CENTRO ESTADUAL DE TECNOLOGIA PAULA SOUZA
FACULDADE DE TECNOLOGIA DE MAUÁ

MELISSA DE FREITAS SANTOS

SISTEMA DE GERENCIAMENTO PARA GAME HOUSE

MAUÁ/SP
2023

MELISSA DE FREITAS SANTOS

SISTEMA DE GERENCIAMENTO PARA GAME HOUSE

Monografia apresentada à FATEC Mauá, como parte dos requisitos para obtenção do Título de Tecnólogo em Informática para Negócios.
Orientador: Prof. Me. Ivan Carlos Pavão

Comentado [mf1]: 1º orientação: 13/02

MAUÁ/SP

2023

Santos, Melissa de Freitas.
Sistema de gerenciamento para Game House. Santos, Melissa de Freitas

0 p.; 30 cm.

TCC (Trabalho de Conclusão de Curso).
CEETEPS-FATEC Mauá/SP, 0º Sem. 2023.
Orientador: Prof. M. Sc.

Referências: p. 0.
Palavras-chave: Informação, Logística, Sistema, Game House.

MELISSA DE FREITAS SANTOS

SISTEMA DE GERENCIAMENTO PARA GAME HOUSE

Monografia apresentada à FATEC Mauá, como parte dos requisitos para obtenção do Título de Tecnólogo em Informática para Negócios.

Aprovação em: 0 jun. 2023.

_____

Prof. Me. Ivan Carlos Pavão

FATEC Mauá

Orientador

_____

Prof (a) _____

FATEC Mauá

Avaliador(a)

_____

Prof (a) _____

FATEC Mauá

Avaliador(a)

Dedicarei este trabalho, primeiramente, a Oxalá, por ser essencial em minha vida, aos meus pais, pelos incentivos, os quais tornaram possível a conclusão desta etapa em da minha vida e, por fim, aos professores que me auxiliaram nessa jornada de estudo.

“As coisas mais importantes não estão escritas num livro, é preciso aprendê-las vivenciando-as sozinho.” - Sakura Haruno (Naruto)

Comentado [LC-W2]: Perguntar se pode ser usado

RESUMO

O presente trabalho

Comentado [LC-W3]: Fazer resumo.

Comentado [mf4R3]: Introdução+Conclusão

Palavras-chave: Game House, Tecnologias, MVC, Desenvolvimento, Sistema.

ABSTRACT

Comentado [LC-W5]: Adicionar Abstract.

Keywords: Information, Logistics, System, Game House.

Lista de Figuras

Lista de Tabelas

Lista de Diagramas

Lista de Telas

1. INTRODUÇÃO

Tecnologia se tornou algo essencial na vida das pessoas, facilitando seu dia a dia. Atualmente ela vem sendo inovada em grande velocidade e trazendo otimização nos sistemas. Muitas empresas necessitam de sistemas para auxiliar no gerenciamento e tomada de decisão.

Com a crescente onda de jogos na era digital, muitas pessoas começaram a comprar jogos e consoles, porém muitos se arrependem de suas escolhas por serem caros ou não gostarem. Visto isso, foi criado uma adaptação das antigas Locadoras, chamado Game House, onde as pessoas podem ir para alugarem jogos e levarem para casa ou até mesmo jogar no local utilizando os consoles da loja, os mesmos que servem para testes de comparação de qualidade, usabilidade, entre outros fatores, de um console para outro e de jogo para outro.

Desta maneira, o presente trabalho tem como objetivo apresentar a prototipagem de um sistema de gerenciamento que facilite a administração de uma Game House, para que o sistema seja efetivo deve-se ser cadastrado os funcionários, os clientes, os produtos, os agendamentos, os pagamentos e em função disso, foi feito um planejamento após análises de requisitos necessários para o desenvolvimento, que serão apresentados ao decorrer do trabalho.

Considerando a grande relevância que um sistema tem para uma empresa, trazendo uma dependência cada vez maior para o fluxo de serviços aplicados pelas entidades empresariais, esse tipo de ferramenta trará valor para a empresa, centralizando os dados, trazendo a agilidade e facilitando na análise da situação da organização, garantindo o melhor desempenho empresarial.

Este trabalho foi redigido por meio de pesquisas bibliográficas e on-line. Através de experiências de profissionais que a autora do trabalho vivenciou, percebe-se a grande necessidade da criação de softwares específicos para o gerenciamento de organizações.

Serão abordados os temas: Game House, tecnologias utilizadas no desenvolvimento, arquitetura Model-View-Controller, desenvolvimento e o manual do sistema. Ao final das abordagens do tema será apresentado, também, o protótipo do projeto proposto pela autora do trabalho.

2. CONCEITUANDO UMA GAME HOUSE

Comentado [mf6]: Conceituar a game house

https://www.legiaodosherois.com.br/2019/locadoras-de-games-no-brasil-mais-que-diversao-um-estilo-de-vida.html

https://www.youtube.com/watch?v=H4ZAEhuZpjA

2.1 Modelo de negócio

Comentado [SdE-FM7]: Adicionar definição do modelo de negócio.

Um modelo de negócio é uma generalização do próprio negócio. Na visão do desenvolvimento de sistemas de informação, pode-se concluir que os processos e as regras são os objetos mais importantes do modelo de negócio, pois é a partir desses objetos que são definidos os requisitos de sistema de informação que apoiarão os procedimentos de um determinado negócio. (DIAS el at., 2014)

Comentado [mf8]: DIAS, Felipe; Morgado, Gisele; Oscar, Pedro; Silveira, Denis; Alencar, Antonio J.; Lima, Priscila; Schmitz, Eber. **Uma Abordagem para a Transformação Automática do Modelo de Negócio em Modelo de Requisitos.** NCE – IM – Universidade Federal do Rio de Janeiro. Publicado em 28 feb. 2014. Disponível em: <https://www.researchgate.net/profile/Antonio-Alencar/publication/221235083_Uma_Abordagem_para_a_Transformacao_Automatica_do_Modelo_de_Negocio_em_Modelo_de_Requisitos/links/0c960531014620a3a8000000/Uma-Abordagem-para-a-Transformacao-Automatica-do-Modelo-de-Negocio-em-Modelo-de-Requisitos.pdf>.

Como objeto de estudo e prototipagem, foi escolhido utilizar um empreendimento fictício de uma locadora de games ou Game House, e para que o software tivesse regras de negócio baseado, a autora criou um plano de negócio resumido que contemplasse algumas exigências para o negócio fictício.

Tipo de empreendimento	Locadora de games
Nome fantasia do sistema	Violet_Games
Clientes	Jogadores de todas as idades, pessoas que querem testar algo novo ou tem medo de comprar o jogo/videogame e não gostar.
Proposta de valor	Sala de jogos equipada com consoles disponíveis + Jogos compatíveis para esses consoles que podem ser jogados no espaço ou alugados.
Relacionamentos com clientes	O plano mensal permite que o cliente utilize a sala de jogos com prioridade no agendamento de uso, além de poder alugar 3 jogos por vez. Os 10 primeiros clientes que assinarem o plano mensal, terão um desconto de 50% na mensalidade por 6 meses, os aniversariantes ganham 10% de desconto no plano mensal.
Fontes de receita	A sala de jogos pode ser usada por meio de agendamento, qualquer console pode ser usado pelo valor de R$4,00/H, para locação de jogos o valor da diária é R$8,00/por jogo. Plano mensal R$ 49,90 (inclui o uso da sala de jogos + locação de jogos).
Recursos principais	Xbox 360, Xbox One, PS1, PS2, PS3, PS4, PS5.

Atividades-chave	Agendamento do uso da sala de jogos, locação dos jogos, monitoramento do uso consoles/jogos.

Tabela 1 - Plano de negócio [Autora]

O protótipo deve auxiliar os funcionários da Game House e para isso o desenvolvedor deve entender melhor o negócio de forma macro, assim visualizar um plano de negócio como o apresentado faz com que o desenvolvedor entenda as necessidades dos clientes, nesse caso, dos futuros usuários e procure modelar um software que atenda a demanda com eficiência e simplicidade.

3. TECNOLOGIAS UTILIZADAS NO DESENVOLVIMENTO

3.1 C#, .NET e ASP.NET

Para a criação de sistemas, escolher a linguagem e as tecnologias a serem utilizadas é um passo fundamental, normalmente, a escolha da linguagem é determinada a partir da plataforma, da natureza e da cultura da empresa, para esse projeto a plataforma principal escolhida foi o .NET 5 e como linguagem o C#.

Conforme a documentação oficial da Microsoft (2023), o .NET (DotNet) é um sistema de execução virtual chamado Common Language Runtime (CLR) e um conjunto de classes. O CLR é a implementação da Microsoft da Common Language Infrastructure (CLI), um padrão internacional. A CLI é a base para a criação de ambientes de execução e desenvolvimento nos quais as linguagens e bibliotecas funcionam em conjunto diretamente.

Figura 1 - .NET 5 [devblogs]

Comentado [mf9]: MICROSOFT. **Introdução ao .NET Framework - .NET Framework**. Publicado em 10 mar. 2023. Disponível em: <https://docs.microsoft.com/pt-br/dotnet/framework/get-started/>. Acesso em: 10 mar. 2022.

A versão 5 desse framework unifica a criação de aplicativos executados em todas as plataformas (Windows, Linux) e dispositivos (IoT, Mobile), possibilitando o melhor desempenho para o desenvolvedor na criação de sistemas.

Ainda em conformidade com a documentação da Microsoft (2023), o C# (CSharp) é uma linguagem de programação orientada a objetos e orientada a componentes, criada pela Microsoft e lançada no mesmo mês do lançamento do .NET 1.0, tendo como base a familia da linguagem de programação C, além de ter sido influênciada por outras linguagens de programação, como Pascal e Java.

Foi desenvolvido por Anders Hejlsberg, um engenheiro de software dinamârques em conjunto com uma equipe de programadores que trabalhavam para a Microsoft. A linguagem foi desenvolvida no intuito de flexibilizar o desenvolvimento de aplicativos e possibilitar a criação de soluções executáveis sobre a plataforma .NET, assim o desenvolvedor não cria soluções para um dispositivo de aplicativos, e sim para a plataforma .NET. (DEVMEDIA, 2013)

Comentado [mf10]: MICROSOFT. **O histórico da linguagem C# – Guia do C#**. Publicado em 8 mar. 2023. Disponível em: <https://docs.microsoft.com/pt-br/dotnet/csharp/whats-new/csharp-version-history>. Acesso em: 10 mar. 2022.

Comentado [mf11]: DEVMEDIA. **A evolução da linguagem de programação C#**. Publicada em: 2013. Disponível em: <https://www.devmedia.com.br/a-evolucao-da-linguagem-de-programacao-csharp/28639>. Acesso em: 26 mar. 2023.

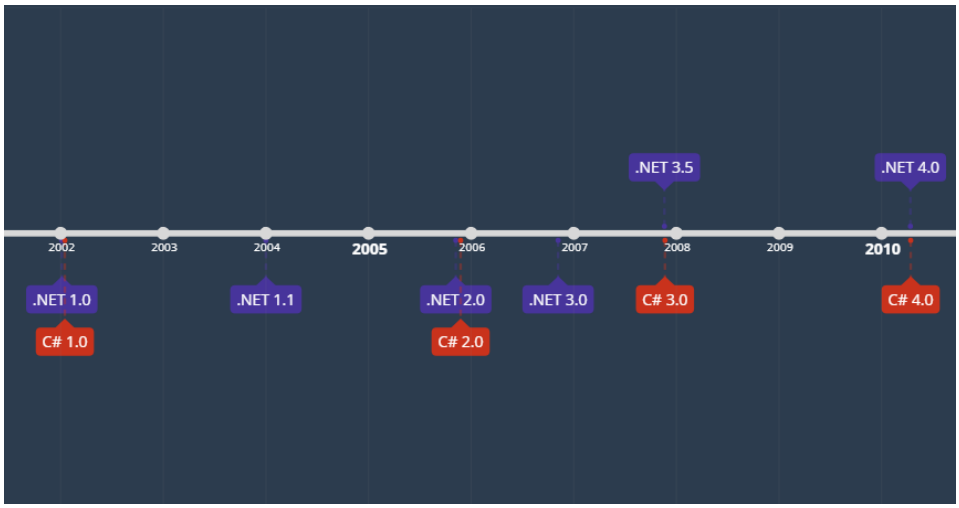


Figura 2 – Evolução do C# e .Net no Período de 2002 a 2010 [time.graphics]

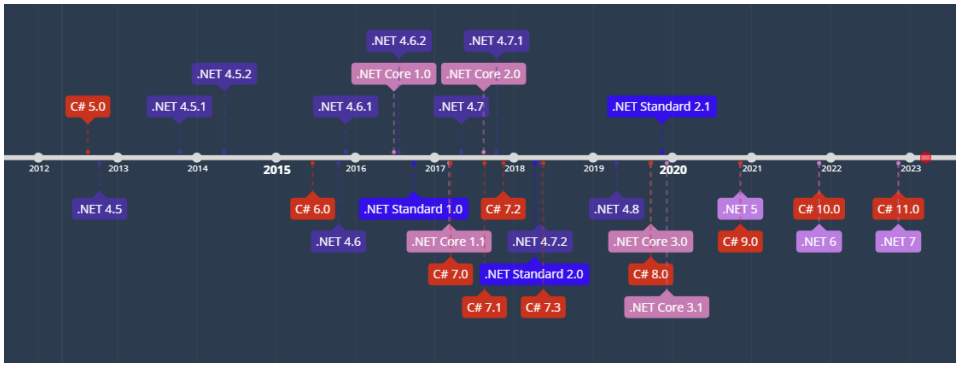


Figura 3 – Evolução do C# e .Net no Período de 2012 a 2023 [time.graphics]

"Dentre algumas das características essenciais do C#, em ordem cronológica de versão, destacam-se: (1.0) totalmente orientada a objetos, gerenciamento automático de memória; (2.0) tipos genéricos, métodos anônimos, classes estáticas; (3.0) tipos anônimos, expressões de consulta e árvore de expressões; (4.0) propriedades indexadas, suporte COM e DLR; (5.0) Async e Caller Information." (DEVMEDIA, 2013)

Atualmente o C# está em sua versão 11, como foi destacado acima cada versão trouxe um ponto de melhoria, nessa versão foram adicionados atributos genéricos entre outros recursos. Porém é o compilador do C# mais recente que determina qual versão da linguagem a ser utilizada com base nas estruturas de destino do projeto, e como apontado pela autora esse projeto utilizará o .Net 5 e para essa versão do framework o C# 9.0 é a versão utilizada, nela foi adicionado o recurso de suporte a geradores de código C#.

De acordo com Sanchez e Althmann (2013), conforme o cenário web transformou-se rápida e largamente (mudança alavancada especialmente pelos negócios, que igualmente passaram a ser realizados neste ambiente), fez-se necessária a estruturação de uma nova tecnologia para desenvolvimento web incorporado da plataforma .NET, que atendesse às novas demandas de mercado.

Comentado [mf12]: SANCHEZ, F.; MÁRCIO FÁBIO ALTHMANN. **Desenvolvimento web com ASP.NET MVC**. [s.l.] Editora Casa do Código, 2013.

Surgiu em 2002 o ASP.NET, um framework projetado e desenvolvido pela Microsoft, usado para criar aplicativos da web na estrutura do .NET, como por exemplo, serviços e sites dinâmicos. O ASP.NET é o sucessor da ferramenta ASP (Active Server Pages) e uma extensão da plataforma .NET com ferramentas e bibliotecas adicionais, sua estrutura é baseada no protocolo HTTP padrão, que é o protocolo padrão utilizado em todos os aplicativos web.

https://blog.betrybe.com/framework-de-programacao/asp-net-o-que-e/

https://acervolima.com/introducao-ao-asp-net/

3.2 SQL Server

Nos anos 70, a IBM sentia a necessidade de um método padronizado para a manipulação de dados em um banco de dados relacional. Para esse objetivo, Dr. E. F. Codd, pesquisador da IBM, desenvolveu um produto chamado de SEQUEL, o qual mais tarde viria a ser rebatizado de SQL. Codd, em sua teoria, aplicou regras matemáticas rigorosas para a manipulação dos dados, obedecendo a uma lista de critérios que um banco de dados deve respeitar para ser considerado relacional. (BUSS el at., 2013 apud KLINE, 2010)

Em 1981, a IBM lançou a primeira implementação da linguagem como parte do sistema de gerenciamento de banco de dados relacional System R (RDBMS). A popularidade do SQL foi crescendo, sendo adotado por outros fornecedores de RDBMS, como Oracle, Sybase e Microsoft. Respectivamente em 1986 e 1987, o American National Standards Institute (ANSI) e a International Standards Organization (ISO) publicaram padrões SQL oficiais.

O SQL Server é um Sistema de Gerenciamento de Banco de Dados (SGBD) da empresa Microsoft, foi criado em parceria com a Sybase, em 1988, inicialmente como um complemento do Sistema Operacional Windows NT, e logo depois passou a ser refinado e vendido separadamente. A parceria da Microsoft com a Sybase terminou em 1994, e o desenvolvimento do programa continuou a ser feito pela Microsoft. (WIKIPÉDIA, 2023)

Esse banco de dados é classificado como um Banco de Dados Relacional (RDBMS, na sigla em inglês), um banco de dados relacional é um tipo de banco de dados que armazena e fornece acesso a pontos de dados relacionados entre si. Bancos de dados relacionais são baseados no modelo relacional, uma maneira intuitiva e direta de representar dados em tabelas. Em um banco de dados relacional, cada linha na tabela é um registro com uma ID exclusiva chamada chave. As colunas da tabela contêm atributos dos dados e cada registro geralmente tem um valor para cada atributo, facilitando o estabelecimento das relações entre os pontos de dados. (ORACLE)

Comentado [mf13]: OK

Comentado [mf14]: BUSS, Liana; Punchirolli, Lucas C.; Olivo, Ricardo R. **Análise das vulnerabilidades encontradas em técnicas de programação PHP para tratamento de queries SQL**. Universidade do Estado de Santa Catarina - Centro de Ciência e Tecnologia – CCT. Joinville, 2013.

Comentado [mf15]: W3SCHOOLS. **SQL History**. Disponível em: <https://www.w3schools.in/sql/history>. Acesso em: 23 mar. 2023.

Comentado [mf16]: WIKIPÉDIA. **Microsoft SQL Server**. 24 feb 2023. Disponível em: <https://pt.wikipedia.org/wiki/Microsoft_SQL_Server>. Acesso em: 28 feb. 2023.

Comentado [mf17]: ORACLE. **O que é um banco de dados relacional?** Disponível em: <https://www.oracle.com/br/database/what-is-a-relational-database/>. Acesso em: 23 feb. 2023.

Como o próprio nome sugere, esse banco de dados utiliza os padrões da linguagem SQL, atualmente, esse SGBD continua a ser um dos mais utilizados, ele possui versões gratuitas e pagas. As versões posteriores ao SQL Server 2008 R2 impressionaram desenvolvedores pela simplicidade, diminuindo o tempo para a criação do banco de dados, além de permitirem ao desenvolvedor usar uma linguagem de programação gerenciada, como C# ou VB.NET, para o endereçamento das consultas, ao invés de usar declarações SQL. Outra vantagem são as consultas transparentes orientadas ao conjunto, escritas em .NET.

O SQL Server foi escolhido como umas das tecnologias que se integram nesse protótipo, pois é o banco de dados que melhor atende as necessidades do projeto e por ser a ferramenta apresentada nas aulas de banco de dados ministradas pelo orientador desse trabalho de conclusão.

3.3 Entity Framework Core

Comentado [mf18]: OK

O EF Core exerce o papel de um ORM (Object-Relational Mapping) que é um mapeador de objetos relacionais, sendo sua principal funcionalidade: permitir que os desenvolvedores do .NET trabalhem com o com um banco de dados usando objetos .NET, eliminando a necessidade da maior parte do código de acesso a dados que normalmente precisa ser gravado. (MICROSOFT, 2022)

Comentado [mf19]: MICROSOFT. **Visão geral do Entity Framework Core – EF Core**. Publicado em 28 set. 2022. Disponível em: <https://learn.microsoft.com/pt-br/ef/core/>. Acesso em: 10 mar. 2022.

Essa ferramenta pode acessar diversos bancos de dados distintos por meio de bibliotecas de plug-in chamadas de provedores de banco de dados, um dos provedores que é possível seu acesso é o SQL Server que como apresentado é uma das tecnologias escolhidas pela autora para esse projeto, a EF Core auxiliará na persistência e interação com o base de dados.

Seguindo a documentação da ferramenta, é descrito que o acesso a dados feito pelo EF Core é executado usando um modelo, esse modelo possuí classes de entidade e os objetos da camada de negócio que representa uma sessão com o banco de dados. Esse processo de acesso ao banco de dados extingue a necessidade da codificação de acesso a dados diretamente na aplicação, como por exemplo, o uso claro de instruções SQL. (MICROSOFT, 2022)

Comentado [mf20]: MICROSOFT. **Visão geral do Entity Framework Core – EF Core**. Publicado em 28 set. 2022. Disponível em: <https://learn.microsoft.com/pt-br/ef/core/>. Acesso em: 10 mar. 2022.

3.4 HTML, CSS e JavaScript
3.5 Bootstrap

4 ARQUITETURA MODEL-VIEW-CONTROLLER (MVC)

Na fase de projeto, o tópico fundamental é a escolha e o desenho da arquitetura da aplicação, Martin Fowler (2018) considera que a arquitetura de uma aplicação possui dois objetivos essenciais: decompor esse sistema em suas partes principais, em alto nível, e representar um modelo geral de forma estável, ou seja, sem grande tendência a alterações. Após muitas pesquisas sobre arquiteturas existentes e as mais utilizadas no mercado de desenvolvimento, foi avaliado que a arquitetura mais compatível para a criação do sistema proposto, seria a Arquitetura Model-View-Controller.

O Model-View-Controller conhecido como MVC, foi desenvolvido na década de 70, pelo cientista da computação norueguês e professor emérito da Universidade de Oslo, Trygve Mikkjel Heyerdahl Reenskaug enquanto trabalhava na Xerox PARC. Utilizando a plataforma de desenvolvimento ASP.NET MVC da Microsoft, esse modelo contribuiu para a redução do acoplamento entre classes, auxiliando no reuso.

O MVC consiste na partição do código do software em três camadas funcionais para serem independentes, criando assim uma facilidade na manutenção do código e sua reutilização em outros projetos. As três camadas são nomeadas de Model (Modelo), View (Visualização) e Controller(Controlador).

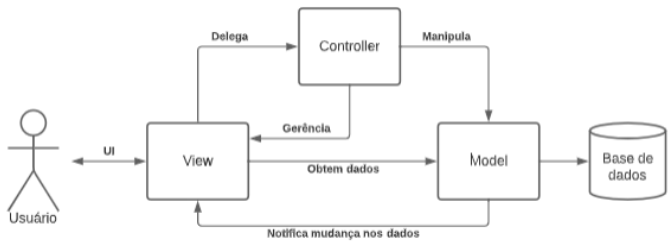


Diagrama 1 - Diagrama do modelo MVC [Autora]

Conceito do padrão MVC (ARAÚJO, 2018):

1. O controlador (Controller), é responsável pelas interpretações das entradas do mouse ou do teclado enviadas pelo usuário, assim ele mapeia essas ações do

Comentado [mf21]: FOWLER, Martin. **Padrões de Arquitetura de Aplicações Corporativas**. Bookman, São Paulo, 9 jul. 2018.

Comentado [mf22]: BURBECK, Steve. **Applications Programming in Smalltalk-80(TM): How to use Model-View-Controller (MVC)**. Disponível em: <https://folk.universitetetioslo.no/trygver/themes/mvc/mvc-index.html>. Acesso em: 10 mar. 2022.

Comentado [mf23]: ARAÚJO, Everton Coimbra. **ASP.NET Core MVC: Aplicações modernas em conjunto com o Entity Framework.** São Paulo: Casa do Código, 2018.

usuário em comandos que são enviados para o modelo (Model) e/ou para a janela de visualização (View) para executar a modificação apropriada;

2. O modelo (Model), faz o gerenciamento de um ou mais elementos de dados, respondendo a perguntas sobre o seu estado, e respondendo a instruções para alterar de estado. O modelo sabe o que o aplicativo quer executar e é a principal estrutura computacional da arquitetura, pois é ele quem modela o problema a ser resolvido;
3. Por fim, a visão (View) faz o gerenciamento do espaço retangular do display e é responsável por exibir as informações para o usuário através de uma combinação de gráficos e textos. A visão não sabe nada sobre o que a aplicação está atualmente fazendo, pois tudo que ela efetivamente faz é receber instruções do controle e informações do modelo e logo exibi-las. A visão igualmente se comunica de volta com o modelo e com o controlador para reportar o seu estado.

Existem vários tipos de Arquiteturas, a MVC por ser uma arquitetura que divide o projeto em três camadas é confundido em várias ocasiões com a Arquitetura de Aplicação de Três Camadas, embora ambas tenham o mesmo primórdio em separarem a aplicação em três camadas há fatores que as diferenciam.

A arquitetura Três Camadas possui três camadas principais, sendo elas: Apresentação, Domínio e fonte de dados. As três camadas dessa arquitetura estão sendo representadas em resumo na tabela abaixo. (BROWN et al. apud FOWLER, 2018)

Comentado [mf24]: FOWLER, Martin. **Padrões de Arquitetura de Aplicações Corporativas**. Bookman, São Paulo, 9 jul. 2018.

Tabela 1.1 Três Camadas Principais

Camada	Responsabilidades
Apresentação	Fornecimento de serviços, exibição de informações (p. ex., em Windows ou HTML, tratamento de solicitações do usuário (cliques com o *mouse*, pressionamento de teclas), requisições HTTP, chamadas em linhas de comando, API em lotes)
Domínio	Lógica que é o real propósito do sistema
Fonte de Dados	Comunicação com os bancos de dados, sistemas de mensagens, gerenciadores de transações, outros pacotes

Figura 4 - Três camadas principais [Brown et al. apud Fowler, 2018]

Para entender a diferença, Leonel Sanches da Silva (Arquiteto de Soluções da Twilio), explicou em uma resposta direcionada a um internauta no site StackOverflow, que é necessário entender do ponto de vista de um Model. Um Model é uma classe que define

Comentado [mf25]: **c# - 3 camadas vs MVC**. Disponível em: <https://pt.stackoverflow.com/questions/33352/3-camadas-vs-mvc>. Acesso em: 20 jan. 2023.

não apenas os elementos de dados, mas quais valores eles podem receber, como são validados e as relações de um Model com outro, coisa que não existe no Modelo de três Camadas. Na arquitetura de três Camadas, é necessário colocar validações, relações e características de cada entidade na camada de dados, ou na camada de negócio.

A responsabilidade de um Controller é a de harmonizar e arbitrar as relações entre Models. É ele que comanda a criação, modificação, exclusão e seleção dos dados da aplicação. Além disso, é ele que recebe a requisição e decide o que deve ser retornado como apresentação, como por exemplo, o formato dos dados (HTML, JSON, e assim por diante).

Há abordagens que procuram colocar uma camada extra para trabalhar juntamente com o Controller, sob a alegação de que não é responsabilidade do Controller de cuidar de regras de negócio. Isto não é verdadeiro, se for considerado como aspecto algo que ele é responsável por fazer, no caso, a harmonização de dados entre Models.

5 DESENVOLVIMENTO

5.1 Requisitos do sistema

Requisitos em sua descrição consiste na identificação documentada de uma característica ou comportamento que um produto deve atender. São a princípio, para capturar e transmitir necessidades, gerenciar expectativas, priorizar e atribuir trabalho, verificar e validar o sistema (aceitação) e gerenciar o objetivo do projeto.

> *Requisitos definem o que um sistema deve fazer e sob quais restrições. Requisitos relacionados com a primeira parte dessa definição — o que um sistema deve fazer, ou seja, suas funcionalidades — são chamados de Requisitos Funcionais. Já os requisitos relacionados com a segunda parte — sob que restrições — são chamados de Requisitos Não-Funcionais. (VALENTE, 2020)*

Comentado [mf26]: VALENTE, M. T. **Cap. 3: Requisitos – Engenharia de Software Moderna**. Publicado em 2020. Disponível em: <https://engsoftmoderna.info/cap3.html>.

Para esse projeto foi identificado e documentado os principais requisitos Funcionais e Não-Funcionais:

- **Requisitos Funcionais**

Código	Requisito	Descrição	Ator

RF-001	Cadastramento de funcionários	Deverá permitir o usuário a realizar o cadastramento de funcionários, as ações que estarão disponíveis serão: criar, remover, alterar e consultar o cadastro de funcionários.	Apenas o Gerente (administrador)
RF-002	Cadastramento de clientes	Deverá permitir o usuário realize o cadastramento de clientes, separando os clientes que tem plano mensal e os que vão apenas utilizar o serviço sem plano mensal, as ações que estarão disponíveis serão: criar, remover, alterar e consultar o cadastro de clientes.	Funcionário (padrão) e Gerente (administrador)
RF-003	Cadastramento de produtos	Deverá permitir o usuário realize o cadastramento de produtos, as ações que estarão disponíveis serão: criar, remover, alterar e consultar o cadastro de produtos.	Funcionário (padrão) e Gerente (administrador)
RF-004	Pagamento	Deverá permitir o usuário cobre o pagamento do cliente, dando a opção de pagamento (cartão de crédito e débito), em caso de pagamento em dinheiro, informar qual será o valor do troco se necessário.	Funcionário (padrão) e Gerente (administrador)
RF-005	Realizar o controle do pagamento do plano mensal	Deverá permitir o usuário realize o controle do pagamento do plano mensal, vinculado ao cadastro do cliente deve haver uma tag que indica se o pagamento está em dia ou está pendente. As ações que estarão disponíveis serão: alterar e consultar.	Apenas o Gerente (administrador)
RF-006	Realizar agendamento de console	Deverá permitir o usuário realize o agendamento do uso do console, as ações que estarão disponíveis serão: criar, remover, alterar e consultar reservas. Cada reserva, deverá ter um cliente e um console em respectivo período.	Funcionário (padrão) e Gerente (administrador)
RF-007	Realizar o monitoramento do uso dos consoles	Deverá permitir o usuário realize o monitoramento do uso dos consoles (tempo de uso).	Funcionário (padrão) e Gerente (administrador)

RF-008	Realizar a locação de jogos	Deverá permitir o usuário realize a locação de jogos disponíveis, com a regra de que clientes com plano mensal podem alugar até 3 jogos por vez, clientes sem plano podem alugar 1 jogo por vez, ou seja, deve estar vinculado com o cadastro do cliente. As ações que estarão disponíveis serão: criar, remover, alterar e consultar.	Funcionário (padrão) e Gerente (administrador)
RF-009	Venda de produtos	Deverá permitir o usuário efetue vendas de produtos: criar, remover, alterar e consultar pedidos.	Funcionário (padrão) e Gerente (administrador)

Tabela 2 - Requisitos Funcionais [Autora]

- **Requisitos Não-Funcionais**

Código	Requisito	Descrição
RNF-001	Restrições de Hardware	Requisito mínimo de hardware: processador intel i3, amd ryzen 3(ou sucessores) baseado em x64, memória ram 8GB, com placa de rede.
RNF-002	Restrições de software	O software do cliente, deverá executar no browser, com servidor de banco de dados (SQL-Server) na linguagem C#.
RNF-003	Usabilidade	Facilidade de navegação.
RNF-004	Restrição de acesso	Ter mecanismos de controle de acesso para apenas pessoas autorizadas terem acesso

Tabela 3 - Requisitos Não- Funcionais [Autora]

Assim, os seguintes recursos deverão ser implementados:

Area administrativa da aplicação: área de acesso restrito, onde os usuários precisarão realizar o processo de autenticação para ter o devido acesso. Além do mecanismo de login, dentro da área administrativa serão implementados todos os cadastros (Funcionários, usuários, clientes, produtos, agendamentos e planos de clientes);

Gerenciamento de Funcionários: acoplada a área administrativa será criado uma subárea para CRUD (Create, Read, Update e Delete) de novos funcionários que poderá ser acessado apenas pelo administrador do sistema;

Gerenciamento de usuários: também acoplada a área administrativa, será criada uma subárea para CRUD de usuários que poderá ser acessado apenas pelo administrador do sistema, limitando as ações do usuário no sistema;

Gerenciamento de produtos: acoplada a área administrativa será criado uma subárea para CRUD de produtos, onde o administrador e o funcionário poderão gerenciar o estoque de produtos.

Gerenciamento de Agendamentos: acoplada a área administrativa será criado uma subárea para CRUD de Agendamentos, onde o administrador e o funcionário poderão gerenciar o agendamento de locação de jogos e de consoles.

Gerenciamento de plano de clientes: acoplada a área administrativa será criado uma subárea para CRUD de plano de cliente.

5.2 Diagramas de caso

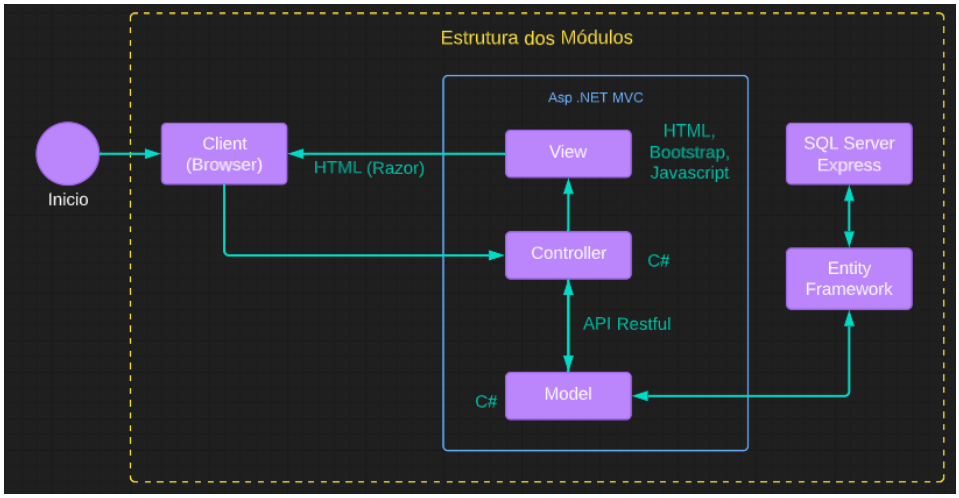


Figura 5 – Estrutura dos Módulos [Autora]

- **Cadastros**

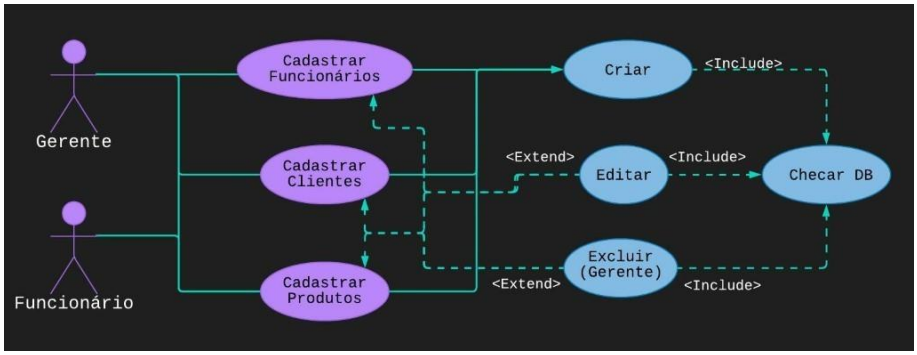


Diagrama 2 - Diagrama de caso: Cadastros [Autora]

- **Alugar jogos**

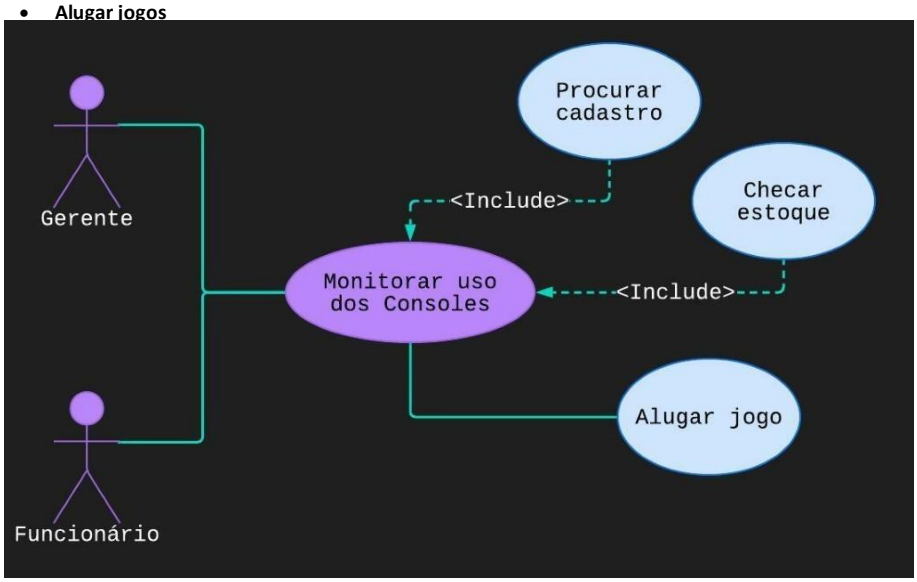


Diagrama 3 - Diagrama de caso: Locação de Jogos [Autora]

- **Monitoramento de consoles**

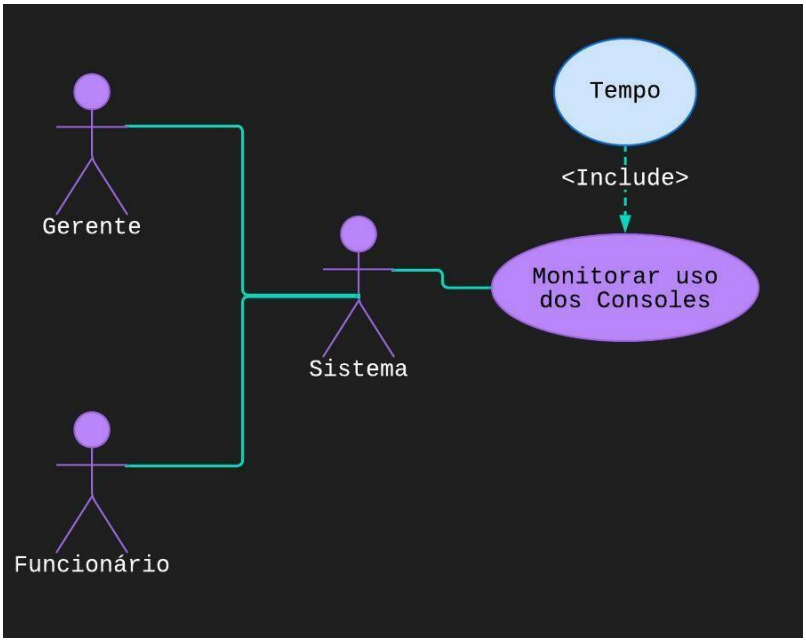


Diagrama 4 - Diagrama de caso: Monitoramento do Consoles [Autora]

- **Pagamentos**

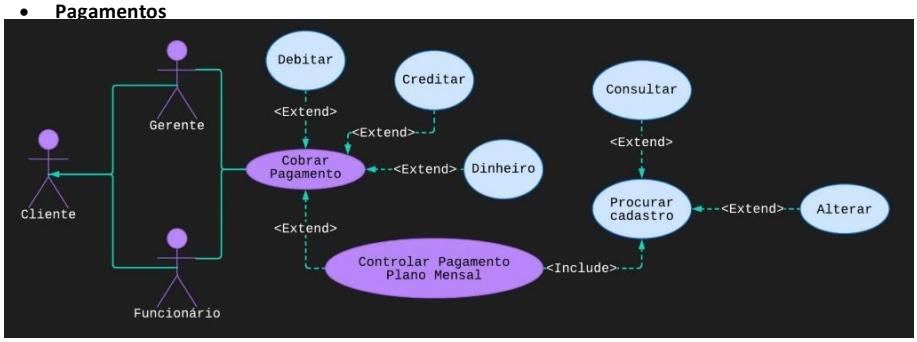


Diagrama 5 - Diagrama de caso: Pagamentos [Autora]

- **Venda de produtos**

5.3 Diagrama de sequência

5.4 Diagrama de classe

5.5 Diagrama entidade relacional

Comentado [mf27]: Pesquisar

Segundo Sanchez e Althmann (2013), o Diagrama Entidade-Relacionamento (ou simplesmente DER) é uma metodologia que permite criar, e posteriormente exibir, de forma

Comentado [mf28]: SANCHEZ, F.; MÁRCIO FÁBIO ALTHMANN. **Desenvolvimento web com ASP.NET MVC**. [s.l.] Editora Casa do Código, 2013.

gráfica e simplificada uma estrutura mais complexa de regras e agrupamento de dados em uma aplicação.

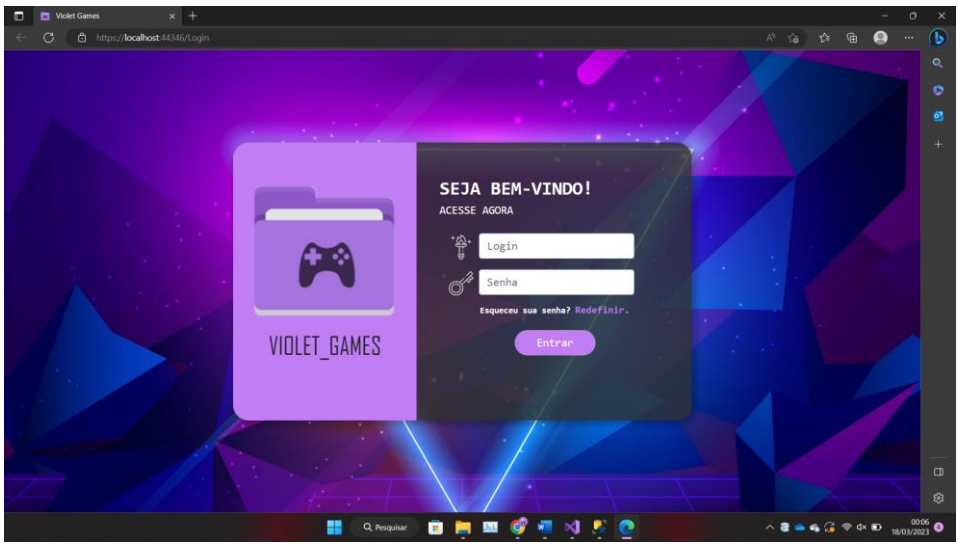
Através da representação gráfica clara possibilitada pelos DER's, é possível compreender a lógica de dados e, até mesmo, algum comportamento da aplicação. Além disso, os diagramas expressam os relacionamentos entre as entidades importantes para o sistema, o que facilita o entendimento da situação problema e seguinte implementação, tanto para desenvolvedores quanto para administradores de bancos de dados. Trata-se do "mapa" criado pelos analistas de requisitos e passado para os desenvolvedores, arquitetos de software e administradores de bancos de dados.

6 MANUAL DO SISTEMA

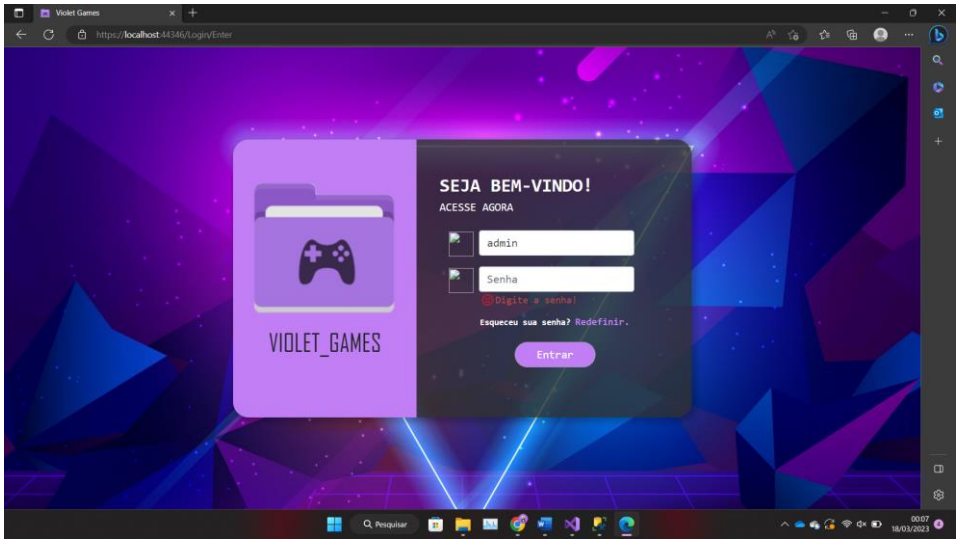
Comentado [mf29]: Colocar print da tela com fonte: próprio autor

6.1 Tela de login

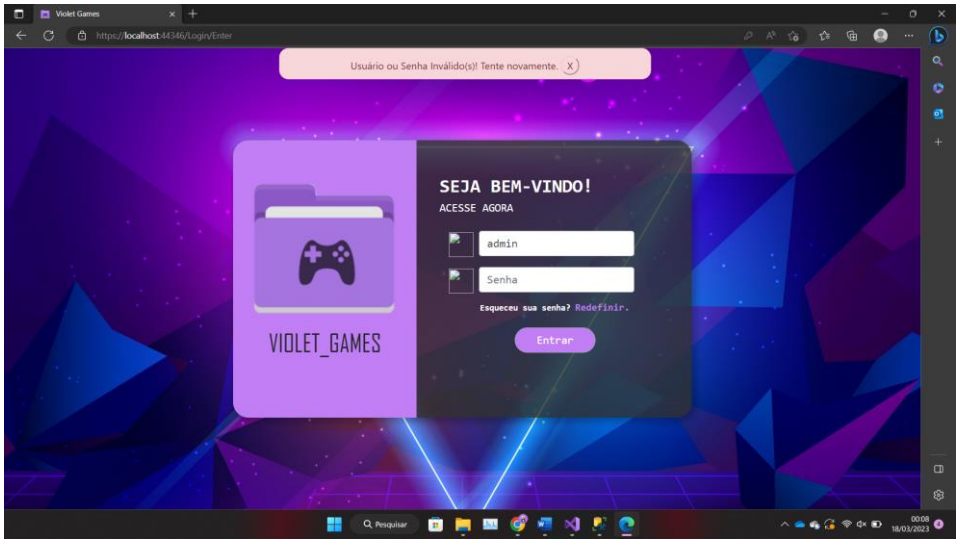
Comentado [mf30]: Colocar o print da tela do usuário standard e da tela de admin, situando a diferencia.



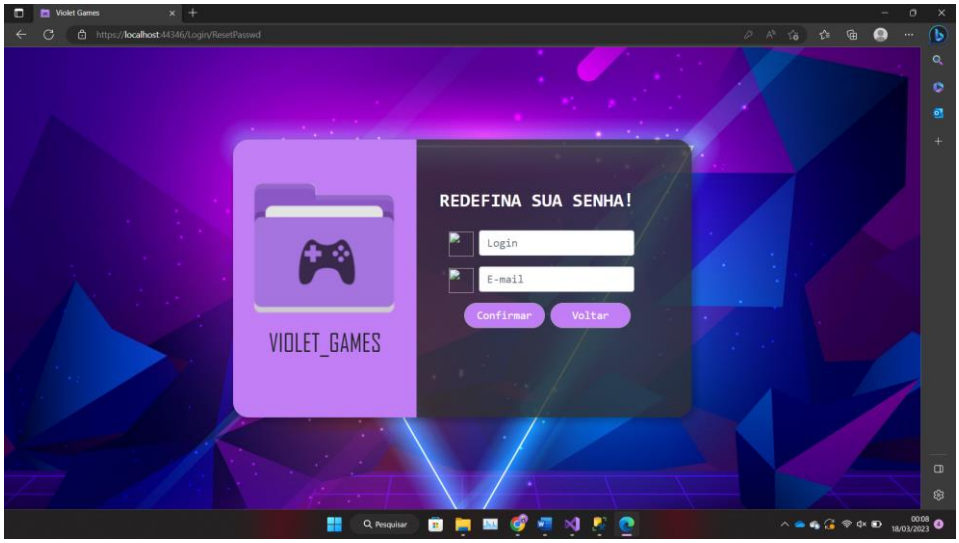
Tela 1 - Tela de Login [Autora]



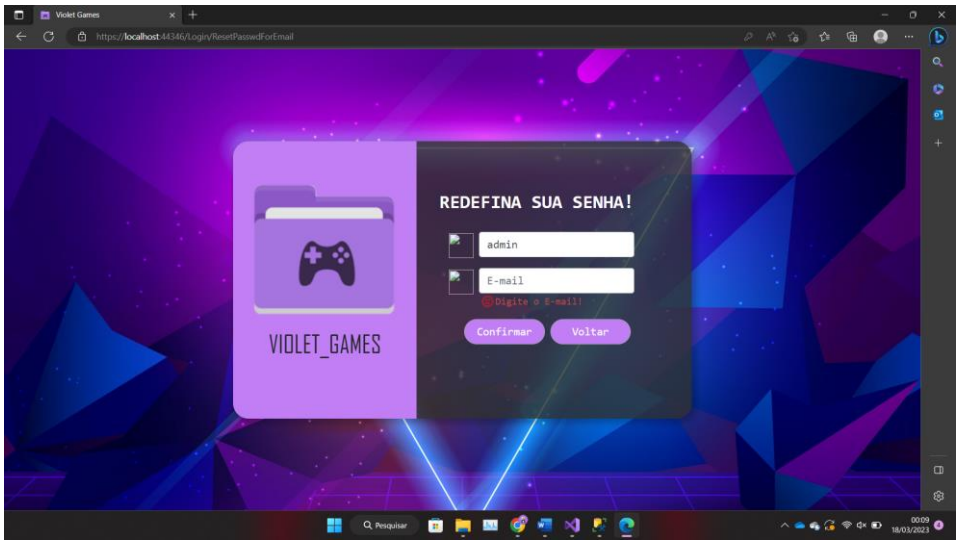
Tela 2 - Tela de Login: Validação de Senha [Autora]



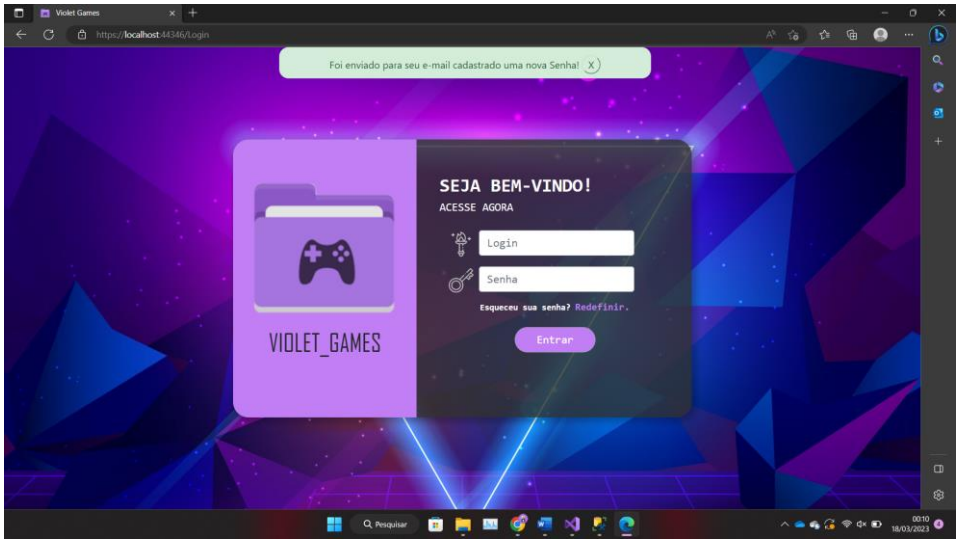
Tela 3 - Tela de Login: Mensagem de Erro do Login [Autora]



Tela 4 - Tela de Login: Tela de redefinir senha [Autora]



Tela 5 - Tela de Login: Validação do e-mail [Autora]



Tela 6 - Tela de Login: Confirmação do envio do reset de senha por e-mail [Autora]

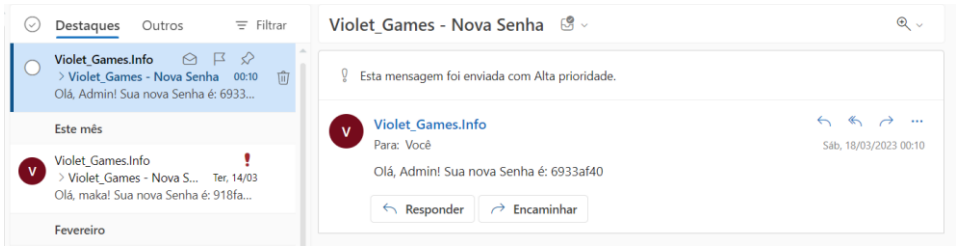


Figura 6 - Reset de senha recebido por e-mail [Autora]

6.2 Tela do dashboard
6.3 Tela do caixa
6.4 Tela de produtos
6.5 Tela de consoles
6.6 Tela de clientes
6.7 Tela de funcionários

7 CONCLUSÃO

8 REFERÊNCIAS

DIAS, Felipe; Morgado, Gisele; Oscar, Pedro; Silveira, Denis; Alencar, Antonio J.; Lima, Priscila; Schmitz, Eber. **Uma Abordagem para a Transformação Automática do Modelo de Negócio em Modelo de Requisitos.** NCE – IM – Universidade Federal do Rio de

Janeiro. Publicado em 28 feb. 2014. Disponível em: <https://www.researchgate.net/profile/Antonio-Alencar/publication/221235083_Uma_Abordagem_para_a_Transformacao_Automatica_do_Modelo_de_Negocio_em_Modelo_de_Requisitos/links/0c960531014620a3a8000000/Uma-Abordagem-para-a-Transformacao-Automatica-do-Modelo-de-Negocio-em-Modelo-de-Requisitos.pdf>. Acesso em: 18 mar. 2023.

MICROSOFT. **Introdução ao .NET Framework - .NET Framework**. Publicado em 10 mar. 2023. Disponível em: <https://docs.microsoft.com/pt-br/dotnet/framework/get-started/>. Acesso em: 18 mar. 2023.

MICROSOFT. **O histórico da linguagem C# – Guia do C#**. Publicado em 8 mar. 2023. Disponível em: <https://docs.microsoft.com/pt-br/dotnet/csharp/whats-new/csharp-version-history>. Acesso em: 18 mar. 2023.

TIME.GRAPHICS. **Microsoft .NET History - Linha do tempo**. Disponível em: <https://time.graphics/pt/line/291016>. Acesso em: 30 mar. 2023.

DEVMEDIA. **A evolução da linguagem de programação C#**. Publicada em: 2013. Disponível em: <https://www.devmedia.com.br/a-evolucao-da-linguagem-de-programacao-csharp/28639>. Acesso em: 26 mar. 2023.

W3SCHOOLS. **SQL History**. Disponível em: <https://www.w3schools.in/sql/history>. Acesso em: 23 mar. 2023.

WIKIPÉDIA. **Microsoft SQL Server**. 24 feb 2023. Disponível em: <https://pt.wikipedia.org/wiki/Microsoft_SQL_Server>. Acesso em: 28 feb. 2023.

ORACLE. **O que é um banco de dados relacional?** Disponível em: <https://www.oracle.com/br/database/what-is-a-relational-database/>. Acesso em: 23 feb. 2023.

SILVA, Leonel Sanches. **c# - 3 camadas vs MVC**. Publicado em 21 set. 2014. Disponível em: <https://pt.stackoverflow.com/questions/33352/3-camadas-vs-mvc>. Acesso em: 20 jan. 2023.

BUSS, Liana; Punchirolli, Lucas C.; Olivo, Ricardo R. **Análise das vulnerabilidades encontradas em técnicas de programação PHP para tratamento de queries SQL**. Universidade do Estado de Santa Catarina - Centro de Ciência e Tecnologia – CCT. Joinville, 2013.

Comentado [mf31]: Colocar em ABNT

ARAÚJO, Everton Coimbra. **ASP.NET Core MVC: Aplicações modernas em conjunto com o Entity Framework.** São Paulo: Casa do Código, 2018.

SANCHEZ, F.; MÁRCIO FÁBIO ALTHMANN. **Desenvolvimento web com ASP.NET MVC**. [s.l.] Editora Casa do Código, 2013.

FOWLER, Martin. **Padrões de Arquitetura de Aplicações Corporativas**. Bookman, São Paulo, 9 jul. 2018.

BURBECK, Steve. **Applications Programming in Smalltalk-80(TM): How to use Model-View-Controller (MVC)**. Disponível em: <https://folk.universitetetioslo.no/trygver/themes/mvc/mvc-index.html>. Acesso em: 10 mar. 2022.

VALENTE, M. T. **Cap. 3: Requisitos – Engenharia de Software Moderna**. Publicado em 2020. Disponível em: <https://engsoftmoderna.info/cap3.html>. Acesso em: 10 mar. 2022.

MICROSOFT. **Visão geral do Entity Framework Core – EF Core**. Publicado em 28 set. 2022. Disponível em: <https://learn.microsoft.com/pt-br/ef/core/>. Acesso em: 10 mar. 2022.

O que é arquitetura de três camadas (tiers). Disponível em: <https://www.ibm.com/br-pt/cloud/learn/three-tier-architecture#toc-outras-arq-uMx8DOIM>. Acesso em: 20 mar. 2022.

FREEMAN, Eric; FREEMAN, Elisabeth. **Use a cabeça! padrões de projeto: Design Patterns**. 2. ed. São Paulo: Alta Books, 2007.

Comentado [mf32]: Verificar

Comentado [mf33]: Verificar

Comentado [mf34]: Verificar