FACULDADE E ESCOLA TÉCNICA ALCIDES MAYA

CURSO TÉCNICO EM INFORMÁTICA

LEONARDO MEDEIROS LEIMAN

Cinema App – Uma aplicação para o gerenciamento de filmes e ingressos de cinema.

Projeto de Pesquisa apresentado como requisito parcial para obtenção do título de Técnico em Informática, pelo Curso de Técnico de Informática.

Orientador: Prof. Me. João Padilha Moreira[1]

Porto Alegre

2024

1 Professor Curso Técnico de Informática – Faculdade e Escola Técnica Alcides Maya - joao_moreira@alcidesmaya.edu.br

Cinema App – Uma aplicação para o gerenciamento de filmes e ingressos de cinema.

RESUMO

Este projeto apresenta um sistema que busca facilitar o contato entre clientes e uma empresa que oferta serviços de cinema permitindo a compra de ingressos pela rede de Internet com a finalidade de reduzir filas e incômodos relacionados de forma fácil e intuitiva dentro de um ambiente configurável e controlado pelos funcionários da empresa.

O sistema foi construído com a livraria .NET e sua expansão ASP, o que permite uma fácil integração com bancos de dados já existentes no mercado e força o código a sempre estar receptivo para novas atualizações.

Em seguida será explicado em detalhes sobre como essas tecnologias foram aplicadas e como executar o aplicativo em um ambiente de servidor.

O projeto não foi avaliado e não foi executado em um servidor de “produção” e portanto não possui avaliações de usuários.

Palavras chaves: cinema, ASP.NET, .NET, C#, Web, Blazor, tecnologia, sistema, educação, WEB

LISTA DE FIGURAS

LISTA DE SIGLAS

ABNT	Associação Brasileira de Normas Técnicas
NBR	Normas Brasileiras de Regulação
SWBES	Semantic Web-Based Educational Systems
TIC	Tecnologias da Informação e Comunicação

SUMÁRIO

1 INTRODUÇÃO

A expansão dos serviços de internet no mercado mundial mostrou para todos que para uma empresa se manter relevante ela precisa centralizar seus dados conforme ela cresce no mercado, não fazer impedirá o crescimento futuro da empresa por não possuir dados em comum de seus clientes para descobrir para onde expandir, além de perder oportunidades de atendimento para competidores que fornecem um serviço pela internet.

Um serviço web permite os clientes fazerem pedidos independente de um atendente humano, cortando a barreira de tempo que todo atendimento presencial possui além do requisito de estar presente para fazer a demanda e permitir padronizar a experiência de cada usuário a um preço extremamente baixo.

Para oferecer um serviço web é necessário registrar um domínio web e possuir maquinário para processar os pedidos dos clientes, normalmente estes são custos que apenas grandes empresas conseguiriam sustentar, porém com serviços de processamento em nuvem como AWS,Firebase,Azure entre outros é possível ter um serviço acessado globalmente pela fração do preço de um maquinário local.

A separação do processamento em um servidor dedicado também permite a criação de aplicativos que se interconectam e permitem ofertar o serviço para diferentes situações, como telões na frente do cinema ou a página web na internet.

1.1 – Delimitações do Tema

O cinemaProject é um aplicativo web desenvolvido para permitir a interação de clientes pela internet com uma empresa ofertora de serviços relacionados a sessões de cinemas. O aplicativo em si não se responsabiliza pela organização da empresa em si e apenas se conecta e expande um sistema já existente ao mesmo tempo que permite ter dados locais se mantendo como uma parte independente.

O aplicativo apenas se responsabiliza em mostrar sessões compráveis para um usuário que serão disponibilizadas pela interface gráfica administrativa por

padrão ou por um terceiro aplicativo que se conecta diretamente com o banco de dados. As compras no aplicativo não são reais e apenas são marcadas como o relacionamento entre clientes e ingressos e por padrão o cliente apenas será redirecionado para a tela inicial com uma mensagem de sucesso.

1.1 Objetivos

Mostrar as capacidades de aplicações web para prestação de serviços pela internet a fim de criar atendimentos flexíveis para os usuários.

1.1.1 Objetivo Geral

O objetivo deste trabalho é mostrar a capacidade da linguagem C# com Blazor ASP.NET para a criação de páginas web ao mesmo tempo que se é verbalizado as maneiras como essas tecnologias facilitam o contato entre empresas e clientes com ênfase a um sistema relacionado a cinemas.

1.3.2 Objetivos Específicos

- Identificar o uso de sistemas web para venda de ingressos;
- Verificar nosso sistema web e como ele funcionará;
- Estudar a união da esfera de comércio e serviços com tecnologia.
- Descrever a criação de um sistema para a venda de ingressos.

1.2 Justificativa

A aplicação também permite a centralização dos múltiplos processos administrativos como uma base de dados unificada, expansão para aplicativos de telão e a possibilidade de criar salas VIP com descontos pessoais para cada usuário conforme utilizam o sistema.

A criação de aplicativos web tornam empresas competitivas a um nível local ou potencialmente global permite o contato entre a empresa e um usuário de forma flexível independente de horário por preços que são possivelmente ignoráveis considerando os benefícios da tecnologia.

1.3 Tecnologia e educação

A tecnologia relacionada a computadores avançou de forma exponencial conforme os anos passaram, atigindo todas as empresas da área com um exemplo famoso sendo a evolução dos produtos da Apple: em 1990 o Apple 1, um dos primeiros computadores de mesa pessoal lançados, possuía cerca de 4 kilobytes de memória RAM e cerca de 250 Bytes de memória, em 20 anos a Apple lançou o seu primeiro MacBook em 2009 onde além de ser portátil a ponto de caber confortavelmente dentro de uma mochila também possuía 8 gigabytes de memória RAM e cerca de 256Gb de armazenamento em disco.

O avanço exponencial dos computadores abriu espaço para o processamento em massa de dados de usuários ou da lógica de negócio empresarial ao mesmo tempo que a internet permitia facilmente se conectar a qualquer servidor aberto á ela. A demanda por servidores mais próximos dos clientes para reduzir a latência da conexão e permitir processar mais dados a cada segundo resultou na explosão da indústria de hospedagem em nuvem com a Amazon Web Services assumindo o volante por ser uma das primeiras com um serviço modular e por ser parte da já estabelecida no mercado Amazom.

As páginas web se moldaram aos serviços em nuvem, permitindo que as páginas se separassem não apenas em blogs ou repositórios de informações mas também como uma plataforma de serviço graças ao fácil acesso à hospedagem e processamento de dados dos serviços de nuvem.

1.4 Usabilidade de sistemas Web para satisfazer demandas do mercado

A usabilidade de um sistema web refere-se aqui à qualidade da interação entre o usuário com o sistema hospedado na internet muitas vezes dependendo de muitos aspectos como a latência da conexão, facilidade de entendimento do layout da página, eficacia para transmissão da mensagem principal e eficiência do produto web.

Um serviço web normalmente para ser considerado satisfatório no mercado precisa satisfazer uma demanda popular. O Youtube satisfaz a necessidade de uma plataforma para a transmissão de informações em formato de vídeo ao mesmo tempo que a Wikipédia satisfaz a necessidade de termos documentos atualizados sobre o conhecimento humano e de quais documentos as informações foram retiradas com ambas podendo também trabalhar em conjunto.

A internet possibilita maneiras criativas para se conquistar cada nicho do mercado, mesmo se o CinemaApp tornar-se um aplicativo popular ainda assim poderia satisfazer uma porção maior de forma mais barata hospedando uma cópia digital no banco de dados e transmitindo para os usuários assistirem em seus computadores pessoais, a maneira como atender estas demandas é o que verdadeiramente torna e mantêm um website popular.

1.5 Tecnologias usadas na construção do sistema

De forma geral, o sistema é inteiramente construído em C# com o framework .NET com sua expansão ASP. A linguagem web Javascript sendo substituída em grande parte pela camada de abstração Blazor que permite escrever código C# que é compilado para WebAssembly e executado pelo navegador do

usuário. Para o backend por padrão é utilizado o banco de dados SQLite para armazenar os dados localmente.

Para a estilação da página será utilizada a livraria Bootstrap junto com Popper e Jquery para lidar com as interações básicas na página.

1.5.1 C#

C# é uma linguagem de programação de alto nivel, multiparadigma, orientada a objetos com tipagem forte e desenvolvida pela Microsoft para ser utilizada em conjunto com o framework .NET.

A linguagem surgiu em Abril de 2000 junto com a primeira versão do framework .NET, inicialmente sendo mantido e produzido apenas pela Microsoft porém sendo transformada em um projeto de código aberto a fim de reduzir custos e aumentar o contato com a comunidade.

O pacote .NET é um framework que busca conectar múltiplas livrarias em um único pacote de fácil uso do desenvolvedor, inicialmente integrando apenas as APIs do sistema Windows para aplicativos desktop, porém expandindo para aplicações web com a livraria ASP e eventualmente para múltiplos sistemas operacionais com projetos como Mono runtime.

A C# é uma linguagem de multiúso com uma linha de aprendizado relativamente alta, sendo utilizada para criação desde aplicativos padrões para Desktop como uma calculadora ou editor de texto se conectando automaticamente com as livrarias para renderização de janelas no sistema operacional, Videogames utilizando plataformas como Unity que se conecta com livrarias de vídeo e a criação de executáveis para múltiplos sistemas operacionais, livrarias para auxiliar com a criação de modelos de inteligência artificial e para a criação de páginas web utilizando ASP.

Junto ao C# veio a linguagem F# em 2005 com seu maior diferencial sendo seu foco em programação funcional ao invés de ser dedicada a orientação objeto.

1.5.2 Asp.Net

Asp se refere á tecnologia chamada Active Server Pages lançada pela microsoft em 1990 como a maneira oficial oferecida para criar páginas dinâmicas utilizando a linguagem VB script, atualmente substituída pela C# e F#.

O ASP.NET nasceu em 2002 com integração para C#, F# e visual Basic porem lentamente dando preferencia para C#/F# com atualizações futuras.

O ASP lida com diversos problemas relacionados ao desenvolvimento do lado servidor tipicamente utilizando o modelo MVC para controlar como os usuários acessam os dados do servidor, Muitas funcionalidades básicas já vem por padrão com o ASP, porém existem livrarias que expandem ainda mais como o entity framework para a criação de bancos de dados com código C# ou o Identity framework que tras consigo um sistema completo de login compatível com padrões modernos como e-mail de confirmação, autenticação de dois fatores e uma página pessoal de usuário.

1.5.3 Blazor

Blazor é um framework frontend para páginas web que permite substituir a linguagem Javascript pela C#. O diferencial do Blazor é que a tecnologia pode se integrar com as páginas Razor de um servidor criando API automáticas permitindo ter páginas de cliente similar a uma aplicação de página única com componentes renderizados pelo servidor e automaticamente enviados ao cliente.

Normalmente servidores web são divididos entre duas versões: um servidor API que conversa diretamente com uma aplicação Frontend separando a lógica empresarial da lógica visual de apresentação dos dados ou um servidor com páginas web renderizadas pelo servidor.

Razor é uma ferramenta similar ao Blazor, permitindo criar páginas HTML utilizando a linguagem C#, porem no lado do servidor.

1.5.4 Bootstrap e Popper

Bootstrap é uma livraria de estilos css originalmente criada e desenvolvida no Twitter com o nome de Twitter Blueprint, A livraria foi aberta ao publico e tornada open-source em 2011 e vem sendo atualizada desde então buscando seguir padrões modernos da web como flex-boxes e Sass.

O objetivo do Bootstrap é acelerar a criação de páginas dando um estilo padrão para os elementos e componentes prontos para serem utilizados, permitindo os desenvolvedores focarem apenas na lógica da aplicação sem se preocupar em criar um design responsivo para a página. A maneira que isto é feito é utilizando comandos javascript e css já inclusos na livraria.

A partir da versão 5 do bootstrap o pacote Jquery não é mais utilizado com um aplicativo Blazor utilizando por padrão a versão 5.10 (em 2024) do bootstrap.

O JQuery é uma biblioteca de funções Javascript que interagem com o HTML, criada em 2006 focando em padronizar o antigo javascript com comandos customizados para modificar o DOM, criada em um periodo onde nem todos os navegadores suportavam todas as funcionalidades da javascript ou do css.

No desenvolvimento de páginas web é importante utilizar estas livrarias para padronizar o código a fim de consistência de interface e manter suporte com navegadores diferentes com funcionalidades diferentes.

1.5.5 Banco de dados SQLite e Entity Framework

SQLite é um banco de dados criado por uma livraria em C com código aberto que busca implementar uma solução rápida, confiável com todas as capacidades de um banco SQL dedicado porém compactadas em um único arquivo local, escolhido devido ao tamanho do projeto e por que o Entity Framework e a maneira como o .NET isola as maneiras para se conectar com um banco de dados tornam fácil a transição para um banco de dados dedicado.

SQL é um modelo de banco de dados onde os dados são guardados em formato de tabelas controladas pelas linguagens de alto nivel internas do SQL e

normalmente sendo hospedado como um serviço separado do programa apenas acessado por comandos especiais.

O Entity Framework é uma livraria para auxiliar com a criação de bancos de dados criado pela Microsoft, o Entity Framework possui duas partes: o migrador e o abstraidor de banco. O migrador se responsabiliza em modelos escritos em C# e criar uma sequência válida de comandos SQL para criar o banco baseado no modelo. O abstraidor se responsabiliza em conectar com o banco de dados criados pelo migrador seguindo uma lógica que se distancia dos comandos SQL crus, preferindo utilizar comandos simplificados em C#.

O banco de dados do cinemaApp apenas guarda algumas informações simples: Um modelo de filme com título, descrição, imagem de pôster e outros dados pessoais ao filme, Um modelo de sessão para um filme especifico que guarda o identificador do filme junto aos dados de preço do ingresso e data com hora da sessão, Um modelo de ingresso que se relaciona entre um usuário e uma sessão de cinema para representar e um modelo de usuário para poder entrar em contato e identificar a pessoa que comprou um ingresso utilizando um e-mail, telefone e outras chaves únicas que o usuário preferir dispor, com e-mail sendo o identificador obrigatório.

2 METODOLOGIA

Para a construção do sistema proposto neste trabalho, o CinemaApp, primeiramente foi feito a seleção das bibliotecas e tecnologias, havendo alteração nas mesmas, isto é, conforme o sistema foi sendo desenvolvido foram surgindo novas necessidades de bibliotecas que foram implementadas com o passar do tempo.

A seleção de livrarias foi feita pela “loja” interna do gerenciador de pacotes nuget provido por padrão pela Microsoft para projetos em C#, as principais livrarias utilizadas foram escolhidas por resolverem problemas técnicos já resolvidos como a maneira para se conectar a um banco de dados utilizando o pacote SQLitePCL.

O motivo para usar estas livrarias prontas é evitar precisar solucionar problemas técnicos complexos que já foram resolvidos evitando uma solução com erros, vazamentos de memória ou falhas criticas de segurança básica porem ainda oferecendo serviços complexos como confirmação por e-mail ou autenticação dupla.

A maioria dos problemas técnicos encontrados durante o projeto foram resolvidos utilizando o auxílio do ChatGPT, uma inteligência artificial treinada com dados da internet para ajudar com pequenos problemas técnicos que não dependam de uma lógica complexa, e com buscas em sites como Stack Overflow, Reddit e o servidor Discord oficial da linguagem C# além claro da documentação hospedada no site da Microsoft.

O projeto foi iniciado pelo banco de dados, criando os modelos que serão guardados no banco ao mesmo tempo que foram criadas classes com interfaces para facilitar o contato com o banco, essas funções foram então transformadas em serviços (um objeto instanciado a nível global da aplicação).

A tela Inicial foi o primeiro componente criado com seus componentes filhos (cabeçalho e lista de filmes com sessões válidas), esta tela também seguiu o foco de minimalismo e não contém um conteúdo muito extenso. O cabeçalho possui apenas o nome do sistema com o ícone e um botão para acessar as ações da conta que está logada ou deixar claro que o usuário não esta logado.

Os componentes de login e register (entrar e cadastro) foram criados por padrão pelo Identity framework, contendo em seu corpo apenas um formulário para

efetuar a criação ou login da conta porém contendo código para enviar um e-mail de confirmação e formulários para alterar os dados pessoais do usuário pelo mesmo.

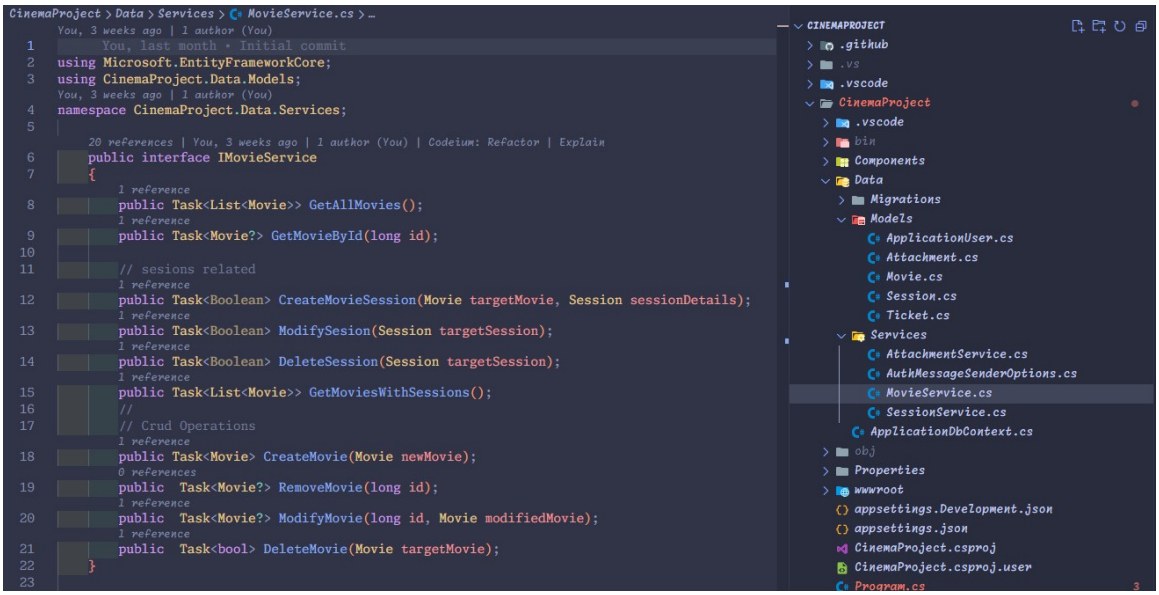
O componente de lista de filmes com sessões válidas foi criado junto à página inicial com apoio das funções já criadas para buscar os filmes no banco, utilizando dados fantasmas implantados manualmente no banco. O componente segue um desenho minimalista com fundo preto buscando criar uma lista com grande foco nas imagens dos posters de cada filme ao mesmo tempo que permite o usuário a ser redirecionado para a tela de compra após selecionar uma sessão da lista apresentada por um modal após clicar em qualquer filme da lista, sendo redirecionado para a tela de registro caso esteja deslogado.

Em seguida foram criados os formulários da área administrativa, seguindo um layout de cartões para mostrar os dados dos filmes com um botão para abrir um modal e modificar ou remover o filme ou adicionar sessões além de cada cartão ter uma lista com as sessões de cada filme e um modal para visualizar as pessoas que compraram ingressos para a sessão.

3 CONSTRUÇÃO DO SISTEMA.

O CinemaApp foi inicializado pelo banco de dados com a criação dos modelos para cada tabela no banco junto com a criação de classes de serviço que recebem o contexto do banco de dados e permitem buscar ou alterar dados de maneira fixa.

FIGURA 1 – Código serviço para lidar com filmes.



Os modelos buscam guardar dados especiais para cada tabela como título, URL da imagem de pôster, descrição, data de lançamento e categoria para o caso dos filmes. Cada tabela possui uma classe de serviço para interagir diretamente com o banco de dados, com funções básicas para puxar dados por identificadores, adicionar novos dados, modificar ou remover os dados do banco de dados.

3.1 Construção das páginas

A aplicação consiste em 3 páginas principais com telas extras para configuração da conta do usuário e login ou registro no sistema. A primeira página criada foi a página Inicial ou Home composta por uma seção de apresentação para introduzir o usuário ao sistema e um componente conectado ao banco de dados para buscar filmes com sessões válidas.

A página inicial foi atribuída como rota padrão ou rota “/”, significando que qualquer pessoa conectando pelo endereço padrão recebera esta página. As páginas foram todas criadas utilizando Razor para misturar C# com HTML.

A função da tela Inicial é permitir os usuários facilmente identificarem as sessões disponíveis e comprar um ingresso rapidamente, redirecionando usuários não logados para o formulário de cadastro enquanto usuários logados são direcionados para a tela de compra.

3.1.1 Entrar e cadastrar

O processo de cadastro junto ao de login é realizado enviando uma página ao cliente com um formulário. Ao enviar o formulário o servidor recebe os dados do formulário pela integração Blazor e automaticamente lida com o pedido do cliente.

O registro de usuário junto ao login busca no banco de dados um usuário com um e-mail especifico utilizado para buscar um usuário específico, caso o usuário exista o login testará a senha e logara o usuário ou retornará um erro enquanto o registro apenas cria uma nova entidade de usuário no banco com os dados do usuário caso o e-mail não entregue uma conta existente. Caso uma operação falhe o usuário é apenas retornado um erro e redirecionado ao formulário.

3.1.2 Área de pagamento

A área de pagamento apenas mostra a sessão que o usuário está comprando junto com seu valor, o cinemaApp por padrão apenas cria um relacionamento no banco de dados quando o usuário clica comprar e não executa nenhuma transação de pagamento verdadeira.

Porem para aplicações reais é possível utilizar serviços como Stripe para automatizar a conexão entre o aplicativo e o banco responsável pelo pagamento do usuário.

3.1.3 Área administrativa

A área administrativa é uma página simples com um cabeçario adicional pada adicionar filmes, que são mostrados como cartões interagiveis com dados de cada filme junto ao seu poster e um simples formulário para cada ação relevante ao filme e suas sessões.

4 VALIDAÇÃO

A análise deste projeto foi feita do ponto de vista de desenvolvimento, isto é, foram feitas avaliações por parte dos desenvolvedores em relação a criação do sistema e objetivos propostos.

Todas as funcionalidades propostas foram implementadas no sistema, com o uso e auxílio das as tecnologias mencionadas nos capítulos anteriores. As tecnologias usadas tiveram um bom desempenho para a construção do CinemaApp, cumprindo corretamente com as funções pretendidas e gerando poucos bugs e erros para serem concertados ao longo do desenvolvimento. O que proporcionou maior fluidez no processo de construção do sistema.

4.1 Análise das tecnologias utilizadas

Em questão de tecnologias, destaca-se o uso do Razor para a criação das páginas HTML e Blazor para torná-las interativas, com essas tecnologias a aplicação teve uma pequena quantidade de bibliotecas utilizadas ao mesmo tempo que facilitou a criação de todo o frontend da aplicação ao mesmo tempo que o backend era unificado de forma invisível e simplificada. enquanto o SQLite lidou de maneira eficiente com os pedidos em ambiente de desenvolvimento.

A velocidade da aplicação não é diferente de outras aplicações, porém com a C# é possível facilmente executar o servidor em qualquer maquinário compilando o projeto ao mesmo tempo que inclui todo o ambiente de execução no arquivo executável para um sistema operacional e arquitetura de processador.

O Bootstrap foi usada apenas para tarefas menores e responsáveis por alguns aspectos de responsividade e dinâmica do sistema. Ainda assim a escolha de tecnologias para tarefas menores, é algo importante em qualquer projeto de programação, pois concedem agilidade ao desenvolvimento e economizando esforços de tempo. Um grande exemplo disso são funcionalidades de grid e flexbox do Bootstrap, que é implementado no atributo “classe” dos elementos, fazendo com que não seja necessário criar a responsividade de certas telas com códigos

extensos e complexos em css, que hoje é o principal mecanismo de estilo para sistemas web.

4.2 Análise geral

Foram observados inúmeros pontos positivos no sistema construído, que foi capaz de alcançar todos os objetivos propostos de desenvolvimento e de forma que todas as funcionalidades se encaixaram perfeitamente com a proposta de um sistema para a venda de ingressos. Além de ter superado as nossas expectativas, foi analisado que o sistema foi finalizado com grandes possibilidades de mudanças e atualizações futuras, aprimorando-o cada vez mais.

No que toca as atualizações do sistema, foi considerada também a forma com que o sistema foi construído, gerando uma maior facilidade na busca e correção de erros e bugs. Esta facilidade está ligada com as funcionalidades de componentização do Razor junto com a modalidade forçada de programação orietada a objetos da C#. o sistema tem uma boa eficiência quando se trata de reparos, fazendo com que possam ser realizados com maior rapidez caso surjam bugs e erros.

5 CONCLUSÃO

O CinemaApp atingiu todos os objetivos propostos no projeto em questão de programação e desenvolvimento. Ao longo do projeto foi estudado e pesquisado como ocorre a mesclagem de duas áreas, neste caso, a economia e a tecnologia. O que se fez a partir de estudos focados em prática de conhecimento, que foram aplicados no sistema a partir da construção de questões, cumprindo com um dos principais objetivos deste projeto, que era estudar a união da tecnologia web com a economia.

Foi concluído que as tecnologias usadas apresentaram um comportamento adequado, cada uma fazendo seu papel dentro do sistema. Tanto o design quanto a programação superaram expectativas em relação às propostas iniciais, contando com, uma programação repleta de bibliotecas auxiliares, com inúmeras funcionalidades implementadas, tanto complexas quanto simples, em especial no que toca o design, desde o início foi proposto um design minimalista, com poucos detalhes, porém atrativo, além de algumas funcionalidades de design como modo claro e escuro. Características que foram implementadas no sistema com o uso de bibliotecas auxiliares disponibilizadas por comunidades de programação na Internet.

O uso de bibliotecas auxiliares foram de suma importância no projeto, o uso de bibliotecas foi extremamente pequeno com o uso destas bibliotecas fazendo com que inúmeras funcionalidades fossem implementadas de maneira rápida e fácil desde tarefas simples até tarefas mais complexas, o que facilitou o desenvolvimento da aplicação.

O sistema é semi-dinâmico, responsivo e assíncrono, isto é, as funcionalidades de comunicação com o backend pelo Blazor não necessitam de atualizar a página. Todo o processo de desenvolvimento foi pensando na experiência do usuário no sistema CinemaApp.

REFERÊNCIAS BIBLIOGRÀFICA

[1]

M. O. contributors Jacob Thornton, and Bootstrap, “About”. Acesso em: 9 de setembro de 2024. [Online]. Disponível em: https://getbootstrap.com/docs/5.3/about/overview/

[2]

“Apple I”, *Wikipedia*. 7 de setembro de 2024. Acesso em: 9 de setembro de 2024. [Online]. Disponível em: https://en.wikipedia.org/w/index.php?title=Apple_I&oldid=1244441484

[3]

“ASP (Active Server Pages)”, Mailrelay. Acesso em: 9 de setembro de 2024. [Online]. Disponível em: https://mailrelay.com/en/glossary/asp-active-server-pages/

[4]

guardrex, “ASP.NET Core Blazor”. Acesso em: 9 de setembro de 2024. [Online]. Disponível em: https://learn.microsoft.com/en-us/aspnet/core/blazor/?view=aspnetcore-8.0

[5]

“C Sharp (programming language)”, *Wikipedia*. 6 de setembro de 2024. Acesso em: 9 de setembro de 2024. [Online]. Disponível em: https://en.wikipedia.org/w/index.php?title=C_Sharp_(programming_language)&oldid=1244362359

[6]

“Comparing F# with C#: A simple sum | F# for fun and profit”. Acesso em: 9 de setembro de 2024. [Online]. Disponível em: https://fsharpforfunandprofit.com/posts/fvsc-sum-of-squares/

[7]

“Creating Custom Database Tables with Entity Framework | Umbraco CMS”. Acesso em: 10 de setembro de 2024. [Online]. Disponível em: https://docs.umbraco.com/umbraco-cms/tutorials/getting-started-with-entity-framework-core

[8]

“History of cloud computing”, *Wikipedia*. 26 de julho de 2024. Acesso em: 26 de agosto de 2024. [Online]. Disponível em: https://en.wikipedia.org/w/index.php?title=History_of_cloud_computing&oldid=1236708640

[9]

gewarren, “Introduction to .NET - .NET”. Acesso em: 26 de agosto de 2024. [Online]. Disponível em: https://learn.microsoft.com/en-us/dotnet/core/introduction

[10]
“Introduction to the IETF”, IETF. Acesso em: 26 de agosto de 2024. [Online]. Disponível em: https://www.ietf.org/about/introduction/

[11]
“MacBook”, *Wikipédia, a enciclopédia livre*. 10 de novembro de 2023. Acesso em: 9 de setembro de 2024. [Online]. Disponível em: https://pt.wikipedia.org/w/index.php?title=MacBook&oldid=66944249

[12]
“Moore’s law”, *Wikipedia*. 16 de agosto de 2024. Acesso em: 22 de agosto de 2024. [Online]. Disponível em: https://en.wikipedia.org/w/index.php?title=Moore%27s_law&oldid=1240585254

[13].
“NET”, *Wikipedia*. 23 de agosto de 2024. Acesso em: 26 de agosto de 2024. [Online]. Disponível em: https://en.wikipedia.org/w/index.php?title=.NET&oldid=1241776330

[14].
“NET compilation process explained (C#)”, DEV Community. Acesso em: 26 de agosto de 2024. [Online]. Disponível em: https://dev.to/kcrnac/net-execution-process-explained-c-1b7a

[15]
“Service-oriented architecture”, *Wikipedia*. 24 de julho de 2024. Acesso em: 22 de agosto de 2024. [Online]. Disponível em: https://en.wikipedia.org/w/index.php?title=Service-oriented_architecture&oldid=1236468830

[16]
“Software as a service”, *Wikipedia*. 8 de agosto de 2024. Acesso em: 22 de agosto de 2024. [Online]. Disponível em: https://en.wikipedia.org/w/index.php?title=Software_as_a_service&oldid=1239365762

[17]
“SQLite Home Page”. Acesso em: 10 de setembro de 2024. [Online]. Disponível em: https://www.sqlite.org/

[18]
erikdietrich, “The history of C#”. Acesso em: 26 de agosto de 2024. [Online]. Disponível em: https://learn.microsoft.com/en-us/dotnet/csharp/whats-new/csharp-version-history

[19]

v8/v8. (26 de agosto de 2024). C++. V8. Acesso em: 26 de agosto de 2024. [Online]. Disponível em: https://github.com/v8/v8

[20] "W3C", W3C. Acesso em: 26 de agosto de 2024. [Online]. Disponível em: https://www.w3.org/