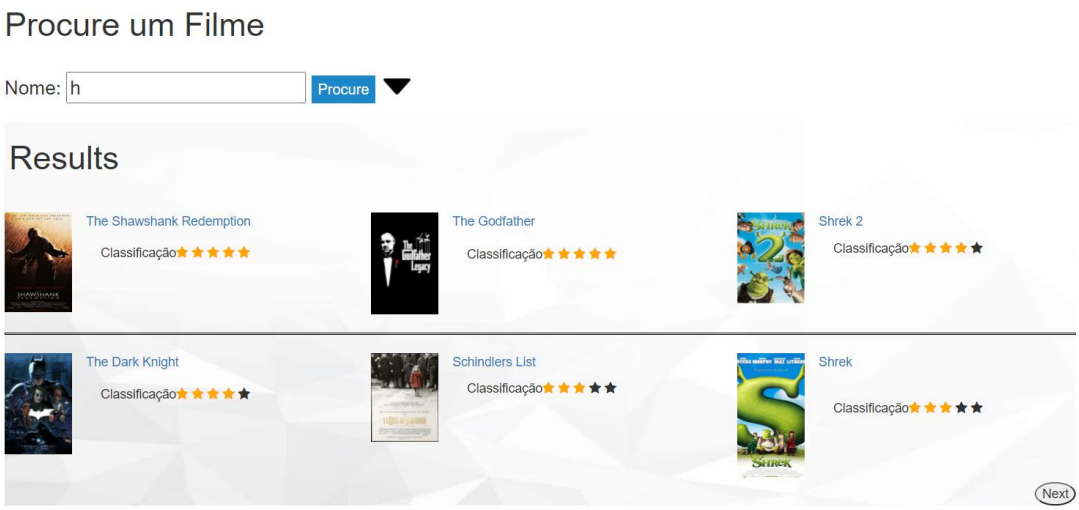

Figura 34 Inicio da lista procurada de filmes ***Fonte****:autor*

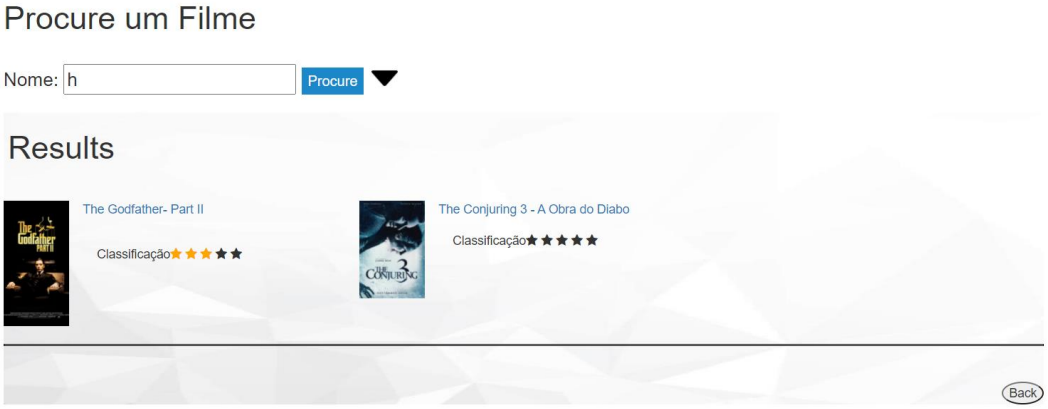

Figura 35 Fim da lista procurada de filmes ***Fonte****:autor*

Este segundo par de imagens mostra o início e fim da lista procurada de filmes (segundo par de botões). O utilizador consegue perceber, observando o título da zona de filmes, se se encontra na lista total de filmes ou na lista da sua procura. É necessário também assinalar que se o número de resultados ou filmes totais não forem múltiplos de seis então será necessário esconder certas células da *fluid grid*.

Por fim existem seis *Linkbuttons* na forma de nomes de filmes. Ao clicar num nome de um filme a aplicação irá esconder o painel de Filmes, revelar o painel de Filme específico, recolher informação na base de dados sobre o filme e irá recolher os dois primeiros comentários feitos por utilizadores sobre o filme. Toda esta informação será recolhida através do uso de *stored procedures*. Estas *stored procedures* necessitam de identificar o IDF do filme em questão, isto porque a informação encontra-se em várias tabelas diferentes. Para isto utiliza-se o nome do filme para chegar ao seu IDF. Este procedimento requer que todos os filmes dentro da base de dados tenham um nome diferente. Este processo tem dois passos:

1. Utilizar a seguinte *stored procedure* para recolher o IDF do filme em questão.

```
Create proc [dbo].[GetFid] @var varchar(8000)
as
select IDF from filme2
where nome = @var
```

2. Utilizar vários leitores e o valor de IDF recebido para recolher os dados necessários. Foram também criadas duas variáveis globais que guardam os IDComen dos comentários a serem demonstradas ao utilizador.

Ao finalizar este procedimento o utilizador terá acesso ao painel de Filme específico.

11.3 Painel de Filme específico

11.3.1 Html, CSS e Bootstrap

Este painel tem as seguintes formas:

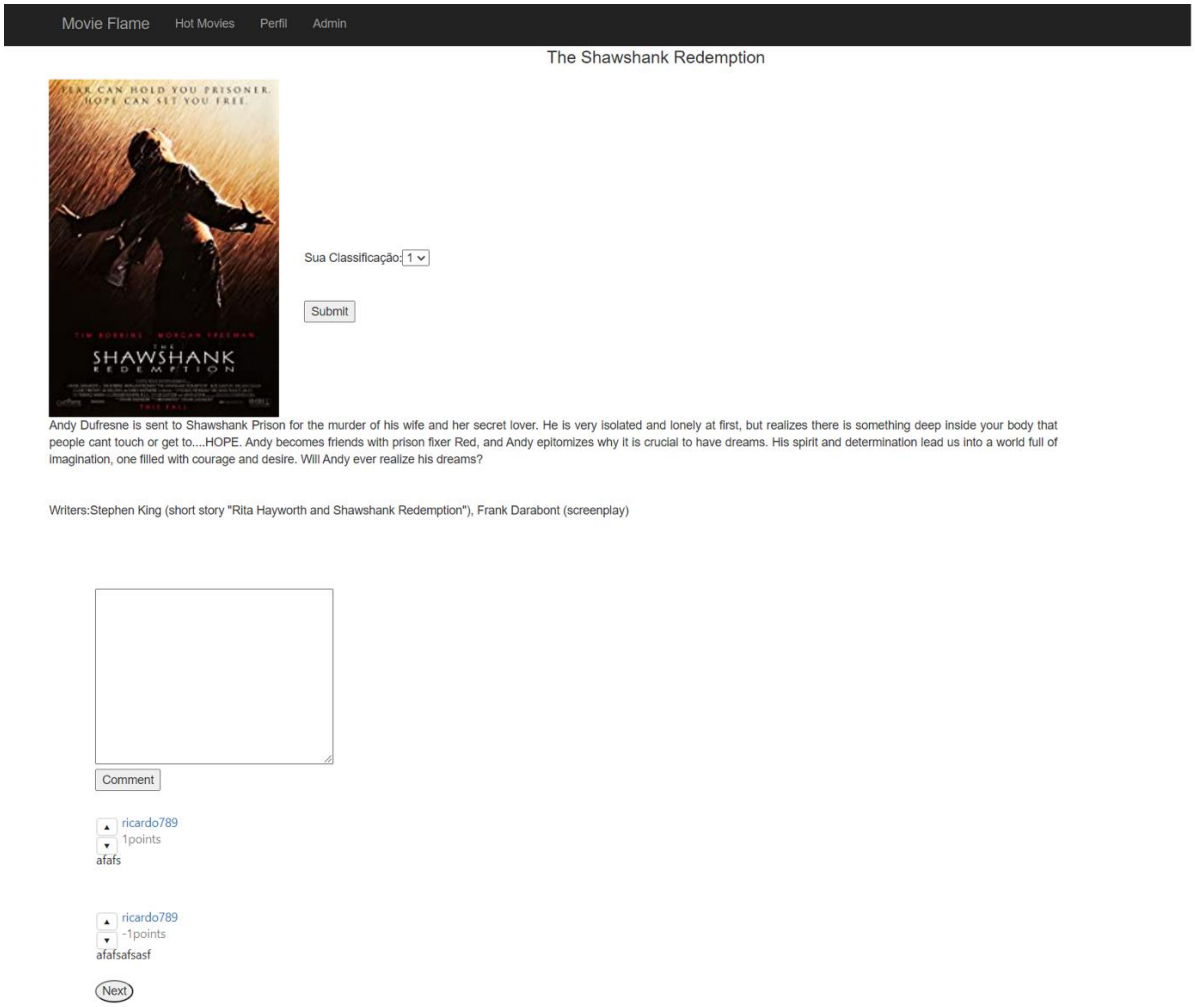

Figura 36 Painel de Filme específico com largura de ecrã superior a 1999px ***Fonte****:autor*

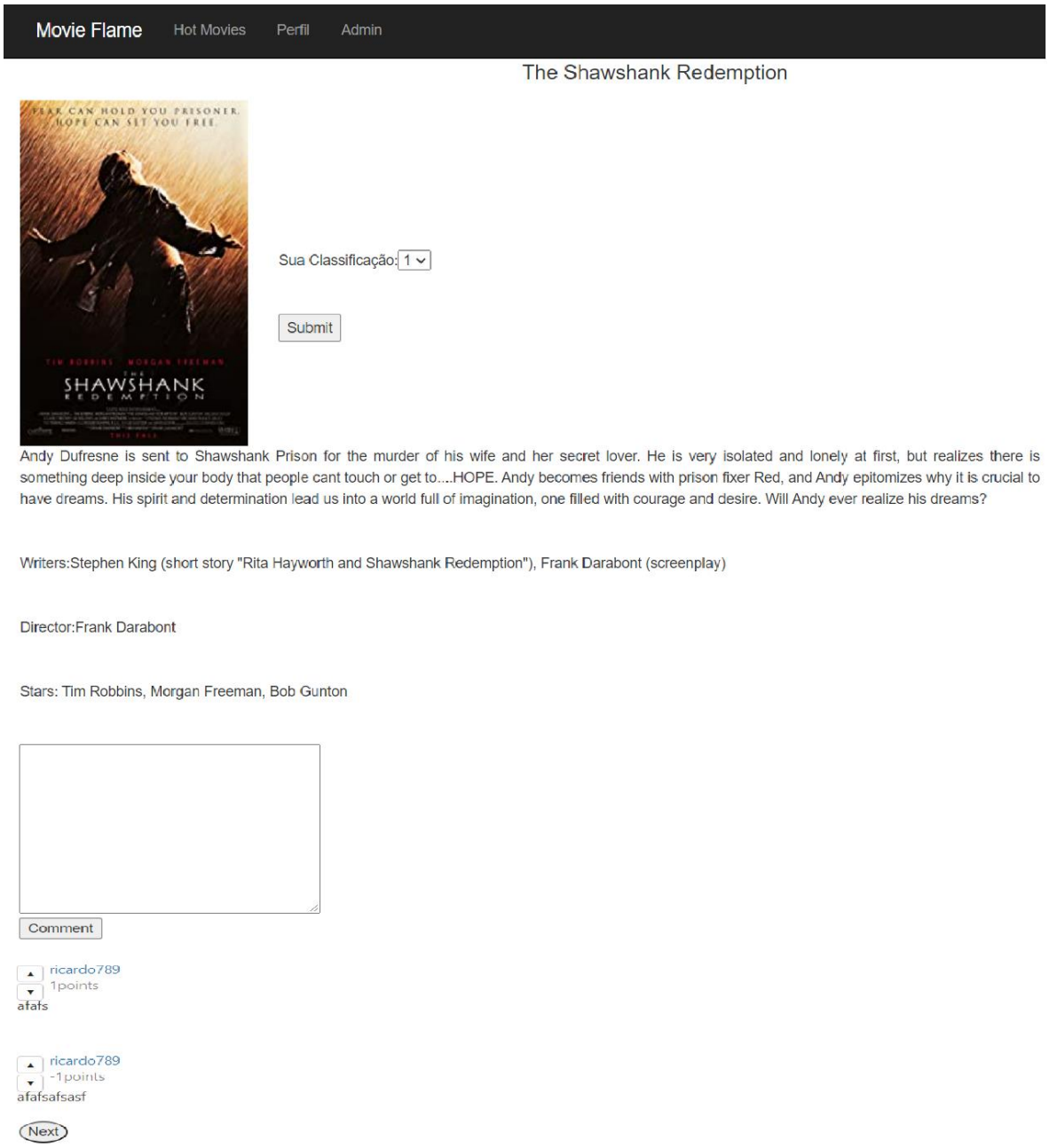

Figura 37 Painel de Filme específico com largura de ecrã entre 992px e 1199px ***Fonte****:autor*

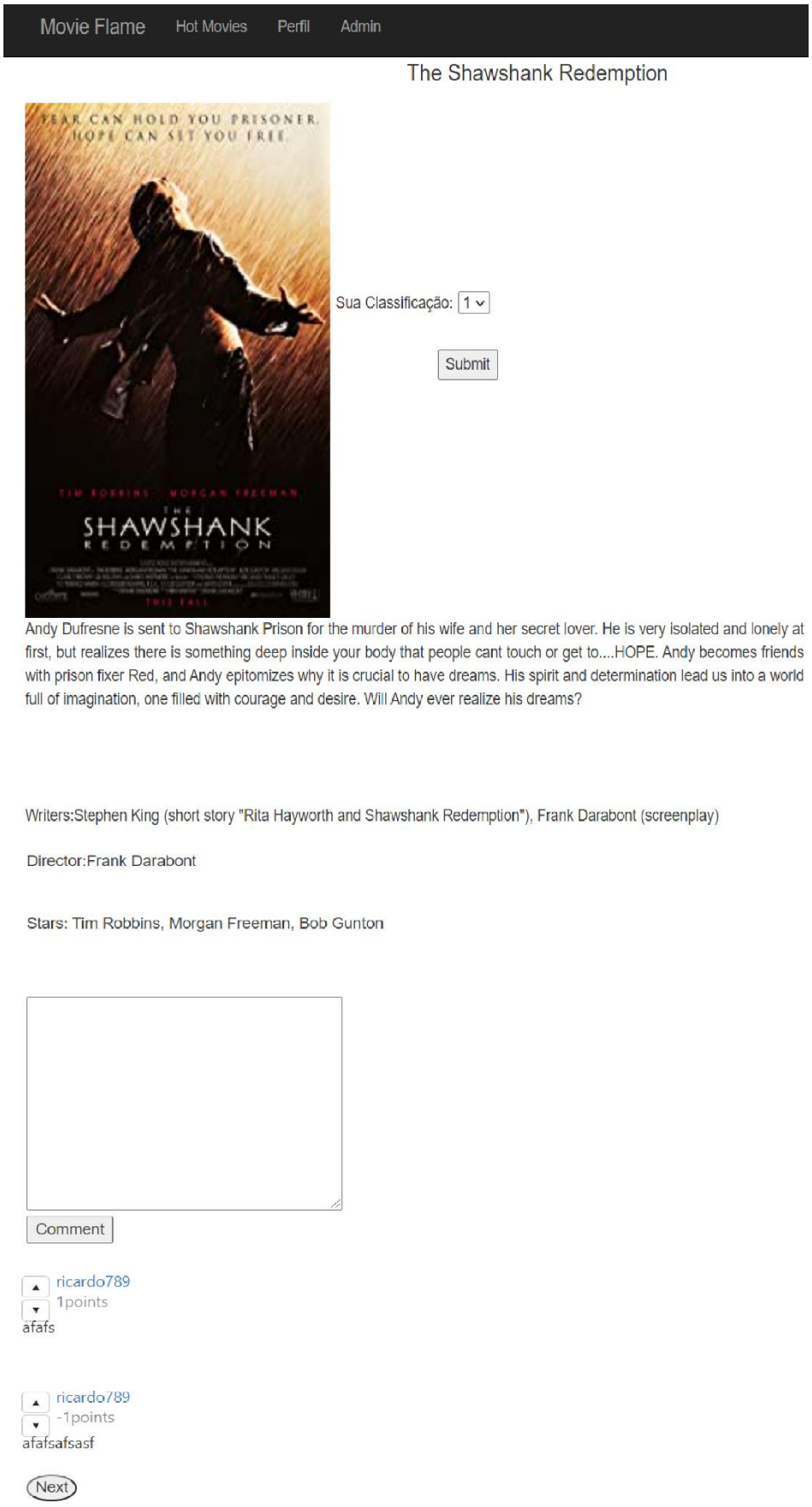

Figura 38 Painel de Filme específico com largura de ecrã entre 768px e 991px ***Fonte****:autor*

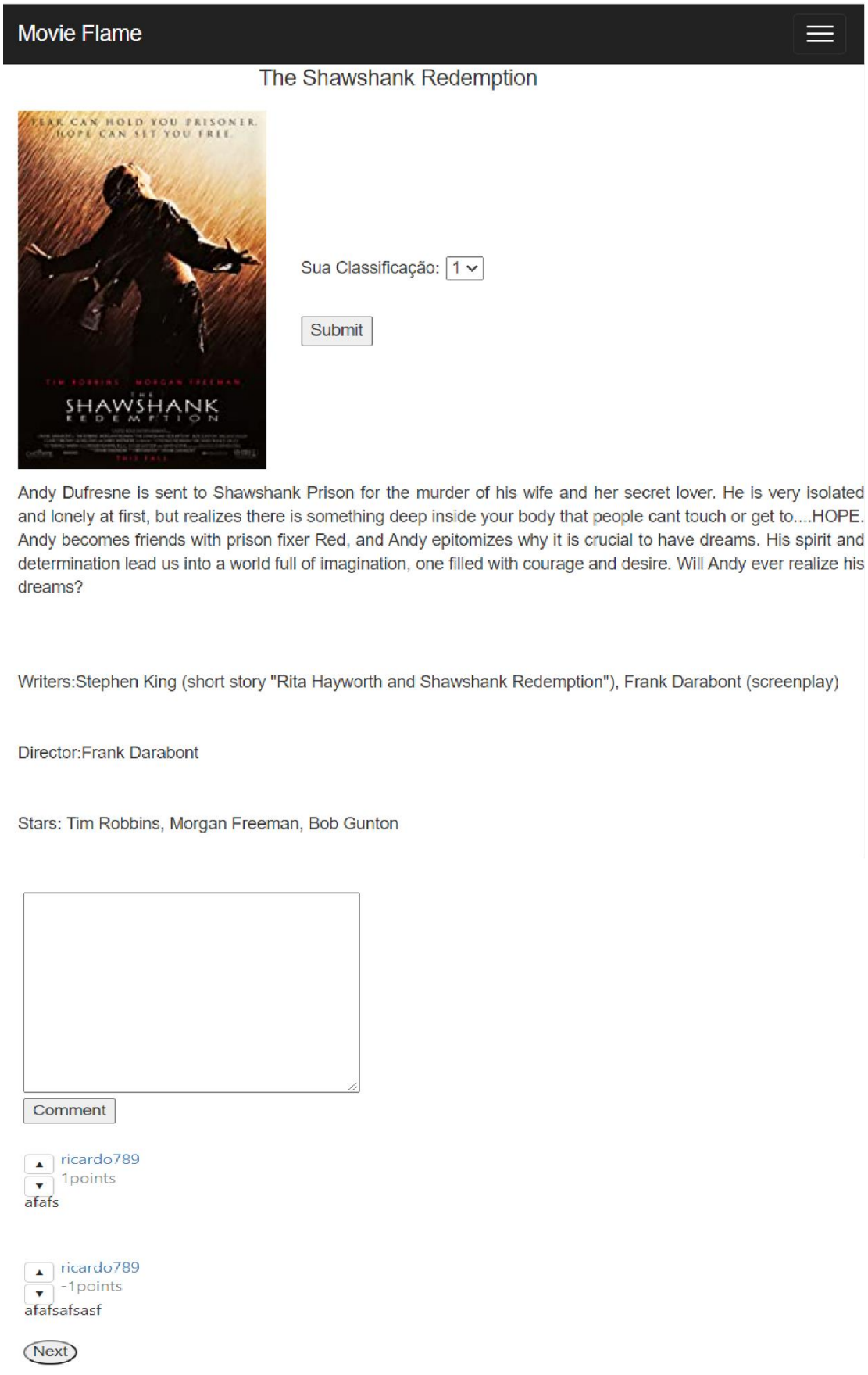

Figura 39 Painel de Filme específico com largura de ecrã inferior a 768px ***Fonte:****autor*

As quatro imagens em cima representam a totalidade do painel em questão. Um utilizador terá de utilizar a barra lateral ou arrastar o ecrã para poder interagir com a altura completa do painel. Em ordem de alcançar um *design* responsivo foram utilizadas variadas técnicas. O primeiro passo na criação deste painel foi a construção de uma *Fluid Grid*. Esta grelha terá três linhas. A primeira linha tem duas células e a segunda e terceira terão apenas uma célula. As duas células da primeira linha possuem tamanhos diferentes. Esta grelha terá a seguinte forma:

```
<div class="row">
<div class="col-md-3"> </div>
<div class="col-md-9"> </div>
</div>
<div class="row">
<div class="col-md-12" > </div>
</div>
<div class="row">
<div class="col-md-12" > </div>
 </div>
```

O segundo passo será criar os elementos visíveis pelo utilizador. Foi necessário aplicar em todos eles, técnicas de controlo de tamanho e margem em relação ao tamanho do ecrã. Além disto foi também necessário controlar a posição dos elementos do painel. Isto é possível através do uso das propriedades de CSS *top*, *bottom*, *left* e *right*. Estas propriedades servem para descrever qual a distância entre os elementos e os limites da aplicação. É também necessário declarar que a posição destes elementos será absoluta. Um exemplo do uso destas propriedades será:

```
#add {
position: absolute;
right: 68%;}
```

O exemplo acima fará com que o elemento com a classe *add* esteja posicionado 68% para a esquerda do limite onde se encontra. Por fim foram também utilizadas *tags* hr com existência e tamanho condicional em relação ao tamanho do ecrã.

No canto inferior esquerdo deste painel podemos encontrar uma zona de comentários com um *design* inspirado por Tahmid [50].

11.3.2 C# e SQL

O painel em questão possui cinco grupos de eventos possíveis. O primeiro será representado pelo botão *Submit*. Este botão irá permitir ao utilizador inserir a sua classificação pessoal do filme em questão. Tendo em conta que cada utilizador apenas tem direito a uma classificação por filme, será primeiro necessário identificar se a avaliação do utilizador deste filme é a primeira ou se é uma alteração de classificação. Para este efeito é necessário procurar na tabela Classificação2 se a combinação de IDF e idC já existe. Para este efeito utilizamos a *stored procedure* rateC:

```
Create proc [dbo].[rateC] @var int, @var2 int
as
SELECT [IDClass], [IDF], [idC], [class]
FROM [MovieFlame].[dbo].[classificacao2]
where   IDF=@var and idC = @var2
```

Para descobrir o idC utiliza-se uma *stored procedure* similar à getFid (capítulo 11.2.2) mas que utiliza a variável global *username* (capítulo 11.1.2). Se a *stored procedure* rateC devolver um valor vazio significa que esta classificação será a primeira. Neste caso será apenas necessário inserir o valor da classificação na tabela Classificação2 com o IDF e idC corretos. É possível inserir o valor através de um *insert* com a seguinte forma:

```
INSERT INTO classificacao2(IDF, idC, class)
VALUES (@var, @var1, @va
```

No caso de não ser a primeira classificação torna-se necessário mudar a classificação do utilizador. Isto é possível através de um comando *update* com a seguinte forma:

```
ALTER proc [dbo].[updateC] @var int, @var2 int, @var3 int
as
update [classificacao2]
Set
class= @var3
where   IDF=@var and idC = @var2
```

O segundo grupo tem apenas um elemento e este será ativado ao clicar no botão *Comment*. Este evento irá apenas inserir na tabela Comentários uma entrada com o IDF, idC e o texto escrito na caixa de texto em cima. Estes comentários são depois revelados na zona de comentários.

O terceiro grupo de eventos é ativado ao clicar no nome do autor de qualquer um dos comentários na zona de comentários. Este evento irá revelar o painel de Perfil próprio ou de outros. Este painel recebe e demonstra os primeiros dois comentários e todas as classificações do utilizador com o nome igual ao clicado. Toda esta informação é recolhida nas tabelas Classificações2 e Comentários. No caso das classificações, estas serão inseridas numa *GridView*. Este processo será explicado no capítulo 11.4.2.

O quarto grupo de eventos será ativado ao clicar em qualquer um dos botões com setas. Estes botões servem para atribuir um *like* ou *dislike* em comentários. Este evento utiliza a mesma lógica que o primeiro evento para descobrir se esta é a primeira interação do utilizador com o comentário em questão. Em caso de ser a primeira interação é apenas necessário inserir na tabela *likes* o valor 1 ou -1 com idC e IDComen corretos. Se não for a primeira interação será necessário utilizar o comando *update* para mudar o valor do *like*. O IDcomen é descoberto através uso de duas variáveis globais declaradas no capítulo 11.2.2.

O último grupo de eventos consiste nos botões *next* e *previus* na zona de comentários. Estes botões servem para navegar pelos vários comentários associados ao filme em questão. Ao selecionar os dois primeiros comentários utilizou-se a *stored procedure* Getcoment1:

```
ALTER proc [dbo].[Getcoment1] @var1 int, @var2 int
as
select a.Comentario, b.nome,a.IDComen
from (SELECT ROW_NUMBER() OVER (ORDER BY IDComen) AS Row, Comentario,
idC,IDComen
FROM Comentarios where idC=@var1 ) as a join Cliente1 as b on a.idC=b.idC
where Row =@var2
```

Como é possível observar este *select* cria uma coluna *Row Number* que irá criar um índice de número de linha. Assim se criarmos uma variável global e a incrementarmos e diminuirmos podemos recolher qualquer comentário desejado. Além disto se criarmos uma *stored procedure* que obtenha o número total de comentário, de um certo filme, é possível descobrir quando nos encontramos no início ou fim da lista de comentários. Esta lógica é similar à lógica utilizada nos botões *next* e *previus* do painel de Filmes.

11.4 Painel de Perfil próprio ou de outros

11.4.1 Html, CSS e Bootstrap

Este painel tem as seguintes formas:

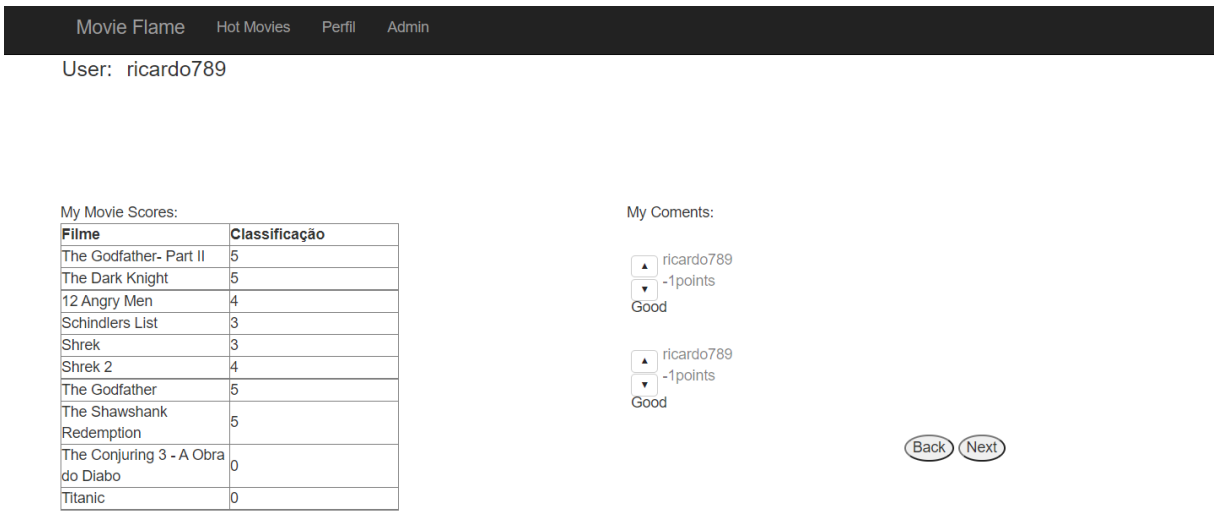

Figura 40 Painel de Perfil com largura de ecrã superior a 1999px ***Fonte****:autor*

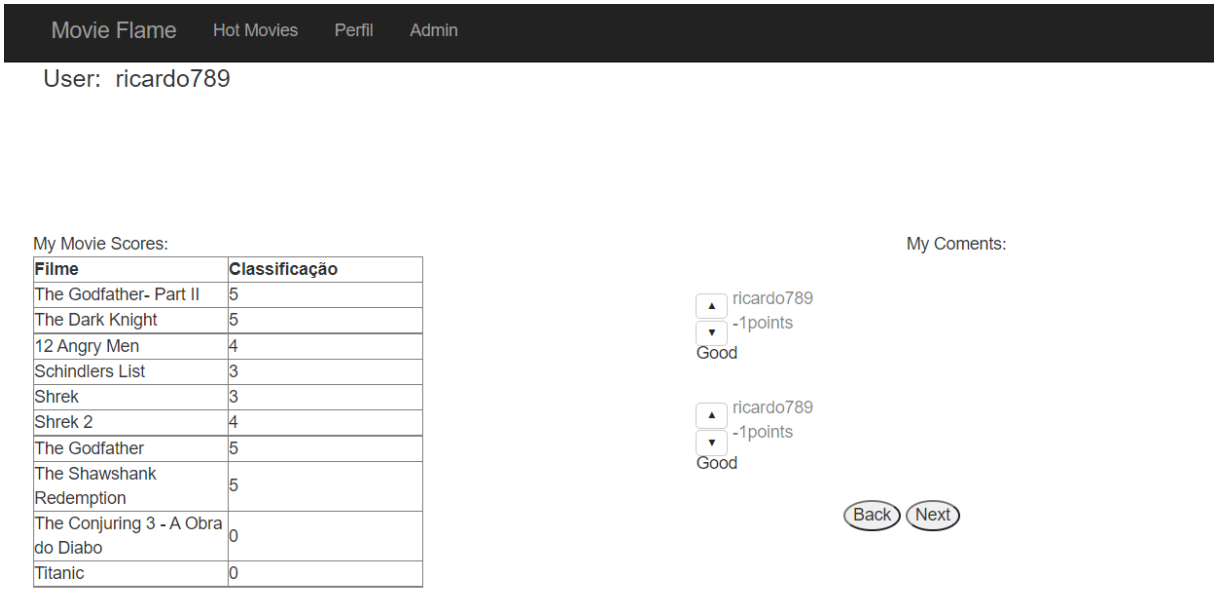

Figura 41 Painel de Perfil com largura de ecrã entre 992px e 1199px ***Fonte****:autor*

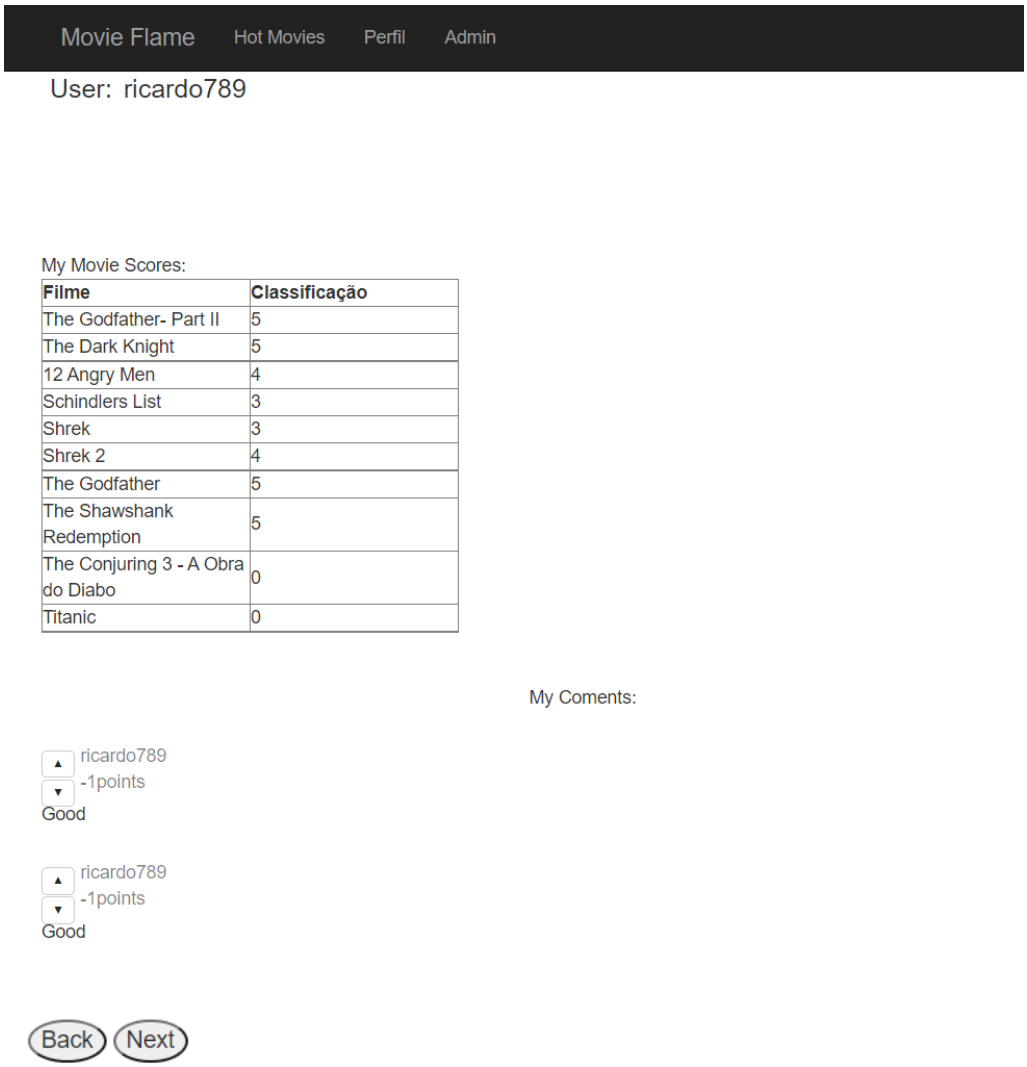

Figura 42 Painel de Perfil com largura de ecrã entre 768px e 991px ***Fonte****:autor*

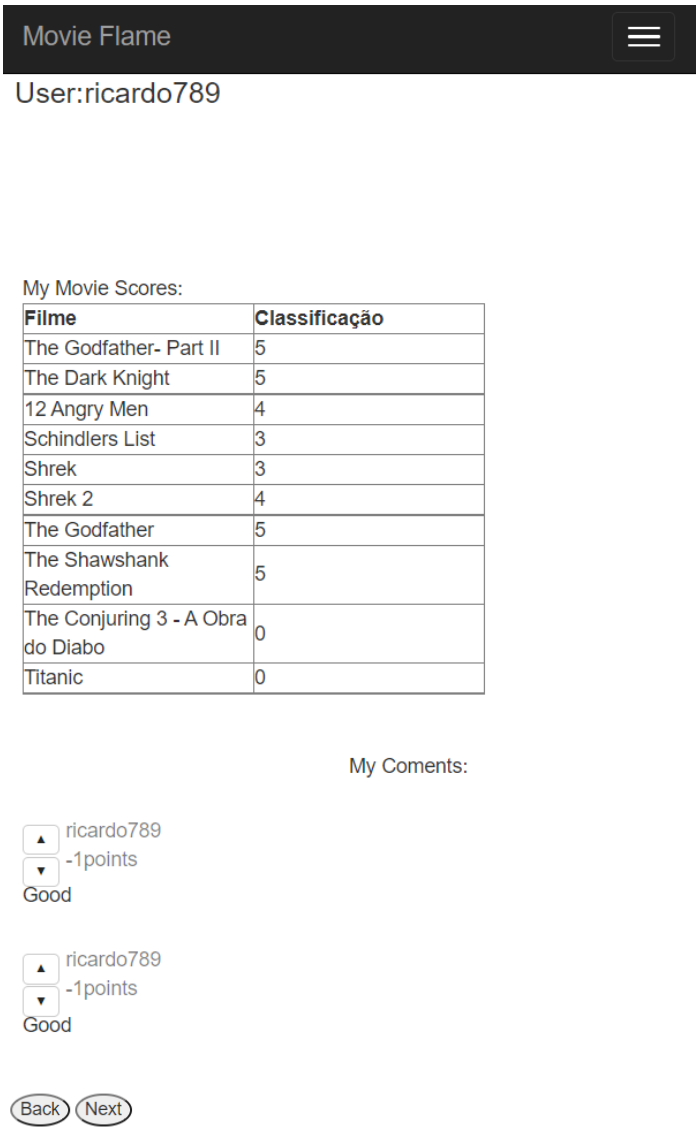

Figura 43 Painel de Perfil com largura de ecrã inferior a 768px ***Fonte****:autor*

Existem duas formas de aceder a este painel. A primeira será clicar num nome de um utilizador e a segunda será clicar no botão Perfil. Este painel serve para observar as classificações e comentários próprios (botão Perfil) ou de outros (nome de utilizador em zona de comentários). O primeiro passo ao criar este painel foi construir uma *Fluid Grid*. A grelha tem uma linha com duas células. A primeira tem um tamanho de oito e a segunda tem um tamanho de quatro. A segunda célula é uma zona de comentários igual à do capítulo anterior. A primeira célula contém uma *GridView*. Uma *GridView* é uma tabela onde é possível representar resultados retirados da base de dados. Esta *GridView* tem o seguinte código:

```
<asp:GridView id="tabela1" runat="server" AutoGenerateColumns="false">
<Columns>
<asp:BoundField DataField="nome" HeaderText="Filme" ItemStyle-Width="160px"/>
<asp:BoundField DataField="class" HeaderText="Classificação" ItemStyle-Width="160px"/>
```

```
</Columns>
</asp:GridView>
```

Como é possível observar na imagem acima, é primeiro necessário criar a *tag* de *GridView*. Utiliza-se a propriedade *AutoGenerateColumns* para referenciar que as colunas não são geradas automaticamente. De seguida declara-se as várias colunas. Cada coluna é representada por um *BoundFiled*. Foi também necessário escolher o valor mais apropriado para a propriedade *ItemStyle-Width*. O valor selecionado mantém o *design* desejado e o valor foi conseguido por processo de tentativa erro. Por fim foram utilizadas técnicas de *design* responsivo, referidas em capítulos anteriores, para manter o *design* responsivo.

11.4.2 C# e SQL

Este painel não tem qualquer evento único. Na zona de comentários é possível clicar em nomes de utilizadores, botões de *likes* e botões de *next* e *previus*. Todos estes tipos de botões foram explicados previamente. Ao entrar neste painel foi necessário recolher todas as classificações dadas pelo utilizador em questão e inseri-las dentro de uma *GridView*. O processo de inserir dados nesta *GridView* começa por descobrir o id do utilizador em questão. De seguida utiliza-se o seguinte código:

```
SqlDataAdapter gerirT = new SqlDataAdapter("select b.nome, class from classificacao2 a join filme2 b on a.IDF =b.IDF join Cliente1 c on c.idC = a.idC where c.idC= " + x, conn);
DataTable x123 = new DataTable();
gerirT.Fill(x123);
tabela1.DataSource = x123;
tabela1.DataBind();
```

No código acima x representa o id do utilizador selecionado. Tabela1 é o id da *GridView* e x123 é nome de uma estrutura *DataTable* que serve para ser possível dar valor à propriedade *DataSource* da *GridView*.

11.5 Painel de Administrador

11.5.1 Html, CSS e Bootstrap

Este painel tem as seguintes formas:

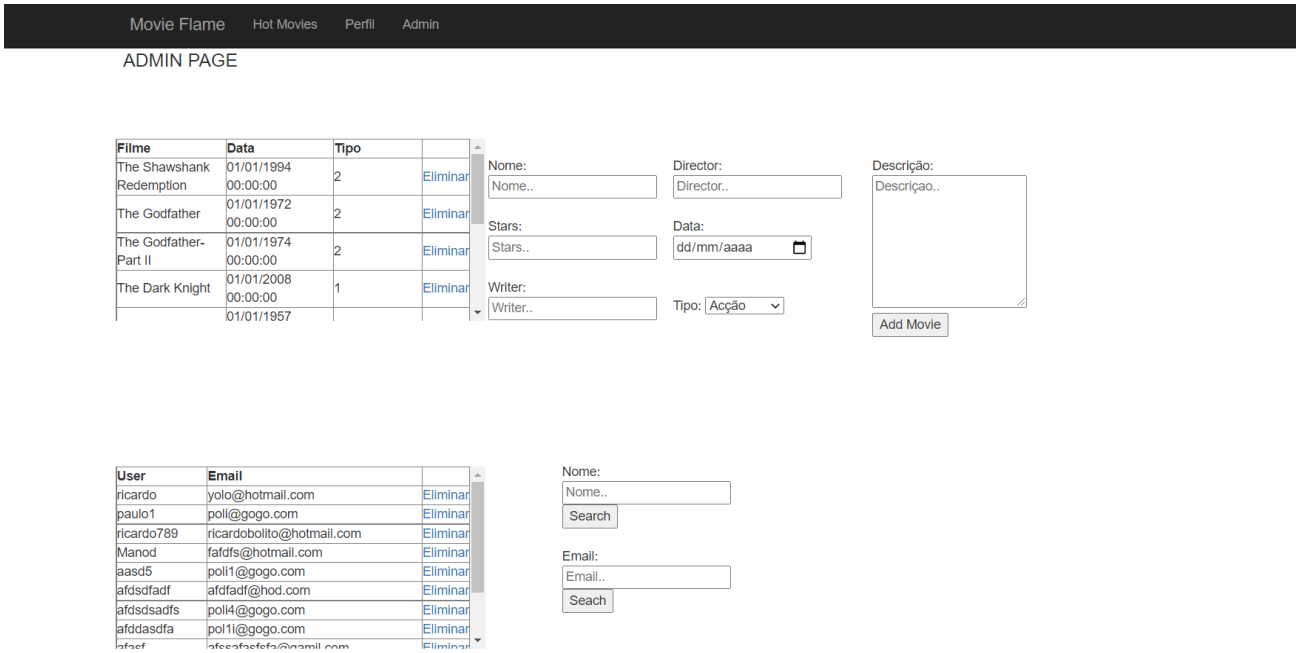

Figura 44 Painel de Admin com largura de ecrã superior a 1999px ***Fonte****:autor*

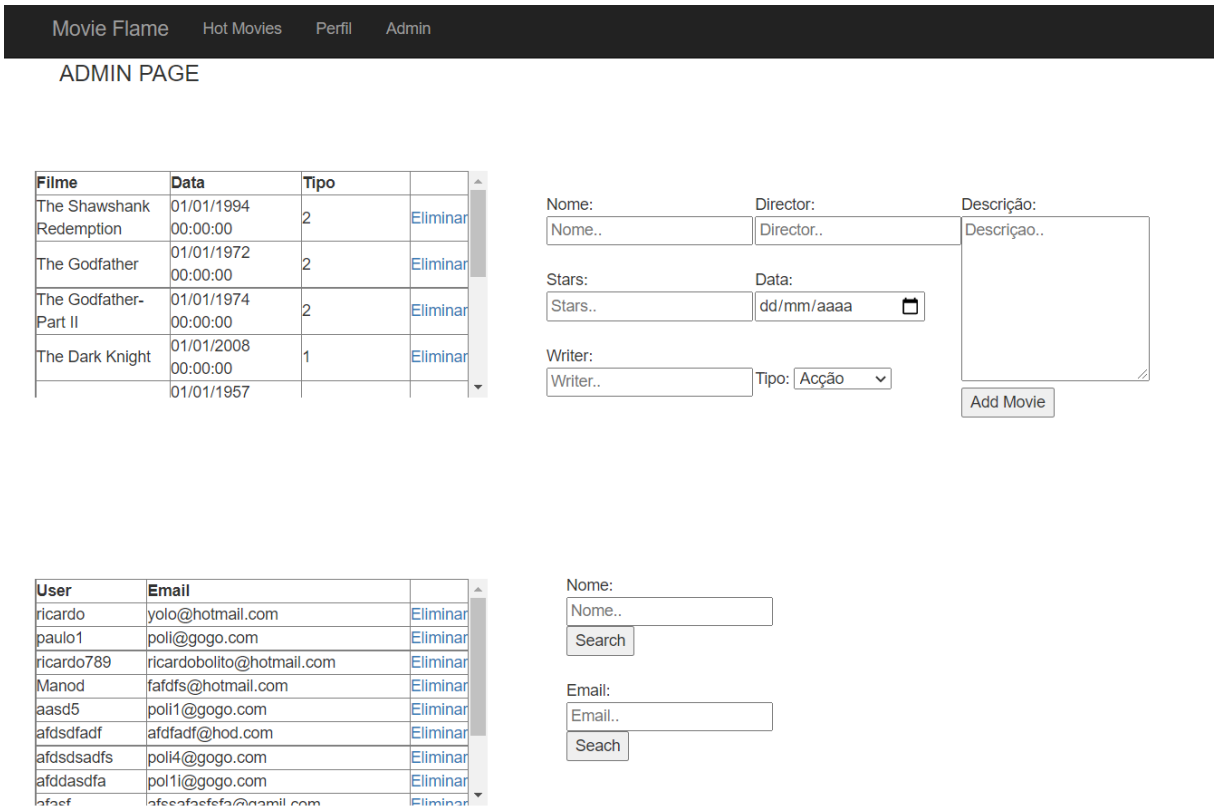

Figura 45 Painel de Admin com largura de ecrã entre 992px e 1199px ***Fonte****:autor*

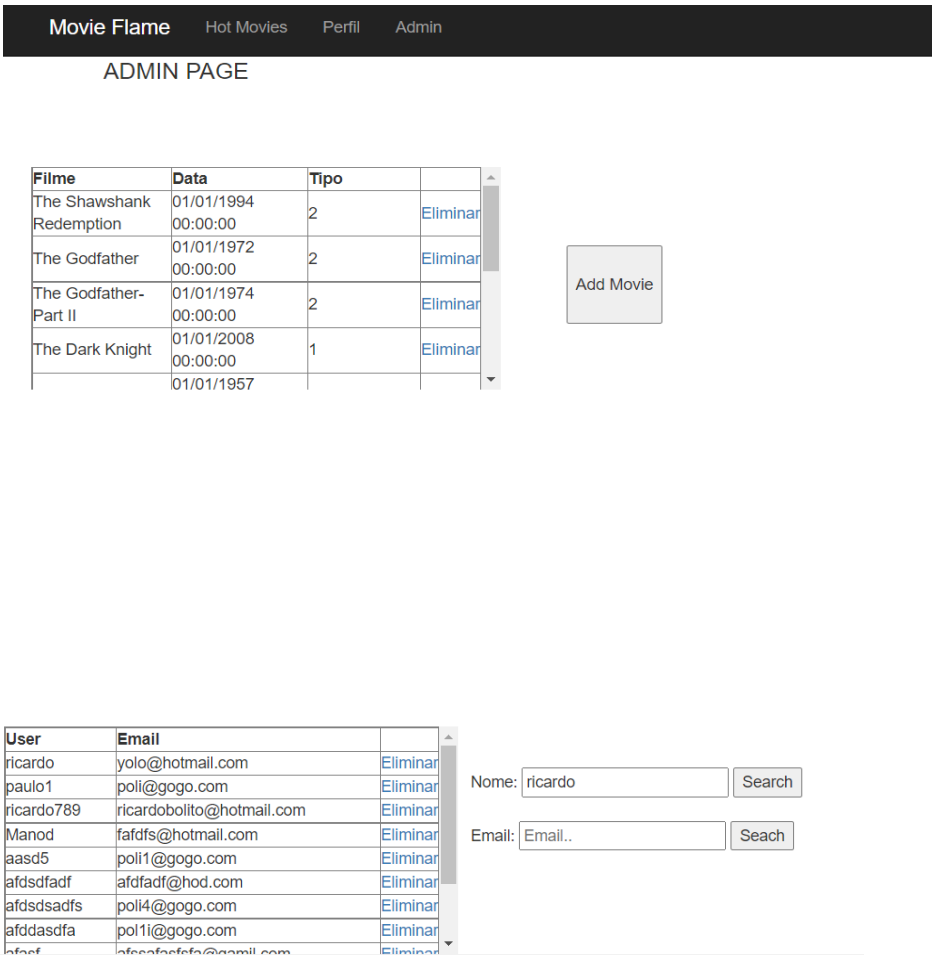

Figura 46 Painel de Admin com largura de ecrã entre 768px e 991px ***Fonte****:autor*

Este painel serve para gerir a aplicação na perspetiva de um administrador com permissões especiais. O primeiro passo na criação deste painel é a criação da sua *Fluid Grid*. Esta grelha tem duas linhas. A primeira linha possui quatro células. A primeira célula tem um tamanho de seis e contém a *GridView* com o nome de todos os filmes na aplicação. A segunda célula tem o tamanho de dois e contém os campos de *input* Nome, Escritor e Estrelas. A terceira célula tem um tamanho de dois e contém os campos de *input* Diretor, Data e Tipo. A quarta célula tem um tamanho de dois e contém o campo de *input* Descrição e o botão pequeno com o texto *Add Movie* (Figuras 44 e 45). A segunda linha contém duas células. A primeira célula tem um tamanho de oito e contém uma *GridView* com todos os utilizadores. A segunda célula contém os campos de input Nome e *Email*.

As *GridViews* utilizadas neste painel têm duas diferenças principais em relação à *GridView* utilizada no painel anterior: *scrollbar* vertical, e um *LinkButton* em cada linha. Para criar a *scrollbar* foi necessário envolver a *GridView* num painel e adicionar a propriedade *ScrollBar* com o valor Vertical. Para desenvolver os *Linkbuttons* foi necessário criar um *ItemTemplate* dentro das colunas da *GridView*. Este *ItemTemplate* tem uma função atribuída e um valor que

será recebido pela função desse botão. Utilizando a *GridView* de filmes como exemplo, esta existe graças ao seguinte código:

```
<asp:Panel ID="Panel4" Text="Label" ScrollBars="Vertical" Height="200" Width="400"
runat="server">
<asp:GridView id="clientlist" runat="server" AutoGenerateColumns="false">
<Columns>
<asp:BoundField DataField="nome" HeaderText="User" ItemStyle-Width="160px"/>
<asp:BoundField DataField="email" HeaderText="Email" ItemStyle-Width="160px"/>
<asp:TemplateField>
<ItemTemplate>
<asp:LinkButton ID="DelT" Text="Eliminar" runat="server"
CommandArgument='<%#Eval("idC")%>' Onclick="apagarC" ItemStyle-Width="30px"/>
</ItemTemplate>
</asp:TemplateField>
</Columns>
</asp:GridView>

</asp:Panel>
```

Neste caso, o *LinkButton* irá enviar para a função Eliminar o idC do filme em questão. Este idC irá ser utilizado para apagar o filme da aplicação.

A figura 46 demonstra uma versão do painel onde os campos de *input* para adicionar um filme são substituídos por um botão. Isto foi conseguido utilizando CSS para esconder os campos e mostrar o botão. Existem duas maneiras de chegar a este efeito em CSS. A primeira será posicionar o elemento a ser escondido fora do ecrã visível da aplicação. A segunda é utilizar a propriedade *visibility* e atribuir-lhe o valor *hidden*. É também necessário posicionar todos os elementos da melhor maneira (elementos deixam de ser clicáveis quando existem outros elementos em cima ou quando o elemento se encontra dentro de margens de outro elemento) e utilizar técnicas de *design* responsivo explicadas em capítulos em cima. Ao clicar no botão *Add Movie*, representado na figura 46 o painel irá esconder os seus elementos e revelar o seguinte formulário:

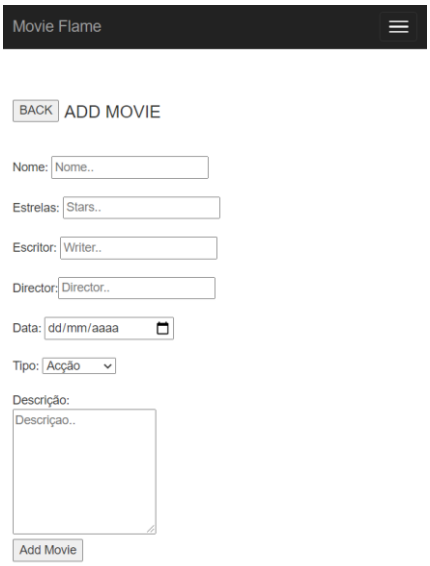

Figura 47 Formulario de Add Movie ***Fonte****:autor*

Este formulário recebe a informação do utilizador e adiciona na base de dados um novo filme.

11.5.2 C# e SQL

Este painel possui vários eventos representados pelos vários botões e *Linkbuttons* no painel. Os dois botões *search* utilizam os seus campos de *inputs* para fazer uma procura na base de dados por utilizadores com o *email* ou *username* inserido pelo administrador. Isto é possível através do seguinte código (no caso da procura por *email*):

```
string sql = "SELECT idC, [nome],[email] FROM[MovieFlame].[dbo].[Cliente1] where nome = @var";
SqlDataAdapter adapter = new SqlDataAdapter(sql, conn);
adapter.SelectCommand.Parameters.AddWithValue("@var", Semail.Text);
DataTable x124 = new DataTable();
adapter.Fill(x124);
clientlist.DataSource = x124;
clientlist.DataBind();
```

Na eventualidade do utilizador clicar no botão sem introduzir nenhum valor de *input* irá aparecer uma mensagem de erro.

Todos os *Linkbuttons* neste painel servem para apagar a linha e toda a informação relacionada com a linha onde se encontram. No caso de apagar um utilizador ou um filme será necessário também apagar toda a informação relacionada com estes. Para esta função de *delete* conseguir apagar toda esta informação é necessário primeiro fazer com que as chaves secundárias das

várias tabelas da base de dados permitam delete em *cascade*. Isto é possível seguindo os seguintes passos:

- Abrir o programa *Microsoft SQL Management Studio*.
- Ligar-se ao servidor.
- Abrir o *object explorer* e abrir a pasta de *Keys* dentro da pasta da tabela que se pretende alterar.

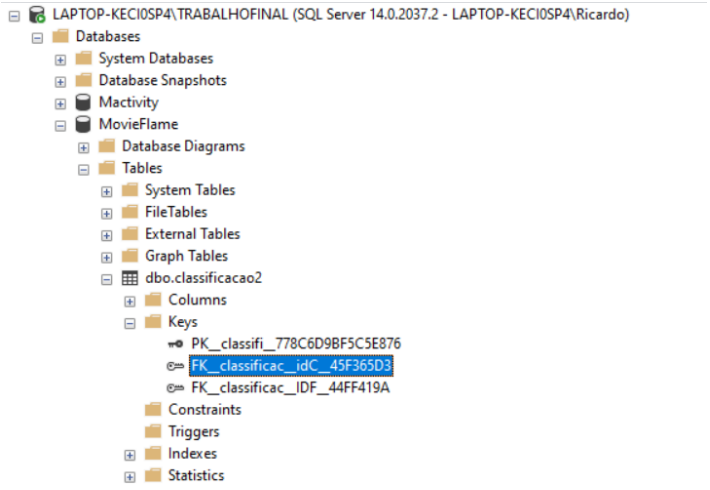

Figura 48 Pasta Key ***Fonte****:autor*

- Clicar na chave que se pretende alterar.
- Mudar a *delete rule* para *cascade*.

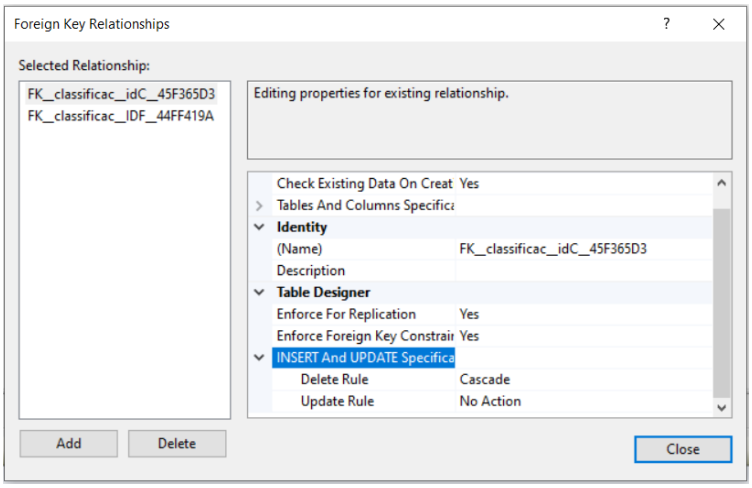

Figura 49 Cascade delete ***Fonte****:autor*

Depois deste processo, será necessário criar o código que permite apagar estes registos da base de dados. Para isto, a função terá de receber o valor de id que é enviado pelo *linkbutton*. O *linkbutton* possui um valor de id porque o *select* usado para dar valor à *GridView* seleciona o id, mesmo que não o mostre na tabela. A função recebe o valor e iguala-o a uma variável da seguinte forma:

```
LinkButton btn = (LinkButton)(sender);
string value = btn.CommandArgument;
int idC = 0;
bool res2 = int.TryParse(value, out idC);
```

A função *TryParse* e o *bool* servem para transformar o tipo de valor recolhido num número inteiro. De seguida é apenas necessário correr um método de *delete* com o id recolhido. No caso da *GridView* de utilizadores, o delete tem a seguinte forma:

```
string sql = "DELETE FROM Cliente1 WHERE idC =" + iduser;
SqlCommand cmd4 = new SqlCommand(sql, conn);
cmd4.ExecuteScalar();
```

Por fim é apenas necessário voltar a dar valor à *GridView* e o utilizador poderá observar que a linha em questão foi apagada.

O botão *Add Movie* representado nas figuras 44 e 45 serve para recolher toda a informação introduzida pelo administrador e adicionar um filme à base de dados. Para este efeito faz-se uso de um método *insert*. O botão *Add Movie* representado na figura 46 irá revelar o formulário da figura 48. Este formulário possui um botão *back* e um botão *Add Movie*. O botão *Add Movie* tem a mesma função que o botão *Add Movie* das figuras 44 e 45 e o botão *back* servirá para esconder o formulário.

12 Testes de Usabilidade

Usabilidade, neste contexto, refere-se à facilidade de utilizadores utilizarem *interfaces*. Tendo isto em mente, a aplicação web deverá seguir os seguintes conceitos [51] [52]:

- Aprendizagem- facilidade de utilizadores em realizar tarefas na primeira vez que utilizam uma aplicação;
- Eficiência- facilidade de utilizadores que já utilizaram a aplicação de interagir com a *interface* da aplicação e o quão rápido conseguem realizar as tarefas;
- Memória- facilidade de utilizadores voltarem a usar a aplicação depois de algum tempo de desuso;
- Satisfação- nível de satisfação de utilizadores ao utilizarem a aplicação;

Além disto, a aplicação também deverá seguir as dez heurísticas de usabilidade de Nielsen para o *design* da *interface* (capítulo 7.2.2) [52]. Os testes realizados nesta *interface* foram testes presenciais com dois intervenientes: um utilizador e um moderador.

12.1 Descrição de Teste

Este teste consiste na interação guiada do utilizador na *interface*, ou seja, o utilizador irá receber uma lista de tarefas e terá de as realizar. O moderador foi responsável por recolher a informação dada pelos utilizadores. Uma tarefa é considerada um sucesso se o utilizador conseguir realizar a tarefa através de um dos caminhos esperado pelo moderador (sem a utilização de *bugs* e *features* fora do conhecimento do moderador). Em caso de sucesso, o utilizador irá classificar de um (muito fácil) a cinco (muito difícil) o nível de dificuldade da tarefa. A lista de tarefas tem a seguinte forma:

- 1- Registo na aplicação.
- 2- *Log-in* na aplicação.
- 3- Selecionar três filmes dentro da lista de filmes existente na aplicação.
- 4- Utilizar a barra de procura para encontrar um dos filmes na aplicação.
- 5- Classificar três filmes.
- 6- Atribuir três *likes* ou *dislikes* num comentário.
- 7- Adicionar três comentários.
- 8- Aceder à sua página de perfil.
- 9- Aceder à página de perfil de outro utilizador através da zona de comentários.

12.2 Resultados

Foram inquiridas dez pessoas com idades entre os vinte e um e quarenta e seis anos. Com base nos dados recolhidos, podemos concluir que em média 85% das tarefas foram concluídas com êxito por novos utilizadores. Os dados recolhidos podem ser observados no gráfico seguinte:

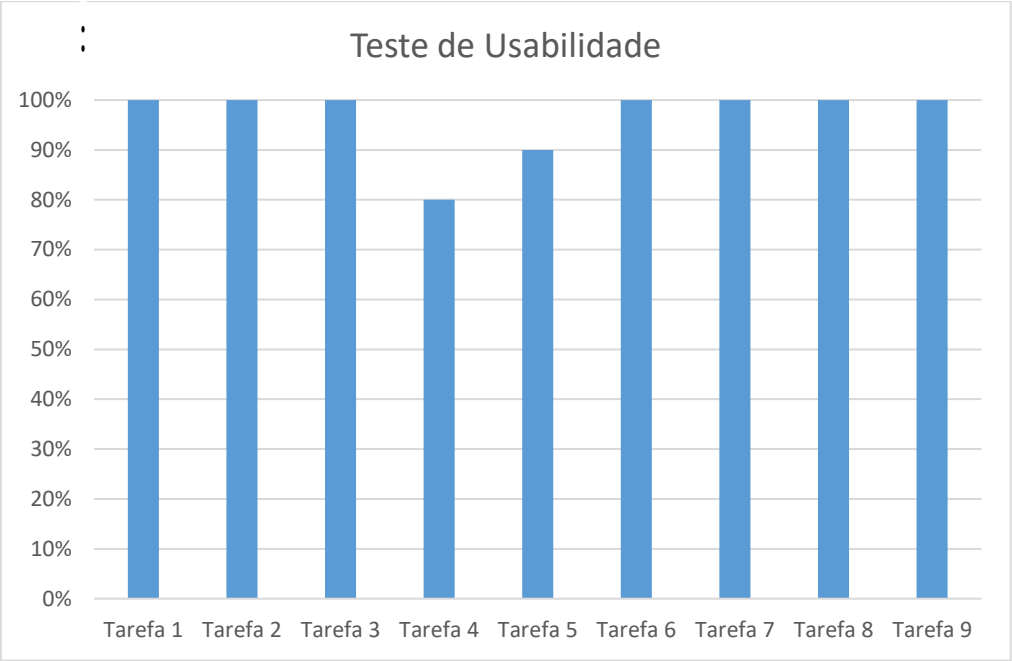

Figura 50 Teste de Usabilidade ***Fonte****:autor*

Segundo o gráfico, a tarefa mais difícil é a quatro. A principal dificuldade encontrada nesta tarefa foi a introdução de uma data para procura de filmes. Esta dificuldade surgiu devido a um erro de configuração no campo de datas que permitia a entrada de números demasiado grandes no campo de ano. A tarefa cinco não foi concluída por todos os utilizadores devido a uma falta de *feedback* por parte da aplicação. Quando um utilizador classifica um filme, deveria existir um indicador visual para notificar o utilizador (este indicador não existe). O resto das tarefas foi concluído com sucesso.

De seguida foi perguntado a cada utilizador que tarefas deveriam ser melhoradas. Os resultados podem ser observados neste gráfico:

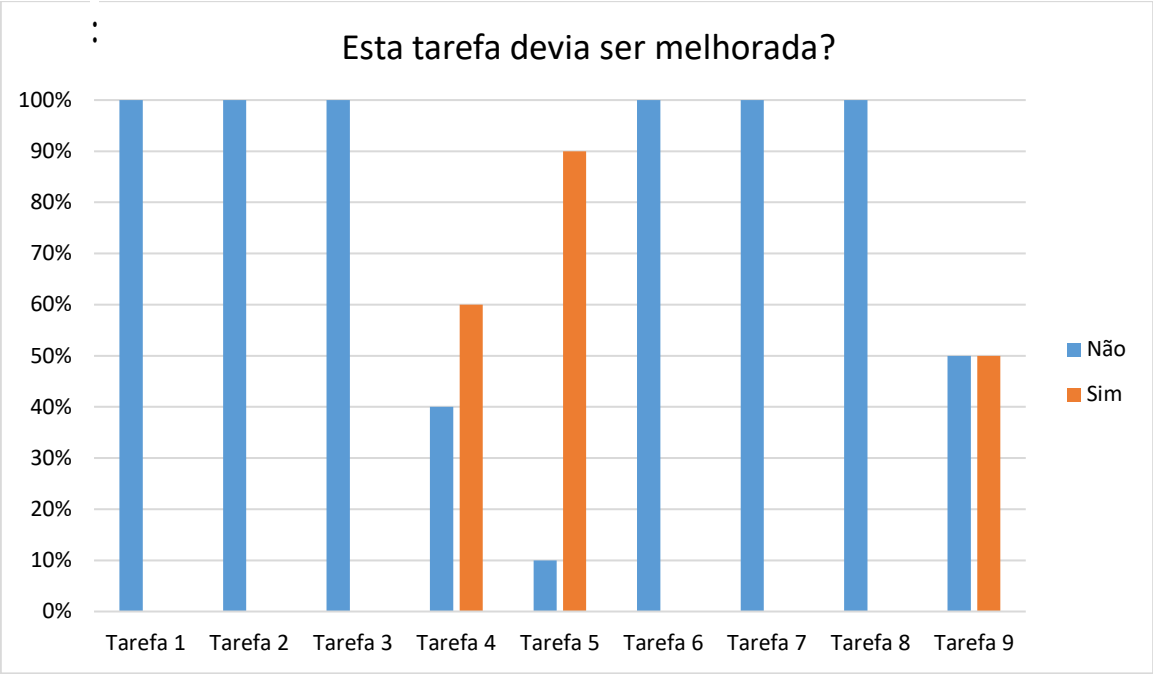

Figura 51 Teste de melhoramento de tarefas ***Fonte****:autor*

Observando o gráfico acima podemos concluir que utilizadores consideram que as tarefas quatro, cinco e nove devem ser melhoradas. Os utilizadores sugeriram as seguintes melhorias:

- Tarefa quatro- Correção do erro de formatação do campo de tempo e introdução de mais fatores na pesquisa avançada.
- Tarefa cinco- Utilização de um recurso visual em forma de estrelas cheias e vazias ao selecionar uma classificação e criar *feedback* para informar que a classificação foi guardada.
- Tarefa nove- Adicionar a capacidade de procurar utilizadores através do seu *username*.

Para além disto, os utilizadores classificaram a dificuldade de cada tarefa de um a cinco. O resultado pode ser observado neste gráfico:

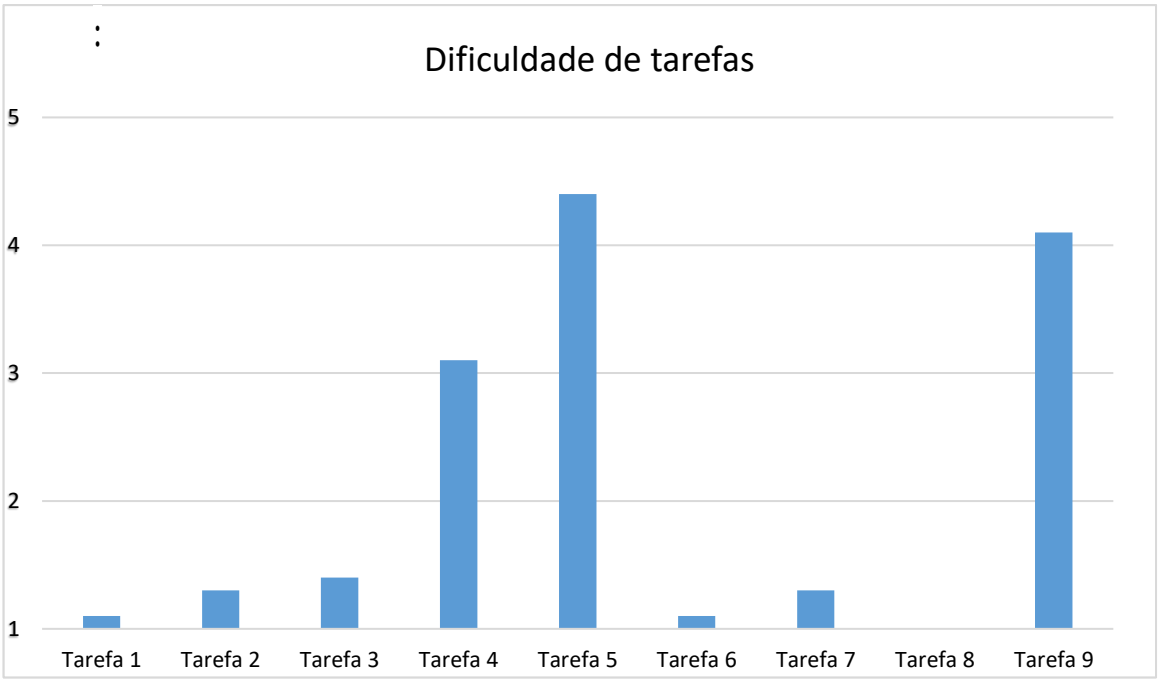

Figura 52 Dificuldade de tarefas ***Fonte****:autor*

Tarefas com uma classificação superior a dois foram consideradas problemáticas. Foi selecionado o número dois, uma vez que é o último número que se encontra mais afastado de cinco do que de um. Comparando a figura 44 com a figura 43, podemos concluir que as tarefas problemáticas são as que mais necessitam de ser melhoradas.

As soluções possíveis para os problemas apresentados em cima poderiam ser:

- Corrigir o erro de formatação no campo temporal.
- Adicionar campos adicionais à pesquiza avançada. Os utilizadores sugeriram adicionar filtros tendo em conta atores, diretores, produtores e classificação.
- Adicionar um campo visual em forma de estrelas cheias e vazias para representar a classificação atribuída pelo utilizador.
- Adicionar um aviso visual quando o utilizador atribui uma classificação a um filme.
- Introduzir a possibilidade de mudar o tipo de procura entre filmes e utilizador.

13 Conclusão

Após todo o processo de escolha é possível afirmar que o elevado número de tecnologias e ferramentas existentes atualmente interfere na decisão final de qualquer programador ou gestor. A seleção correta de tecnologias pode poupar uma grande quantidade de recursos a empresas e programadores.

De modo geral, pode-se concluir que todos os objetivos foram cumpridos. Em primeiro lugar foi feita uma análise sobre tipos de aplicações e decidiu-se construir uma aplicação web. De seguida, realizou-se uma análise de várias plataformas de desenvolvimento de aplicações e selecionou-se o Visual Studio. Foi investigado o mercado no qual o tipo de aplicação a ser desenvolvida estaria inserida. Foi também necessário realizar uma análise sobre tipos de linguagens de programação, tipos de *design* e que tipo de base de dados seria necessário criar. Em termos de linguagens foram utilizadas as seguintes: C#, HTML, CSS, Bootstrap e SQL. Quanto ao *design* decidiu-se utilizar *design* responsivo. Para a gestão de base de dados foi utilizado o *Microsoft SQL Management Studio*.

O último objetivo de criar e testar uma aplicação exemplo também foi concretizado, tendo assim sido atingido todos os objetivos propostos.

14 Trabalho futuro

Para a aplicação desenvolvida poder competir com outras aplicações do mesmo estilo, iria ser necessário efetuar algumas alterações, nomeadamente, seguir as sugestões dadas pelos utilizadores dos testes de usabilidade, um aperfeiçoamento do *design*, criação de mais *features* sociais como mensagens diretas ou fóruns de discussão, possibilidade de criar conta com *Facebook* ou *Twitter* e utilização de o email para recuperar e confirmar palavra-passes. Todos estes pontos teriam de ser investigados e testados antes da sua inclusão no projeto.

15 Bibliografia

[1] J. Musser, Web 2.0 Principles and Best Practices, O'Reilly Media, 2006.

[2] Broadband Search, “Mobile Vs Desktop Internet Usage(latest 2021 Data),” 2021. [Online]. Available: https://www.broadbandsearch.net/blog/mobile-desktop-internet-usage-statistics. [Acedido em 5 5 2021].

[3] R. Quivy e L. Campenhoudt, Manual de Investigacao em Ciencias Sociais, Gradiva, 1995.

[4] L. Kleinrock,, “An Early History of the Internet,” *IEEE Communications Magazine*, vol. 8, pp. 26-36, 2010.

[5] T. O'Reilly, “Web 2.0 Compact Definition: Trying Again - O'Reilly Radar,” 2021.

[6] K. Lee, M. K. Williams e K. Kim, “Learning Through Social Technologies: Facilitating Learning Experiences,” 2010.

[7] D. DiNucci, “Fragmented future,” *darcyd,* 1999.

[8] J. E. C. Graces, “Estudo do Comportamento do ícone no ambito do responsive design,” Fevereiro 2015.

[9] P. Zervas, A. Trichos, D. G. Sampson e N. Li, “A Responsive Design Approach for Supporting Mobile Access to Virtual and Remote Laboratories,” 2014.

[10] D. Nations, “What Is a Web Application?,” LifeWire, 25 Junho 2020. [Online]. Available: https://www.lifewire.com/what-is-a-web-application-3486637. [Acedido em 13 Novembro 2020].

[11] S. Lopez, “Aplicações mobile híbridas com Cordova e PhoneGap,” 2016.

[12] A. Charland e B. Leroux, “Mobile application development: web vs. native,” Communications of the ACM, 2011, pp. 49-53.

[13] D. Bonhaure, M. González, M. Núñez e L. Cernuzzi, “A model-driven approach for the development of native mobile applications focusing on the data layer,” 3 Dezembro 2019.

[14] R. Pires, “Top 20 plataformas de software para desenvolvimento de aplicativos,” Codificar, 7 10 2020. [Online]. Available: https://codificar.com.br/top-20-plataformas-de-software-para-desenvolvimento-de-aplicativos/. [Acedido em 12 6 2021].

[15] Appery, “Appery,” Appery, [Online]. Available: https://appery.io/. [Acedido em 04 04 2021].

[16] Edson, “Introdução ao Visual Studio Code,” DEV Media, 2016. [Online]. Available: https://www.devmedia.com.br/introducao-ao-visual-studio-code/34418. [Acedido em 21 6 2021].

[17] A. Troelsen, Pro C# 8 with. NET core 3: foundational principles and practices in programming, Apress, 2020.

[18] S. Robinson, O. Cornes, J. Glynn, B. Harvey, C. Mcqueen, J. Moemeka, C. Nagel, M. Skinner e K. Watson, Professional C#, WROX Press Ltd., 2001.

[19] A. Carriço e J. Carriço, Programação em Visual Basic.Net, CTI, 2002.

[20] A. Adam e P. Dragos-Paul , “Designing an MVC Model for Rapid Web Application Development,” *ScienceDirect,* 2013.

[21] A. Ulalah, “Model-View-Controller (MVC),” *Academia,* 2009.

[22] Interserver, “What is MVC? Advantages and Disadvantages of MVC,” Interserver, 28 Outobro 2016. [Online]. Available: https://www.interserver.net/tips/kb/mvc-advantages-disadvantages-mvc/. [Acedido em 4 4 2021].

[23] Y. Pacievitch, “HTML,” Info Escola, 2006. [Online]. Available: https://www.infoescola.com/informatica/html/. [Acedido em 21 6 2021].

[24] Raimundo, “As Novidades do HTML5,” DEV Media, 2012. [Online]. Available: https://www.devmedia.com.br/as-novidades-do-html5/23992. [Acedido em 21 6 2021].

[25] Y. Pacievitch, “Cascading Style Sheets (CSS),” Info Escola, 2006. [Online]. Available: https://www.infoescola.com/informatica/cascading-style-sheets-css/. [Acedido em 21 6 2021].

[26] T. Patton, “Put Bootstrap's responsive design features to good use,” TechRepublic, 21 Outubro 2013. [Online]. Available: https://www.techrepublic.com/blog/web-designer/put-bootstraps-responsive-design-features-to-good-use/. [Acedido em 20 Março 2021].

[27] S. S. Gaikwad e A. Pratibha, “A Review Paper on Bootstrap Framework,” *IRE Journals,* vol. 2, nº 10, 2019.

[28] D. Rabelo e C. M, “Análise Comparativa de Desempenho de Consultas entre um Banco de Dados Relacional e um Banco de Dados Não Relacional,” 2017.

[29] Brooks, “What is SQL,” *Buisiness News Daily,* 2014.

[30] L. Damas, SQL Structed Language, Fca, 2017.

[31] E. Stroparo, “História do SQL Server,” 2010. [Online]. Available: https://elderstroparo.blogspot.com/2010/01/historia-do-sql-server.html. [Acedido em 2021 6 21].

[32] G. Shultz, “Manage network logon credentials in Microsoft Windows,” Techrepublic, 7 Julho 2010. [Online]. Available: https://www.techrepublic.com/blog/windows-and-office/manage-network-logon-credentials-in-microsoft-

windows/#:~:text=Windows%20Credentials%20are%20user%20names,cards%20and%20other%20similar%20devices.. [Acedido em 02 05 2021].

[33] X. Zheng e J. Jin, “Research for the application and safety of MD5 algorithm in password authentication,” *IEEE,* 2012.

[34] H. Gascho Rempel e L. Bridges, “That Was Then, This Is Now:,” Dezembro 2013.

[35] L. Filippova e R. Svidelskyi, “Research of key approaches to responsive website development and their practical application,” 2016.

[36] A. Marisa e M. Tavares, “seleção de uma framework mvc clientside para uma aplicação web e mobile,” 2014.

[37] W. Jiang, M. Zhang, B. Zhou, Y. Jiang e Y. Zhang, “Responsive web design mode and application,” 29-30 Setembro 2014.

[38] S. Mohorovičić, “Implementing responsive web design for enhanced web presence,” 20-24 Maio 2013.

[39] T. C. Nogueira, “Evaluating Responsive Web Design’s Impact on Blind Users,” Junho 2017.

[40] E. . S. A. Majid, Z. Mansor e N. Kamaruddin, “Adaptation of Usability Principles in Responsive,” 10 Agosto 2015.

[41] Appgrove, “Best 10 Apps for Movie Reviews,” Appgrove, [Online]. Available: https://appgrooves.com/bundle/top-apps-to-help-manage-all-aspects-of-a-stage-production. [Acedido em 2021 5 20].

[42] Á. Fontenelle, “Metodologia científica: Como definir os tipos de pesquisa do seu TCC?,” 2017. [Online]. Available: https://andrefontenelle.com.br/tipos-de-pesquisa/#Metodologia_TCC_tipos_de_pesquisa_caracteristicas_e_modos_de_utilizacao. [Acedido em 7 5 2021].

[43] C. C, A. Sousa, A. Dias, F. Bessa, M. J. Ferreira e S. Vieira, “Investigação Acção Metodologias. Psicologia, Educaçao E Cultura,,” 2009.

[44] C. P. Coutinho, “Metodologias de Investigaçao em Ciências Sociais e Humanas,” 2011.

[45] D. P. Lacerda1, A. Dresch, A. Proença e J. A. A. V. Antunes Júnior, “Design Science Research: método de pesquisa,” *Gest. Prod.,* vol. 20, nº 4, pp. 741-761, 2013.

[46] J. v. Brocke e A. Maedche, “The DSR grid: six core dimensions for effectively planning,” *Electronic Markets,* pp. 379-385, 2019.

[47] M. Guertler, “Responsive Design: The Key to Responsive Mobile BI Applications,” *Buisiness Intelligence Journal,* 1 Março 2014.

[48] Adama, “Connect SQL Server with Visual Studio 2019,” Ozanecare, Abril 2019. [Online]. Available: https://ozanecare.com/connect-sql-server-with-visual-studio-2019/. [Acedido em 4 6 2021].

[49] F. Awesome, “fa-star,” Font Awsome, [Online]. Available: https://fontawesome.com/v4.7/icon/star. [Acedido em 01 01 2021].

[50] Tahmid, “Styling Comment Threads,” CSS-Tricks, 7 Dezembro 2020. [Online]. Available: https://css-tricks.com/styling-comment-threads/. [Acedido em 2020 12 12].

[51] L. M. R. Oliveira, “Desenvolvimento de Aplicação Web de Pesquisa, Gestão e partilha de eventos,” 2015.

[52] [J. Nielsen, “Usability 101: Introduction to Usability,” Nielsen Norman Group logoNielsen Norman Group, 3 1 2012. [Online]. Available: https://www.nngroup.com/articles/usability-101-introduction-to-usability/. [Acedido em 7 6 2021].