**Documentação do Projeto VLJ Carnes - Qualidade e Sabor em Cada Corte**

**1. Introdução**

O VLJ Carnes é um projeto desenvolvido para oferecer uma plataforma de e-commerce especializada em cortes de carne de alta qualidade, atendendo desde consumidores casuais até entusiastas da gastronomia. O sistema foi construído com o objetivo de proporcionar uma experiência de compra intuitiva, segura e eficiente, garantindo a procedência e a qualidade dos produtos.

**1.1 Objetivos**

Fornecer uma plataforma online para comercialização de cortes de carne selecionados.

* Garantir um controle rigoroso de qualidade e procedência dos produtos.
* Oferecer um atendimento especializado e um portfólio variado de cortes nobres e opções do dia a dia.
* Implementar uma solução escalável e de fácil manutenção.

**1.2 Público-Alvo**

* Consumidores que buscam carnes de qualidade para churrascos e pratos sofisticados.
* Restaurantes e estabelecimentos que desejam adquirir produtos premium.
* Pessoas que valorizam conveniência e segurança em compras online.

**2. Tecnologias Utilizadas**

**2.1 Back-end**

* **Linguagem:** Java
* **Framework**: Spring Boot (para criação da API RESTful)

**Banco de Dados:**

1. **Inicialmente:** PostgresSQL (devido à estrutura robusta)
2. **Migrado para:** MongoDB (por simplicidade e flexibilidade)
3. **Retorno para:** PostgresSQL (devido a necessidades específicas do projeto)

**2.2 Front-end**

* **Biblioteca:** React.js (para construção da interface dinâmica)
* **Gerenciamento de Estado:** Context API ou Redux (se aplicável)
* **Estilização:** CSS Modules ou Styled Components**.**

**2.3 Hospedagem e Deploy**

* **Front-end:** Hospedado no Render.
* **Back-end:** Hospedado no Render
* **Banco de Dados:** PostgresSQL em nuvem.

**3. Arquitetura e Fluxo do Sistema**

**3.1 Fluxograma do Processo**

**1. Acesso ao Site**  **→ Navegação pelos produtos.**

**2. Seleção do Produto → Adição ao carrinho.**

**3. Checkout**  **→ Inserção de dados de entrega e pagamento.**

**4. Confirmação**  **→ Processamento do pedido e envio de confirmação.**

**5. Entrega**  **→ Acompanhamento do status.**

**Fluxograma**:

Diagrama, Esquemático

O conteúdo gerado por IA pode estar incorreto.

A elaboração do fluxo de funcionamento foi determinada pelo analista, Jefferson Custodio e direcionada pelos demais desenvolvedores, Lucca Russo e Vinicius Abs.

**3.2 Estrutura da API**

**Todos os endpoints principais**

**CustomerController:**

* GET /customer/login/{email}/{password} – Verifica o login de um cliente utilizando o email e a senha.
* POST /customer/insert – Insere um novo cliente.
* PUT /customer/update – Atualiza as informações do cliente.

**ProductController:**

* GET /product/get/all – Retorna todos os produtos.
* GET /product/get/{id} – Retorna um produto pelo seu ID.
* GET /product/get/category/{id} – Retorna todos os produtos de uma categoria específica.
* POST /product/insert – Insere um novo produto.
* PUT /product/update – Atualiza as informações de um produto.
* DELETE /product/delete/{id} – Deleta um produto pelo seu ID.

**3.3 Banco de Dados**

* **Tabelas Principais (PostgresSQL):**

**Product:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **id** | **name** | **price\_in\_cent** | **description** | **id\_imagem** | **id\_category** |

**Customer:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Id** | **name** | **email** | **password** | **address** |

**Category:**

|  |  |
| --- | --- |
| **Id** | **nome** |

**4. Desenvolvimento**

**4.1 Fases do Projeto**

**1. Prototipagem e Planejamento**

* Definição de requisitos e fluxo de usuário.
* Escolha das tecnologias.

**2. Desenvolvimento da API**

* Configuração do Spring Boot.
* Implementação dos endpoints.
* Migração entre bancos de dados (PostgresSQL→ MongoDB → PostgresSQL ).

**3. Construção do Front-end**

* Criação das telas com React.
* Integração com a API.
* Testes de usabilidade.

**4. Testes e Ajustes**

* Validação por um usuário de teste.
* Correção de bugs e melhorias na experiência.

**5. Deploy e Entrega**

* Hospedagem do sistema.
* Documentação final.

**4.2 Desafios e Soluções**

* **Problema**: Inconsistências na API durante o desenvolvimento.
* **Solução:** Refatoração de endpoints e melhor tratamento de erros.

* **Problema:** Mudança de banco de dados**.**
* **Solução:** Adequação do modelo de dados e queries.

* **Problema:** Telas não previstas inicialmente.
* **Solução:** Adaptação do front-end com novas rotas e componentes.

**Resumo:**   
 Para o desenvolvimento utilizamos prioritariamente o Java na criação da API e o a Biblioteca React no desenvolvimento do Front-end do projeto. Utilizamos o Spring boot como Framework, aproveitando-se da sua estrutura muito comum e disseminada no mercado, que também é utilizados em grandes empresas.

Iniciamos utilizando uma Aplicação já feita anteriormente por um membro do grupo, Vinícius Abs, a modificamos para a finalidade do projeto.

Posteriormente pensamos no banco de dados e decidimos usar o SQL Server primeiramente. No andar do projeto vimos que não tinha necessidade de uma grande estrutura de dados e tabelas, nem da complexidade do SQL, então, mudamos pro MongoDB. Mapeamos os endpoints e nesse momento a estrutura do projeto já estava completa.

(Tivemos muitos problemas com a API pois durantes o desenvolvimento ocorreram diversos imprevistos e eventos não pensados anteriormente)

Começamos o desenvolvimento do Front-end com React e nisso, observamos a necessidade do desenvolvimento de algumas telas que ainda não tínhamos pensado além de mudarmos o banco novamente para SQL Server.

**5. Testes e Validação**

**5.1 Testes Realizados**

* **Testes Técnicos:**
* Validação de endpoints (Postman/Insomnia).
* Testes de carga e desempenho.
* **Teste de Usuário:**
* Simulação de compra completa (seleção, checkout, pagamento).
* Identificação de falhas no fluxo e correções.

**5.2 Melhorias Implementadas**

* Correção de bugs no carrinho de compras.
* Ajustes na responsividade do layout.
* Otimização de consultas ao banco de dados**.**

**6. Conclusão e Próximos Passos**

O VLJ Carnes foi desenvolvido com sucesso, atendendo aos requisitos de qualidade e usabilidade. Como próximas etapas, sugere-se:

* **Implementação de um sistema de avaliação de produtos.**
* **Integração com formas de pagamento (Pix, cartões).**
* **Expansão do catálogo e funcionalidades de fidelidade.**

**---**

**Equipe de Desenvolvimento:**

* **Analista/PO:** Jefferson Custodio
* **Desenvolvedores:** Lucca Russo, Vinicius Abs

**Repositório**: link do GitHubfront e back    
<https://github.com/LuccaRusso/vlj-front.git>   
<https://github.com/PACKAPI/APIPROD.git>

**Deploy:** link do render

[https://vlj-front.onrender.com](https://vlj-front.onrender.com/)