

Instituto Federal de Educação, Ciência e Tecnologia de São Paulo
Campus Bragança Paulista

Geisiele de Oliveira – BP3053563

Thiago Oliveira – BP3053636

Vinicius Arantes – BP3053709

Vinícius Magalhães – BP3054365

Loja – VENEZART

Projeto Semestral do 3º módulo do Curso ADS



Análise Orientada a Objetos (BRAAOOB)

Profa. Ana Paula Müller Giancoli

BRAGANÇA PAULISTA

2025

Sumário

1	INTRODUÇÃO	3
1.1	Contextualização	3
1.2	Problema	4
1.3	Justificativa	4
1.4	Objetivos	5
1.4.1	Objetivo geral	5
1.4.2	Objetivos específicos	5
1.5	Metodologia	5
1.6	Organização do trabalho	6
2	DESENVOLVIMENTO	7
2.1	Características da aplicação	7
2.1.1	Atores e seus papéis	7
2.1.2	Classes de usuários e características	7
2.2	Arquitetura física	8
2.2.1	Tecnologias utilizadas	8
2.3	Arquitetura lógica	9
3	CONCLUSÃO	11
	REFERÊNCIAS	12

1 Introdução

O comércio eletrônico tem se consolidado como um dos principais meios de compra e venda de produtos, permitindo que clientes realizem pedidos de forma rápida, prática e sem a necessidade de deslocamento físico. Nesse cenário, lojas especializadas em segmentos específicos, como materiais artísticos, encontram na *web* uma oportunidade para ampliar o alcance de seu público e oferecer uma experiência de compra mais completa e personalizada.

No contexto dos materiais para pintura em tela, muitos artistas, estudantes e entusiastas da área enfrentam dificuldades para encontrar, em um único ambiente, variedade de produtos, informações detalhadas e facilidade no processo de compra. Além disso, lojas físicas muitas vezes não possuem controle informatizado adequado para organização de catálogo, estoque e pedidos, o que pode afetar diretamente a qualidade do atendimento e a eficiência operacional.

Diante desse cenário, surge a necessidade de uma solução que integre a gestão dos produtos e pedidos com uma interface amigável para o cliente, possibilitando tanto uma boa experiência de navegação quanto um controle administrativo eficiente.

1.1 Contextualização

A informatização de processos comerciais tornou-se essencial para empresas que desejam manter competitividade no mercado atual. O avanço das tecnologias *web* permitiu o surgimento de plataformas cada vez mais robustas, capazes de automatizar tarefas que antes eram realizadas de forma manual, como controle de estoque, registro de vendas e relacionamento com clientes.

No setor de materiais artísticos, essa transformação ainda ocorre de forma gradual, especialmente em pequenas e médias lojas, que muitas vezes dependem de controles informais ou planilhas isoladas. Esse cenário dificulta a gestão eficiente dos produtos, a organização dos pedidos e a análise do comportamento dos clientes.

Diante desse contexto, a criação de sistemas específicos para comércio eletrônico em nichos de mercado, como o de materiais para pintura em tela, torna-se uma alternativa estratégica. Aplicações *web* personalizadas permitem maior controle administrativo, melhoria na experiência do usuário e aumento da eficiência operacional, contribuindo tanto para a modernização do comércio quanto para a ampliação do acesso dos consumidores aos produtos desejados.

1.2 Problema

A ausência de um sistema especializado voltado para a venda de materiais artísticos, com foco em pintura em tela, gera dificuldades tanto para os clientes quanto para o administrador da loja. Do ponto de vista do cliente, problemas como falta de informações sobre os produtos, dificuldade de navegação, ausência de histórico de compras e processos pouco intuitivos de finalização de pedido são recorrentes.

Do ponto de vista do administrador, a inexistência de uma ferramenta integrada pode resultar em falhas no controle de estoque, dificuldade de acompanhamento de pedidos, retrabalho no registro de vendas e baixa visibilidade sobre o comportamento de consumo dos clientes.

Dessa forma, o problema que se busca abordar neste trabalho pode ser sintetizado na seguinte questão: *como oferecer uma solução web que facilite a experiência de compra de materiais artísticos pelos clientes e, ao mesmo tempo, disponibilize ao administrador recursos eficientes para o gerenciamento de produtos e pedidos?*

1.3 Justificativa

O desenvolvimento de um sistema *web* voltado para a comercialização de materiais artísticos, especialmente aqueles relacionados à pintura em tela, justifica-se pela crescente demanda por soluções digitais que atendam nichos específicos de mercado e pela necessidade de modernização de processos em pequenas e médias empresas.

Além de contribuir para a organização interna da loja, um sistema como o Venezart proporciona maior comodidade ao cliente, que passa a ter acesso a um catálogo estruturado, com informações detalhadas sobre os produtos, possibilidade de montar seu carrinho de compras, finalizar pedidos de forma segura e registrar o histórico de compras realizado.

Do ponto de vista acadêmico e profissional, o projeto também se mostra relevante por permitir a aplicação de conceitos de análise orientada a objetos, modelagem de sistemas, arquitetura cliente-servidor e desenvolvimento *full stack*, utilizando tecnologias atuais como *Flask* no *back-end* e *React* no *front-end*. Dessa forma, o trabalho contribui tanto para a formação técnica dos desenvolvedores envolvidos quanto para a geração de uma solução prática com potencial de uso real.

1.4 Objetivos

1.4.1 Objetivo geral

Desenvolver uma aplicação *web*, denominada Venezart, para apoiar a venda de materiais artísticos com foco em pintura em tela, oferecendo uma interface intuitiva para o cliente e ferramentas de gerenciamento para o administrador da loja.

1.4.2 Objetivos específicos

Os objetivos específicos deste trabalho incluem:

- realizar o levantamento e a análise dos requisitos funcionais e não funcionais da aplicação;
- modelar o sistema utilizando conceitos de análise orientada a objetos, incluindo diagramas de casos de uso, atividades e classes de domínio;
- implementar a camada de *back-end* utilizando o *framework Flask*, contemplando a lógica de negócio, regras de validação e acesso ao banco de dados;
- desenvolver a camada de *front-end* utilizando *React*, oferecendo uma interface responsiva e de fácil utilização;
- implementar funcionalidades de cadastro e autenticação de usuários, com diferenciação de perfis de cliente e administrador;
- disponibilizar ao administrador funcionalidades para cadastro, alteração, exclusão e consulta de produtos, bem como visualização de pedidos realizados;
- permitir que o cliente navegue pelo catálogo de produtos, monte um carrinho de compras, finalize pedidos e acesse o histórico de compras em área reservada;
- gerar automaticamente comprovantes de pedido em formato PDF para que o cliente possa salvar ou imprimir suas compras.

1.5 Metodologia

A metodologia adotada para o desenvolvimento do Venezart baseou-se em etapas iterativas e incrementais. Inicialmente, foi realizado o levantamento de requisitos, por meio da identificação das necessidades do administrador da loja e dos futuros clientes. Em seguida, procedeu-se à modelagem do sistema utilizando a abordagem orientada a objetos, com a elaboração de diagramas de casos de uso, de atividades e de classes de domínio.

Na etapa de implementação, foram utilizadas as tecnologias definidas na fase de planejamento, com a construção do *back-end* em *Flask* e do *front-end* em *React*. Ao longo do desenvolvimento, foram realizados testes parciais para validação das funcionalidades, correção de erros e ajustes de interface, buscando garantir a consistência entre o que foi especificado e o que foi implementado.

Por fim, o sistema foi avaliado quanto ao atendimento dos requisitos propostos e à experiência de uso, considerando aspectos como organização das telas, clareza das informações e fluxo de navegação.

1.6 Organização do trabalho

Este trabalho está organizado em quatro capítulos, além dos elementos pré-textuais e pós-textuais.

No Capítulo 1, apresenta-se a introdução do projeto, incluindo a contextualização, o problema de pesquisa, a justificativa, os objetivos e a metodologia adotada.

No Capítulo 2, são descritos o desenvolvimento do sistema, as características da aplicação, a arquitetura física e lógica, os requisitos funcionais e não funcionais, bem como os principais diagramas e protótipos envolvidos.

No terceiro capítulo, é abordada a etapa de codificação, contemplando detalhes de implementação, principais trechos de código e decisões técnicas adotadas.

Por fim, no quarto capítulo, apresentam-se as conclusões do trabalho, as limitações identificadas e sugestões de melhorias e trabalhos futuros.

2 Desenvolvimento

O Venezart é uma aplicação web desenvolvida com Flask no *back-end* e React no *front-end*, projetada para funcionar como uma loja especializada em utensílios artísticos, com foco em materiais voltados para pintura em tela. O objetivo do sistema é oferecer uma experiência clara, rápida e intuitiva para os clientes, ao mesmo tempo em que fornece ao administrador ferramentas eficientes para controle do catálogo e dos pedidos realizados. Os usuários do Venezart podem navegar pelos produtos, visualizar informações detalhadas, adicionar itens ao carrinho, ajustar quantidades, remover produtos e finalizar a compra. Após a conclusão do pedido, o sistema gera automaticamente um comprovante em PDF, permitindo ao usuário baixar e armazenar seu registro da compra. Cada usuário também possui acesso ao próprio histórico de compras por meio de uma área reservada. O sistema conta ainda com um perfil administrador, capaz de cadastrar novos produtos, editar informações existentes, excluir itens do catálogo e visualizar todos os pedidos feitos na plataforma, garantindo um gerenciamento eficiente e centralizado da loja.

2.1 Características da aplicação

2.1.1 Atores e seus papéis

- **Classes User:** Os usuários do Venezart podem navegar pelos produtos, visualizar informações detalhadas, adicionar itens ao carrinho, ajustar quantidades, remover produtos e finalizar a compra. Após a conclusão do pedido, o sistema gera automaticamente um comprovante em PDF, permitindo ao usuário baixar e armazenar seu registro da compra. Cada usuário também possui acesso ao próprio histórico de compras por meio de uma área reservada.
- **Clientes:** Os clientes são cadastrados na plataforma. Cada cliente recebe um código único para sua identificação. As informações de cadastro incluem CPF/CNPJ, nome, endereço, telefone, *e-mail*, descrição de *login* e senha.

2.1.2 Classes de usuários e características

Os usuários e suas características são descritos no Quadro 1.

Quadro 1 – Classes de usuários e características

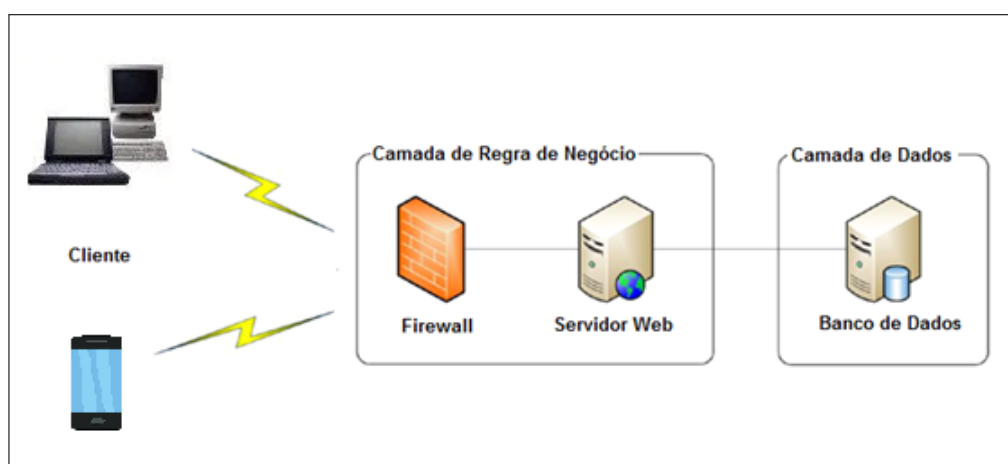
Classe	Descrição
Administradores	São os profissionais com as responsabilidades pelo gerenciamento do sistema, de servidores, de bancos de dados, do aplicativo e dos serviços relacionados.
Usuários/Clientes	São as pessoas que utilizam o <i>software</i> no seu dia a dia; aproveitam as funcionalidades oferecidas pelo software; possuindo o cadastro no site, podem acessar os materiais disponíveis e os recursos eletrônicos.
Desenvolvedores	São os profissionais responsáveis pela criação e implementação do <i>software</i> ; desenvolvem o sistema, escrevendo o código necessário para sua funcionalidade; e colaboram para garantir que funcione corretamente, seja seguro e atenda aos requisitos definidos.

Fonte: Autoria própria (2025)

2.2 Arquitetura física

A arquitetura física adota o modelo cliente-servidor. A visualização das páginas é feita no navegador do usuário, enquanto a execução da aplicação fica por conta da comunicação de um servidor com o banco de dados. A Figura 1 apresenta este modelo.

Figura 1 – Processo de funcionamento da aplicação.



Fonte: Autoria própria (2025)

2.2.1 Tecnologias utilizadas

As tecnologias utilizadas foram definidas com base no levantamento de requisitos, no ambiente operacional e nas restrições de projeto e implementação. A seleção visa garantir compatibilidade, escalabilidade, segurança e produtividade no desenvolvimento da aplicação.

- **Plataforma de desenvolvimento:** O ambiente de desenvolvimento integrado (IDE) adotado foi o *Visual Studio Code (1.100)*, utilizado por sua compatibilidade com múltiplas plataformas e integração com ferramentas de desenvolvimento.
- **Front-end:** Trata-se da parte do desenvolvimento responsável pela interface gráfica com o usuário, ou seja, o que ele vê e interage com a aplicação. Foram utilizados:
 - *Vue.js (3.5.16)*: Estrutura *JavaScript (ES2024)* acessível, de alto desempenho e versátil para criar interfaces de usuário;
 - *Nuxt.js (3.17)*: Estrutura para criar aplicativos que facilita o desenvolvimento, tanto da interface gráfica quanto da lógica e dados, com *Vue.js*.
- **Back-end:** Trata-se da parte da aplicação responsável pela lógica, armazenamento e processamento de dados, e a interação com bancos de dados e servidores. Foram utilizados:
 - *Node.js (22.16.0)*: Ambiente de execução de multiplataforma, que permite criar servidores, aplicações e programas de automação de tarefas.
 - *Express.js (5.1.0)*: Estrutura para aplicativos em *Node.js* que fornece um conjunto robusto de recursos para aplicativos *web* e móveis.
- **Banco de dados:**
 - *SQLite (3.50.0)*: Biblioteca em linguagem C que implementa um mecanismo de banco de dados SQL pequeno, rápido e completo.
- **Mapeamento Objeto-Relacional(ORM):**
 - *Sequelize (v5)*: Mapeamento relacional para diversos bancos de dados (*SQLite* e outros) que apresenta suporte para carregamento e integração.

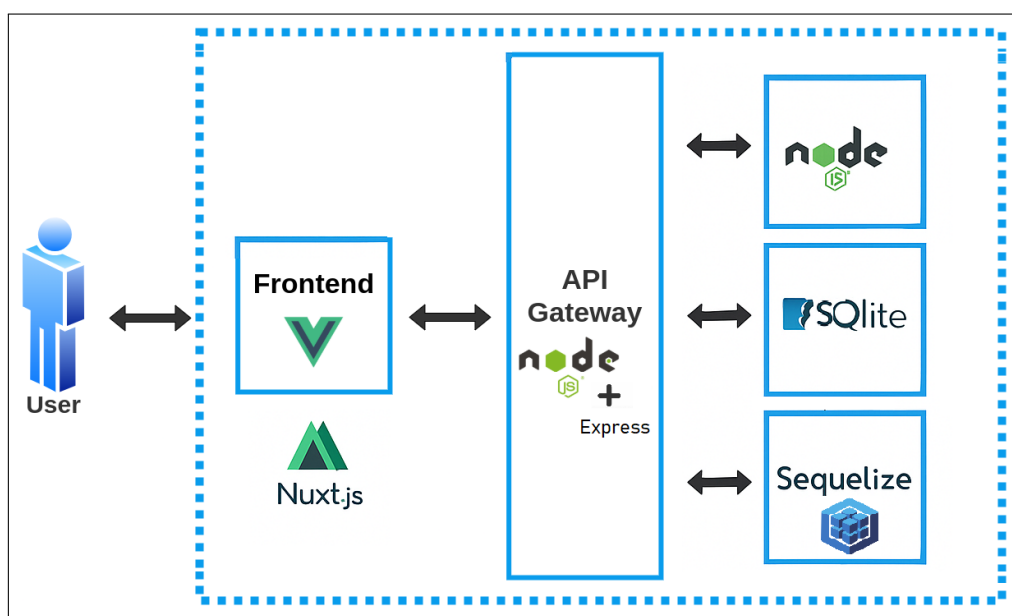
2.3 Arquitetura lógica

Na arquitetura lógica da aplicação, foram definidos os diferentes componentes e módulos que interagem e se organizam para realizar as funcionalidades da aplicação. A Figura 2 apresenta de forma simplificada a estrutura utilizada.

- **Front-end:** A camada de interface do usuário foi desenvolvida com o *Vue.js*, utilizando o *Nuxt.js* para estruturação do projeto, o roteamento e a renderização das páginas, tanto no lado do cliente quanto do servidor.
- **Back-end:** A lógica foi implementada em *Node.js*, com o uso do *Express.js* para a criação da *API RESTful*.

- Banco de dados: Os dados da aplicação são armazenados utilizando o *SQLite*.
- Mapeamento Objeto-Relacional(ORM): No mapeamento da aplicação com o banco de dados, foi utilizado o *Sequelize*.
- Tecnologia *web* e integração: A comunicação entre o *front-end* e o *back-end* foi feita por meio de *API RESTful*, utilizando o padrão JSON para a troca de dados. A *API Restful* é um tipo de interface de programação de aplicações (API) que se baseia em recursos (como URLs) e utiliza os métodos (*GET*, *POST*, *PUT*, *DELETE*, etc.) para realizar operações como leitura, criação, atualização e exclusão de dados.

Figura 2 – Arquitetura lógica com as principais tecnologias.



Fonte: Autoria própria (2025)

3 Conclusão

A proposta de criação de uma loja virtual especializada em materiais para pintura em tela mostrou-se relevante diante da necessidade de soluções mais organizadas e específicas para nichos de mercado. Ao longo do projeto, foi possível estruturar um sistema capaz de gerenciar produtos, usuários e pedidos, oferecendo uma experiência mais clara e eficiente tanto para clientes quanto para o administrador.

Mesmo atendendo aos objetivos propostos, o sistema apresenta possibilidades de evolução, como a integração com meios de pagamento online, melhorias nos mecanismos de segurança e a inclusão de relatórios gerenciais mais detalhados.

Referências

Express.js. **Code Editing. Redefined.** 5.1.0. Disponível em: <https://expressjs.com/pt-br/>. Acesso em: 12 abr. 2025. 9

JavaScript. **Code Editing. Redefined.** ES2024. Disponível em: <https://www.java.com/pt-BR/>. Acesso em: 12 abr. 2025. 9

Node.js. **Code Editing. Redefined.** 22.16.0. Disponível em: <https://nodejs.org/pt>. Acesso em: 12 abr. 2025. 9

Nuxt.js. **Code Editing. Redefined.** 3.17. Disponível em: <https://nuxt.com/>. Acesso em: 12 abr. 2025. 9

Sequelize. **Code Editing. Redefined.** v5. Disponível em: <https://sequelize.org/>. Acesso em: 12 abr. 2025. 9

SQLite. **Code Editing. Redefined.** 3.50.0. Disponível em: <https://www.sqlite.org/index.html>. Acesso em: 12 abr. 2025. 9

Visual Studio Code. **Code Editing. Redefined.** 1.100. Disponível em: <https://code.visualstudio.com>. Acesso em: 12 abr. 2025. 9

Vue.js. **Code Editing. Redefined.** 3.5.16. Disponível em: <https://vuejs.org/>. Acesso em: 12 abr. 2025. 9