**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

**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.

“Uma pessoa que nunca cometeu um erro nunca tentou nada de novo”

Albert Einstein

**RESUMO**

O presente trabalho

**Palavras-chave:** Informação, Logística, Sistema, Game House.

**ABSTRACT**

**Keywords:** Information, Logistics, System, Game House.

**Diagramas**

[Figura 1 - Diagrama do modelo MVC 12](#_Toc117688072)

[Figura 2 - Diagrama de caso: Cadastros 15](#_Toc117688073)

[Figura 3 - Diagrama de caso: Locação de Jogos 16](#_Toc117688074)

[Figura 4 - Diagrama de caso: Monitoramento do Consoles 16](#_Toc117688075)

[Figura 5 - Diagrama de caso: Pagamentos 17](#_Toc117688076)

**Tabelas**

Tabela 1 - Três camadas principais 14

Sumário

[1. INTRODUÇÃO 12](#_Toc127207948)

[1.1 Problema 12](#_Toc127207949)

[1.2 Justificativa 12](#_Toc127207950)

[1.3 Hipótese 13](#_Toc127207951)

[1.4 Objetivo 13](#_Toc127207952)

[1.5 Metodologia 13](#_Toc127207953)

[2. CONCEITUANDO UMA GAME HOUSE 13](#_Toc127207954)

[Modelo de negócio 13](#_Toc127207955)

[3. TECNOLOGIAS UTILIZADAS NO DESENVOLVIMENTO 14](#_Toc127207956)

[3.1 C#, .NET e ASP.NET 14](#_Toc127207957)

[3.2 Entity 14](#_Toc127207958)

[3.3 HTML, CSS e JavaScript 14](#_Toc127207959)

[3.4 Bootstrap 14](#_Toc127207960)

[4. ARQUITETURA MODEL-VIEW-CONTROLLER (MVC) 14](#_Toc127207961)

[5. DESENVOLVIMENTO 16](#_Toc127207962)

[5.1 Requisitos do sistema 16](#_Toc127207963)

[5.2 Diagramas de Caso 18](#_Toc127207964)

[5.3 Diagrama de Sequência 21](#_Toc127207965)

[5.4 Diagrama de Classe 21](#_Toc127207966)

[5.5 Modelo Entidade Relacional 21](#_Toc127207967)

[6. MANUAL DO SISTEMA 21](#_Toc127207968)

[6.1 Tela de Login 21](#_Toc127207969)

[6.2 Tela do Dashboard 21](#_Toc127207970)

[6.3 Tela do Caixa 21](#_Toc127207971)

[6.4 Tela de Produtos 21](#_Toc127207972)

[6.5 Tela de Consoles 21](#_Toc127207973)

[6.6 Tela de Clientes 21](#_Toc127207974)

[6.7 Tela de Funcionários 21](#_Toc127207975)

[6.8 Tela de Configuração 21](#_Toc127207976)

[7. CONCLUSÃO 21](#_Toc127207977)

[8. REFERÊNCIAS 21](#_Toc127207978)

# 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 sistemas para auxiliar no gerenciamento e tomada de decisão.

Considerando sua grande relevância que o sistema tem para uma empresa, trazendo uma dependência cada vez maior para o fluxo de serviços aplicados pelas entidades empresariais.

## Problema

Com a crescente onda de jogos na era digital, muitas pessoas começaram a comprar jogos e consoles, porém muitos se arrependem de suas escolhas. Visto isso, foi criado uma adaptação das antigas Locadoras, chamado Game House, onde as pessoas podem ir para alugarem jogos para levarem para casa ou para jogar no local utilizando os consoles da loja.

## Justificativa

Para que o gerenciamento da Game House seja facilitado, a criação de um sistema foi proposto, evidentemente, para funcionar deve-se ser cadastrados no sistema os funcionários e clientes e em função disso, foi feito um planejamento após análises de requisitos necessários para o desenvolvimento.

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 area 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 (Create, Read, Update e Delete) 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 (Create, Read, Update e Delete) 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 (Create, Read, Update e Delete) de plano de cliente.

## 1.3 Hipótese

## 1.4 Objetivo

## 1.5 Metodologia

# CONCEITUANDO UMA GAME HOUSE

## Modelo de negócio

Como objeto de estudo e prototipagem, foi escolhido utilizar um projeto acadêmico feito durante as aulas de ministradas pelo professor que leciona a matéria de Engenharia de Software. O professor propôs aos alunos escolherem um empreendimento fictício, após a escolha deviam criar um pequeno modelo de negócio e a partir do modelo do empreendimento deviam criar alguns requisitos de sistema funcionais e não-funcionais.

**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 (OBS: pode ser jogado na plataforma escolhida um jogo por vez), 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:** 1 Xbox 360, 1 Xbox One, 1 PS1, 1 PS2, 1 PS3, 1 PS4, 1 PS5, 3 PC’s gamer, 1 Óculos VR para PS4, 1 Nitendo Switch.

**Atividades-chave:** agendamento do uso da sala de jogos, locação dos jogos, gestão de compra de novos jogos/aparelhos, monitoramento do uso consoles/jogos, tirar dúvidas dos clientes.

**Fornecedores:** SND Distribuição (vende jogos e itens de informática), ShopB (além de comercializar videogames e acessórios para games, disponibiliza alguns cursos para quem está mergulhando com tudo nesse universo), MH Games (Loja que fornece preços diferenciados para clientes do varejo e do atacado games)

**Estrutura de custo:** energia(R$500,00/mês), água(R$50,00/mês), internet e telefone(R$140,00), salários(R$2400,00), aluguel(R$1000,00).

# 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 Core e como linguagem o C#.

Conforme o cenário web transformou-se rápida e largamente nos anos posteriores (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 e que pudesse endereçar os problemas mencionados anteriormente. Surgiu logo, o ASP.NET MVC. Desenvolvimento web com ASP.NET MVC (Fabrício Sanchez, Márcio Fábio Althmann) (z-lib.org)

## 3.2 Entity

## 3.3 HTML, CSS e JavaScript

## 3.4 Bootstrap

# ARQUITETURA MODEL-VIEW-CONTROLLER (MVC)

Na fase de projeto, o tópico fundamental é a escolha e o desenho da arquitetura da aplicação, Martin Fowler 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).

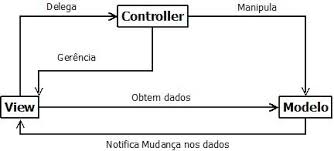


Figura - Diagrama do modelo MVC

Conceito do padrão MVC apresentado no livro “ASP.NET Core MVC: Aplicações modernas em conjunto com o Entity Framework”:

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

Segundo [Brown *et al*.], 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.

Tabela

Descrição gerada automaticamente

Tabela - Três camadas principais

Segundo Leonel Sanches da Silva (Arquiteto de Soluções da Twilio), para entender a diferença, é necessário entender do ponto de vista de um Model. Um Model é uma classe que define 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 3 Camadas. Na arquitetura de 3 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.

# 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. De <<https://engsoftmoderna.info/cap3.html>>

Para esse projeto foram identificados e documentados os principais requisitos Funcionais e Não-Funcionais:

* **Requisitos Funcionais**

|  |  |  |  |
| --- | --- | --- | --- |
| **Código** | **Nome Requisito** | **Descrição** | **Ator** |
| **RF-001** | **Cadastramento de funcionários** | **Esta funcionalidade, deverá permitir o usuário (gerente) a realizar o cadastramento de funcionários, as ações que estarão disponíveis serão: criar, remover, alterar e consultar o cadastro de funcionários.** | **Gerente** |
| **RF-002** | **Cadastramento de clientes** | **Esta funcionalidade, deverá permitir o usuário (funcionário) a realizar 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**  **Gerente** |
| **RF-003** | **Cadastramento de produtos** | **Esta funcionalidade, deverá permitir o usuário (funcionário) a realizar o cadastramento de produtos, as ações que estarão disponíveis serão: criar, remover, alterar e consultar o cadastro de produtos.** | **Funcionário**  **Gerente** |
| **RF-004** | **Pagamento** | **Esta funcionalidade, deverá permitir o usuário (funcionário) cobrar 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**  **Gerente** |
| **RF-005** | **Realizar o controle do pagamento do plano mensal** | **Esta funcionalidade, deverá permitir o usuário (funcionário) realizar 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.** | **Funcionário**  **(apenas consultar)**  **Gerente (consultar e alterar)** |
| **RF-006** | **Realizar agendamento de console** | **Esta funcionalidade, deverá permitir o usuário (funcionário) a realizar 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**  **Gerente** |
| **RF-007** | **Realizar o monitoramento do uso dos consoles** | **Esta funcionalidade, deverá permitir o usuário (funcionário) a realizar o monitoramento do uso dos consoles (tempo de uso).** | **Funcionário**  **Gerente** |
| **RF-008** | **Realizar a locação de jogos** | **Esta funcionalidade, deverá permitir o usuário (funcionário) a realizar 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**  **Gerente** |
| **RF-009** | **Venda de produtos** | **Esta funcionalidade, deverá permitir o usuário (funcionário) a efetuar vendas de produtos: criar, remover, alterar e consultar pedidos.** | **Funcionário**  **Gerente** |

* **Requisitos Não-Funcionais**

|  |  |  |
| --- | --- | --- |
| **Código** | **Nome 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** |

## 5.2 Diagramas de Caso

Diagrama Entidade-Relacionamento (ou simplesmente DER) é uma metodologia que permite criar, e posteriormente exibir, de forma gráfica e simplificada—nível de abstração mais alto—uma estrutura mais complexa de regras e agrupamento de dados em uma aplicação.(Referencia) Desenvolvimento web com ASP.NET MVC (Fabrício Sanchez, Márcio Fábio Althmann) (z-lib.org)  
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.

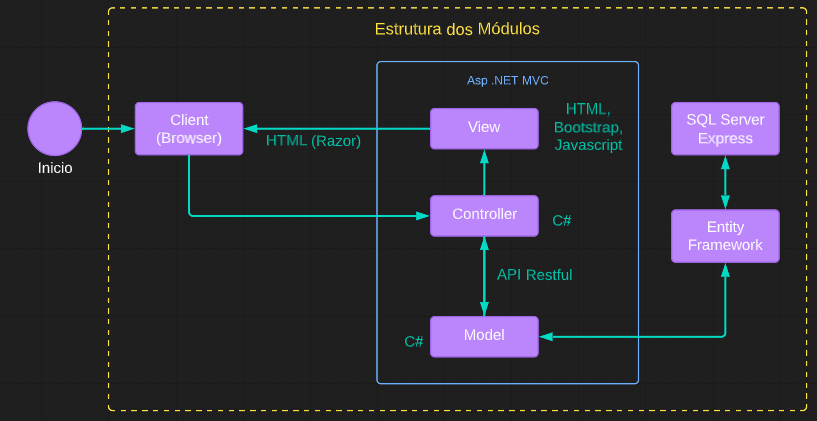


Figura – Estrutura dos Módulos

* **Cadastros**

Diagrama, Esquemático

Descrição gerada automaticamente

Figura - Diagrama de caso: Cadastros

* **Alugar jogos**

Diagrama

Descrição gerada automaticamente

Figura - Diagrama de caso: Locação de Jogos

* **Monitoramento de consoles**

Diagrama

Descrição gerada automaticamente

Figura - Diagrama de caso: Monitoramento do Consoles

* **Pagamentos**

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Figura - Diagrama de caso: Pagamentos

* **Venda de produtos**

## 5.3 Diagrama de Sequência

## 5.4 Diagrama de Classe

## 5.5 Modelo Entidade Relacional

# MANUAL DO SISTEMA

## 6.1 Tela de Login

## 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

## 6.8 Tela de Configuração

# CONCLUSÃO

# REFERÊNCIAS

**O que é arquitetura de três camadas (tiers)**. Disponível em: <https://www.ibm.com/br-pt/cloud/learn/three-tier-architecture#toc-outras-arq-uMx8DOlM>. Acesso em: 10 mar. 2022.

[Brown *et al*.] Brown et al. *Enterprise Java Programming with IBM Websphere*. Addison-Wesley, 2001.

FOWLER, Martin. (2006) Padrões de Arquitetura de Aplicações Corporativas. Bookman, São Paulo.

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

AJCVICKERS. **Visão geral do Entity Framework Core – EF Core**. Disponível em: <https://learn.microsoft.com/pt-br/ef/core/>.

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

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