INSTITUTO FEDERAL
Tocantins

MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO TOCANTINS
***CAMPUS* PARAÍSO DO TOCANTINS**

Relatório

Projeto: Sistema de Gerenciamento de Estoque (CRUD)
Tecnologia Base: Laravel Framework (v. 12.41.1), Php (v. 8.4.15)
Acadêmicos: Leonardo Santos Botelho – Matrícula: 20201060630078 e Elias Ferreira de Borba – Matrícula: 20201060630191

1. Introdução

O presente relatório técnico tem por objetivo analisar a estrutura, o processo de desenvolvimento e as soluções técnicas implementadas em um sistema de gerenciamento de estoque baseado no paradigma **CRUD** (Create, Read, Update, Delete). O sistema foi desenvolvido utilizando o framework **Laravel** v. 12.41. e compreende a gestão de três entidades primárias: **Categorias**, **Produtos** e **Movimentações** de estoque. A análise foca na aderência a boas práticas de engenharia de software, na eficácia das soluções adotadas para desafios de consistência de dados e na otimização proporcionada pelas ferramentas e recursos empregados.

2. Processo de Desenvolvimento

O desenvolvimento do projeto seguiu a arquitetura **Model-View-Controller (MVC)**, inerente ao framework Laravel, o que permitiu uma clara separação de responsabilidades e facilitou a manutenção e escalabilidade do código.

2.1. Estrutura de Rotas e Mapeamento de Funcionalidades

A definição das rotas foi otimizada através do uso do método Route::resource no arquivo routes/web.php. Esta abordagem adere aos princípios **RESTful**, mapeando automaticamente as operações CRUD para os métodos correspondentes nos controladores, conforme detalhado na Tabela 1.

Tabela 1: Mapeamento de Rotas

Entidade	Controller Associado	Operações CRUD (Métodos)
Categoria	CategoriaController	Read, create, store, edit, update, delete
Produto	ProdutoController	Read, create, store, edit, update, delete
Movimentação	MovimentacaoController	Read, create, store, edit, update, delete

2.2. Implementação das Operações CRUD

2.2.1. Entidades Categoria e Produto (CRUD Padrão)

As operações para as entidades Categoria e Produto demonstram a aplicação direta e eficiente dos recursos do Laravel:

- **Create (store):** A persistência de dados é precedida por uma **validação de formulário** ($request->validate), garantindo a integridade e o formato dos dados de entrada. A criação do registro é realizada via *Mass Assignment* (Model::create($request->all())).
- **Read (index):** Para a listagem de produtos, foi implementado o recurso de **Eager Loading** (Produto::with('categoria')). Esta técnica é crucial para otimizar o desempenho, pois carrega os dados de relacionamento (Categoria) em uma única consulta ao banco de dados, mitigando o problema de *N+1 queries*.
- **Update/Delete (edit, update, destroy):** O projeto faz uso do **Route Model Binding**, onde o objeto do modelo ($categoria ou $produto) é injetado diretamente no método do controlador. Isso simplifica o código, eliminando a necessidade de consultas manuais para localizar o registro a ser manipulado.

2.2.2. Entidade Movimentação

A entidade Movimentacao introduz a lógica de negócio central do sistema, pois sua manipulação afeta diretamente o atributo quantidade da entidade Produto.

- **Lógica de Criação (store):**
  - **Validação:** Além da validação de formato, é realizada uma validação de regra de negócio: para o tipo saida, é verificada a **disponibilidade de estoque** ($request->quantidade > $produto->quantidade).
  - **Transação Implícita:** A atualização do estoque do produto ($produto->quantidade += $request->quantidade ou -=) ocorre imediatamente antes da criação do registro de movimentação.
- **Lógica de Atualização (update):**
  - **Reversão:** Para garantir a consistência, o método primeiramente **reverte o impacto** da movimentação antiga no estoque do produto.
  - **Reaplicação:** Em seguida, a nova movimentação é tratada como uma nova operação, com a devida verificação de estoque para saídas, e o novo impacto é aplicado ao produto.
- **Lógica de Exclusão (destroy/delete):**
  - **Compensação:** A exclusão de um registro de movimentação exige uma **operação de compensação** no estoque. Se a movimentação excluída era uma entrada, a quantidade é subtraída do produto; se era uma saída, a quantidade é somada.

3. Desafios e Soluções

O principal desafio técnico do projeto reside na **manutenção da integridade e consistência transacional** do estoque, especialmente em operações que envolvem a alteração de registros históricos de movimentação.

Tabela 2: Desafios Técnicos e Soluções Implementadas

Desafio Técnico	Solução Implementada	Justificativa
Consistência de Estoque em Alterações	Implementação da lógica de **Reversão e Reaplicação** no método update do MovimentacaoController.	Garante a **atomicidade lógica** da atualização. Ao reverter o estado anterior e aplicar o novo, evita-se a acumulação de erros de cálculo e assegura-se que o estoque final seja sempre o resultado da soma de todas as movimentações válidas.
Prevenção de Estoque Negativo	Validação de regra de negócio no método store e update para o tipo saida, que compara a quantidade solicitada com o estoque atual.	Implementação de uma **restrição de domínio** no nível da aplicação (Controller), prevenindo a persistência de dados que violam a regra de negócio fundamental do sistema.
Otimização de Consultas (N+1)	Uso de **Eager Loading** (with('categoria')) na listagem de produtos.	Redução da latência e do número de consultas ao banco de dados, melhorando significativamente o desempenho da aplicação em cenários de alta carga de dados.
Segurança e Higienização de Dados	Uso do *middleware* de **Validação de Formulário** do Laravel.	Proteção contra vulnerabilidades comuns, como *Mass Assignment* e injeção de dados maliciosos, garantindo que apenas dados validados e tipados sejam processados.

4. Recursos e Ferramentas

A escolha das ferramentas e bibliotecas foi determinante para a otimização do ciclo de desenvolvimento e para a qualidade do produto.

Tabela 3: Recursos e ferramentas utilizadas

Ferramenta / Tecnologia	Versão / Informação
PHP	8.4.15
Laravel	12.41.1
Servidor local	127.0.0.1:8000

Ferramenta / Tecnologia	Versão / Informação
Banco de dados	MySQL
Bootstrap	5.3.3
Composer	Não informado
Navegador (testes)	Chrome
Sistema Operacional	Windows
Blade Templates	Nativo do Laravel
Carbon	Nativo Laravel/PHP
Roteamento (Routes)	Nativo do Laravel

5. Conclusão

O projeto de Sistema de Gerenciamento de Estoque demonstra uma execução técnica de alto nível. A adesão ao padrão MVC e o uso idiomático dos recursos do Laravel (Route Model Binding, Eager Loading, Validação) estabelecem uma base de código limpa e manutenível. O ponto de maior complexidade, a **gestão da consistência do estoque** através da entidade Movimentacao, foi resolvido com uma lógica de compensação e reversão, o que é um indicativo de maturidade na resolução de problemas de **integridade de dados** em sistemas transacionais.