**PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS**NÚCLEO DE EDUCAÇÃO A DISTÂNCIA

Pós-graduação *Lato Sensu* em Desenvolvimento Web Full Stack

**Adjamir de Moura Galvão**

**SISTEMA PARA CONTROLE DE LOJAS DE PEQUENO PORTE**

Belo Horizonte

2023

**SUMÁRIO**

1. Apresentação 3

1.1. Contexto 3

1.2. Público alvo 4

1.3. Requisitos 4

2. Modelagem 6

2.1. Diagrama de casos de uso 6

2.2. Atores 6

2.3. Detalhamento dos casos de uso 6

2.4. Projeto de Interface 6

2.5. Diagrama de classes 6

3. Projeto 7

3.1. Arquitetura de *software* 7

3.1.1. Definição do Framework de Trabalho 7

3.2. Arquitetura da informação 9

4. Testes 9

5. URLs 9

5.1. Aplicação web 10

5.2. Repositório código-fonte 10

5.3. Vídeo de apresentação do trabalho 10

REFERÊNCIAS 11

# 1. Apresentação

## 1.1. Contexto

Pequenas lojas/comércios precisam de uma infraestrutura que dê apoio ao controle de estoque e custos dos produtos. Além disso, precisam fazer o registro de suas vendas. Estas informações facilitam a formação do preço de venda dos produtos, otimização das compras e, consequentemente, a lucratividade da loja.

O atendimento aos clientes ocorre na loja física e o registro de vendas pode ser feita simultaneamente pelo caixa tradicional e por vendedores. Desta forma é possível evitar a formação de filas para o pagamento, agilizando o atendimento.

A figura abaixo apresenta o *Business Model Canvas* do negócio.

**Figura 1 - *Business Model Canvas***



.

## 1.2. Público alvo

Neste cenário de negócio identificamos os seguintes perfis para o público alvo:

* Dono do negócio: Em pequenos negócios muitas vezes os donos do negócio não possuem muito conhecimento sobre a formação de preços dos seus produtos e sobre o processo de controle de estoque. Desta forma eles tem dificuldades em avaliar quais os períodos mais propícios para reposição do estoque e definir os preços dos produtos de forma equilibrar a rentabilidade daquele produto e do negócio como um todo.
* Gestor do estoque: É responsável pela compra dos produtos e está subordinado do dono do negócio. Muitas vezes esta atribuição é do próprio dono do negócio.
* Vendedores/Caixa: São os responsáveis pelas vendas de produtos e identificação dos clientes. Eles estão subordinados ao dono da loja, mas necessitam ter flexibilidade na negociação das vendas para melhor atender aos clientes. Embora não façam o controle de estoque, precisam destas informações para evitar vendas de produtos sem estoque disponível. São usuários que precisam de agilidade no registro de vendas para um melhor atendimento aos clientes.
* Clientes: Normalmente não possuem muita relação com o negócio, mas eventualmente precisam de informações sobre compras passadas para comprovação da compra ou até mesmo efetuar devoluções.

## 1.3. Requisitos

Requisitos Funcionais:

* Gerenciamento de fornecedores: o sistema deve permitir o cadastro, atualização e exclusão de fornecedores.
* Gerenciamento de vendedores: o sistema deve permitir o cadastro, atualização e exclusão de vendedores.
* Gerenciamento de clientes: o sistema deve permitir o cadastro, atualização e exclusão de clientes.
* Gerenciamento de produtos e estoque: o sistema deve permitir o cadastro, atualização e exclusão de produtos, bem como o controle do estoque e a atualização automática do estoque quando um produto é vendido.
* Gerenciamento de vendas: o sistema deve permitir a realização de vendas podendo identificar ou não vendedores e clientes. O vendedor deve ser identificado automaticamente ao acessar o sistema, mas deve permitir a escolha de outro. Além disso, o sistema deve permitir ao vendedor oferecer desconto nos produtos vendidos.
* Auxílio a Gestão: o sistema deve exibir indicadores sobre vendas e compras. Além disso deve poder identificar os produtos mais vendidos e, podendo fazer essa análise por cliente.
* Controle de acesso: o sistema deve permitir o controle de acesso aos usuários, definindo diferentes perfis de acesso para cada tipo de usuário.

Requisitos Não Funcionais:

* Usabilidade: o sistema deve apresentar uma interface flexível para que possa ser utilizado em ambientes desktop e mobile simultaneamente.
* Desempenho: o sistema deve ser rápido e responsivo, com tempos de resposta baixos.
* Confiabilidade: o sistema deve ser confiável e estar sempre disponível, minimizando a possibilidade de erros e falhas.
* Segurança: o sistema deve ter política de controle de acesso, protegendo-se de acessos não autorizados.
* Escalabilidade: o sistema deve ser capaz de crescer com o negócio, suportando um número crescente de produtos, vendas e usuários.

# 2. Modelagem

## 2.1. Diagrama de casos de uso

O diagrama de casos de uso oferece uma visão global dos casos de uso e dos atores que dele participam.

## 2.2. Atores

Liste cada um dos atores que participarão dos casos de uso, oferecendo uma descrição sucinta para cada um deles. Os atores são todos aqueles que interagem com o sistema (usuários, outros sistemas, etc.). Agora, como já estamos falando da modelagem do sistema, já podemos incluir os usuários responsáveis pela administração do sistema

## 2.3. Detalhamento dos casos de uso

Nesta seção, cada caso de uso deve ser detalhado. Esse detalhamento deve incluir uma descrição do caso de uso, a lista de atores que participam do caso de uso, as pré e pós-condições e os fluxos de eventos (básico, alternativo, de exceção, sub-fluxos, etc.).

## 2.4. Projeto de Interface

Apresente o fluxo do usuário no sistema e o layout das interfaces, incluindo uma descrição dos campos e comandos em cada uma delas. Como forma de layout, use um wireframes ou as interfaces já implementadas.

Se necessário, use um diagrama de estados para representar o comportamento de uma interface ou fluxo de um processo associado à funcionalidade em questão.

## 2.5. Diagrama de classes

Apresente o diagrama de classes da aplicação web, descrevendo, sucintamente, as classes e as relações entre elas.

# 3. Projeto

## 3.1. Arquitetura de *software – Projeto API First*

O sistema é composto pelo *front end* (seção 3.1.1.1. Framework Front End) e *back end* (seção 3.1.1.2. Framework Back End).

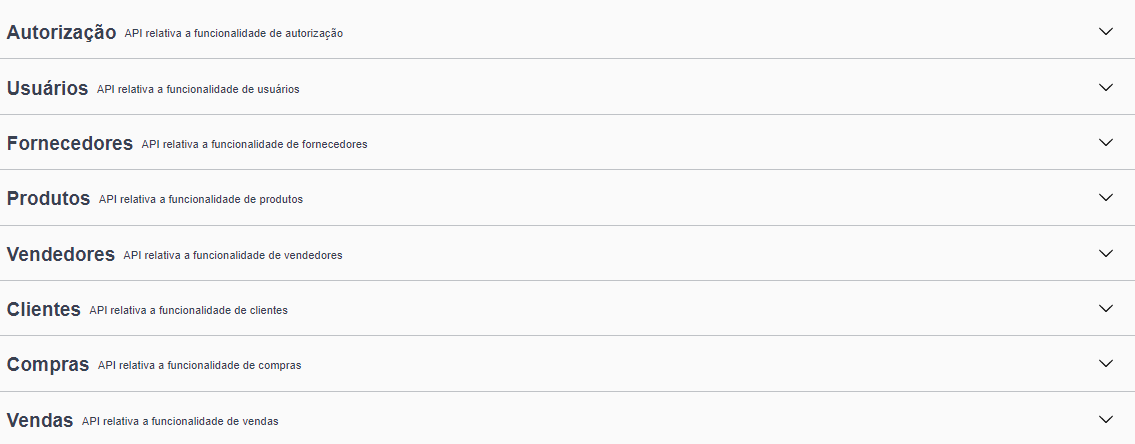
O *back end* é uma API Rest. Sua documentação foi feita via *Swagger* com base na especificação *OpenAPI*. Ela pode se encontra disponível em <https://adjamir2318.c41.integrator.host/api-docs/>.

As figuras abaixo ilustram as principais seções da especificação.

Figura 3 - *Schema* de Dados da API



Figura 2 - Grupos de Funcionalidades da API



### 3.1.1. Definição do Framework de Trabalho

A escolha do framework de trabalho para este projeto levou em consideração os seguintes aspectos:

1. Aderência aos requisitos do projeto: O framework deve atender aos requisitos funcionais e não funcionais do projeto.
2. Aplicabilidade no ambiente de trabalho do aluno: Tendo em vista que trata-se de uma pós-graduação patrocinada e que a instituição possui política própria de escolha de frameworks, o framework utilizado deve ter aplicabilidade no trabalho do aluno.
3. Aprendizado de novas tecnologias: Além de considerar os critérios anteriores, o framework adotado deve servir como estudo de caso para a proposição de novas soluções para a entidade patrocinadora.

#### 3.1.1.1. Framework Front End

Para o *Front End* foram adotadas as seguintes tecnologias:

* Framework: *Angular* (versão 15).
* *Bootstrap* (via ng-bootstrap)
* *Material Design* (via Angular Material).

O framework *Angular* é mantido e adotado pelo Google, possuindo uma grande comunidade de usuários. Aplicações desenvolvidas neste framework funcionam na grande maioria os browsers modernos.

O *Bootstrap* fornece estruturas de *Cascading Style Sheets* (CSS) para a criação de sites e aplicações responsivas. Ele permite tratar sites para ambiente desktop e mobile da mesma forma. Para o projeto, foi adotado o módulo *ng-bootstrap* que é uma implementação do *bootstrap* para *Angular*.

Uma vez definido o framework, é importante a escolha de uma biblioteca de componentes para acelerar o desenvolvimento da aplicação. Considerando que a adoção da biblioteca de componentes *Angular* da instituição patrocinadora do aluno não poderia ser utilizada devido as regras de sigilo e propriedade intelectual, optou-se pela adoção do *Material Design* através da biblioteca *Material Angular*.

*Material Design* é a especificação de design para interfaces interativas do Google. Já o *Angular Material* é a implementação oficial, para Angular, do Material Design.

A escolha dessa arquitetura acelerá a criação de aplicações com alta responsividade para em ambientes desktop e mobile. Além disso, devido a sua grande adoção, permite o acesso a uma grande comunidade de desenvolvedores para compartilhar conhecimento.

#### 3.1.1.2. Framework Back End

Para o *Back End* foi adotada a seguinte tecnologia:

* *Node.js* com *Express.js*

O *Node.js* é uma plataforma de software de código aberto construída com a linguagem de programação *JavaScript*. Ele permite que os desenvolvedores criem aplicativos de rede escaláveis e de alto desempenho usando *JavaScript*. Um ambiente *Node.js* não exige muitos recursos computacionais e por ser baseado em um modelo de E/S não bloqueante e orientado a eventos, torna se ideal para aplicativos que precisam lidar com muitas conexões simultâneas.

O *Express* é um framework web de código aberto para o *Node.js* que fornece um conjunto de métodos HTTP para criar rotas de acesso ao servidor e lidar com solicitações e respostas HTTP, dando suporte para a criação de APIs RESTfull.

Desta forma, a escolha desta arquitetura permitirá a criação de um ambiente de alta disponibilidade, escalável e ao mesmo tempo de baixo custo, uma vez que não possui baixo consumo de recursos computacionais. Além disso, é uma oportunidade de estudo com grandes possibilidades de uso no ambiente de trabalho do aluno.

#### 3.1.1.3. Persistência

Para a persistência foi adotada a seguinte tecnologia:

* *MongoDB*

O *MongoDB* é um banco de dados *NoSQL* (não relacional) de código aberto, que armazena dados em formato de documento *JSON (JavaScript Object Notation).* Ele foi criado para fornecer alta escalabilidade, desempenho e flexibilidade para aplicativos modernos que precisam armazenar e acessar grandes quantidades de dados.

De código aberto e também disponibilizado em conjunto por servidores que hospedam aplicações *Node.js*, o *MongoDB* se torna uma ótima solução para a persistência da aplicação, aliando robustez, escalabilidade e baixo custo.

## 3.2. Arquitetura da informação

Descreva como a informação estará organizada na aplicação – hierarquias, categorias, rótulos (palavras-chave), etc.

Descreva também como será a navegação pelo espaço de navegação, os mecanismos de busca e de recuperação de informações. Opcionalmente inclua um diagrama hierárquico que mostre como as funcionalidades estão distribuídas pelo sistema.

*(https://brasil.uxdesign.cc/qual-a-diferen%C3%A7a-entre-arquitetura-de-informa%C3%A7%C3%A3o-ai-user-flow-e-site-map-b9d6c7461dee)*

# 4. Testes

Apresente os planos de testes (testes de software, testes heurísticos ou testes de usabilidade) além dos registros de testes realizados em sua aplicação Web.

# 5. URLs

## 5.1. Aplicação web

A aplicação web encontra-se hospedada em <https://adjamir2318.c41.integrator.host/>.

Para acessar a aplicação com o perfil de administrador deve-se utilizar o login “admin” e a senha “@admin!”.

## 5.2. Repositório código-fonte

O código fonte da aplicação encontra-se em <https://github.com/adjamirgalvao/pucminas>.

## 5.3. Vídeo de apresentação do trabalho

Inclua o link para o vídeo com a apresentação da sua aplicação. Seu vídeo deve ter duração máxima de 5 minutos e deve apresentar de forma sucinta o seu projeto. Concentre-se principalmente na apresentação da sua aplicação, mostrando o funcionamento de cada um dos requisitos solicitados na proposta escolhida por você.

O software OBS Studio é uma boa sugestão de software para gravação de screencast.

# REFERÊNCIAS

Como um projeto de aplicativo não requer revisão bibliográfica, a inclusão das referências não é obrigatória. No entanto, caso você deseje incluir referências relacionadas às tecnologias ou às metodologias que foram usadas no seu trabalho, relacione-as de acordo com o modelo a seguir.

SOBRENOME DO AUTOR, Nome do autor. **Título do livro ou artigo.** Cidade: Editora, ano.

SOBRENOME DO AUTOR, Nome do autor. **Título do livro ou artigo.** Cidade: Editora, ano.

SOBRENOME DO AUTOR, Nome do autor. **Título do livro ou artigo.** Cidade: Editora, ano.

SOBRENOME DO AUTOR, Nome do autor. **Título do livro ou artigo.** Cidade: Editora, ano.

SOBRENOME DO AUTOR, Nome do autor. **Título do livro ou artigo.** Cidade: Editora, ano.

SOBRENOME DO AUTOR, Nome do autor. **Título do livro ou artigo.** Cidade: Editora, ano.