

MODELAGEM E VALIDAÇÃO DE SISTEMA

Felipe Eduardo Bohnen¹

Miguel Schneiders Flach²

Murilo Morosini³

Roney Bieger Anschau⁴

Resumo

Este relatório detalha a modelagem e validação de um sistema CRUD para gerenciamento de produtos em um mercado. A modelagem inclui Diagramas de Casos de Uso, de Sequência, de Classes e o Modelo Entidade-Relacionamento, que definem a estrutura e o comportamento do software. Padrões de projeto, como Singleton para gerenciamento de conexão e Arquitetura em Camadas, foram aplicados para promover modularidade. A validação, realizada através de testes unitários e funcionais, revelou falhas em validações de dados e na interação da interface, necessitando de correções. Um teste de carga foi tentado, mas a ausência de resultados impede uma análise de desempenho conclusiva. O trabalho visa aprimorar o sistema para garantir funcionalidade e robustez.

¹ Graduando do Curso de Ciência da Computação - Unoesc Campus São Miguel do Oeste.
felipebohnenf@gmail.com

² Graduando do Curso de Ciência da Computação - Unoesc Campus São Miguel do Oeste.
miguelstlach65@gmail.com

³ Graduando do Curso de Ciência da Computação - Unoesc Campus São Miguel do Oeste.
murilomorosini@gmail.com

⁴ Graduando do Curso de Ciência da Computação - Unoesc Campus São Miguel do Oeste.
roneyronyy@hotmail.com

1. INTRODUÇÃO

O presente relatório tem como objetivo apresentar a modelagem, validação e aprimoramento de um sistema de gerenciamento de produtos para um mercado, conforme requisitos da disciplina de Engenharia de Software II. O sistema, que consiste em um CRUD (Create, Read, Update, Delete) completo, visa otimizar o controle de estoque e as operações relacionadas aos produtos. Este documento abordará as etapas de modelagem de software, utilizando diagramas UML (Unified Modeling Language) e o Modelo Entidade-Relacionamento (ER), além de identificar padrões de projeto aplicados. Posteriormente, serão detalhados os testes unitários, funcionais e de carga realizados para verificar a robustez e o desempenho do sistema.

2. DESENVOLVIMENTO

2.1 Modelagem de Software

A fase de modelagem é crucial para a compreensão da estrutura, comportamento e requisitos do sistema. Foram elaborados diagramas de Casos de Uso, de Sequência, de Classes e o Modelo Entidade-Relacionamento, que detalham diferentes perspectivas do software.

2.2 Diagrama de Casos de Uso

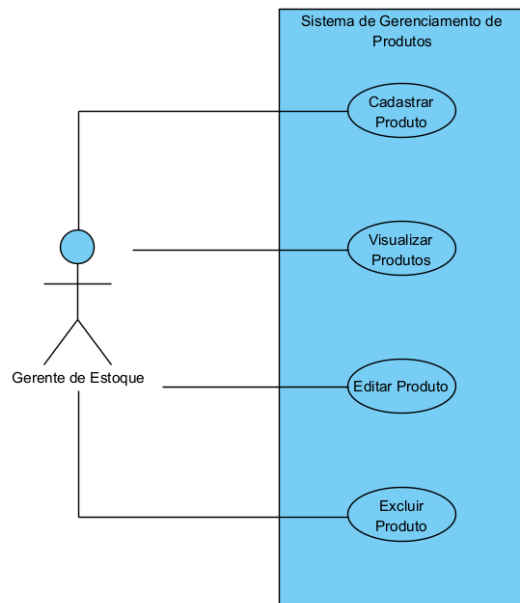
Este Diagrama de Casos de Uso modela as principais funcionalidades do Sistema de Gerenciamento de Produtos e o ator que interage com ele. O Gerente de Estoque é o ator central, representando o usuário que irá operar o sistema. Ele possui associações diretas com os seguintes casos de uso, que são as funcionalidades que o sistema oferece:

Cadastrar Produto: Permite ao Gerente de Estoque inserir novos produtos no sistema, adicionando suas informações essenciais.

Visualizar Produtos: Habilita o Gerente de Estoque a consultar a lista completa de produtos já cadastrados no sistema.

Editar Produto: Concede ao Gerente de Estoque a capacidade de modificar os detalhes de um produto existente, como preço ou quantidade em estoque.

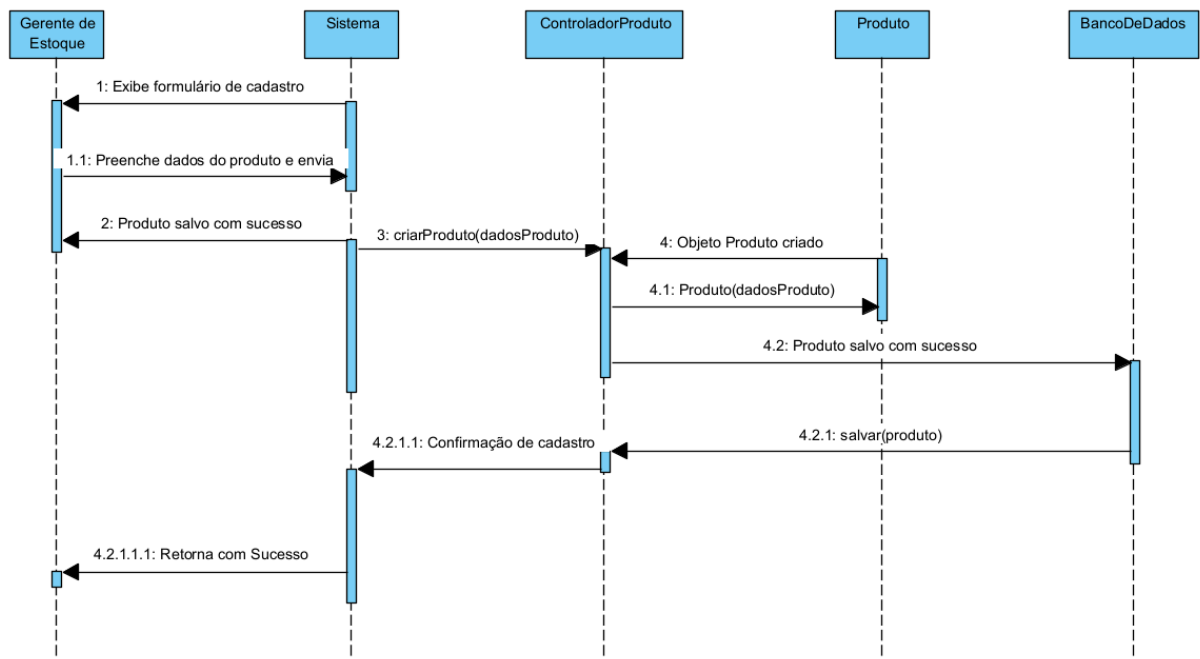
Excluir Produto: Permite que o Gerente de Estoque remova produtos do sistema quando não são mais necessários.



Fonte: Autoria própria (2025).

2.3 Diagrama de Sequência

Este Diagrama de Sequência detalha a ordem cronológica das interações entre os diferentes componentes do sistema quando um usuário (Gerente de Estoque) solicita o cadastro de um novo produto. O Gerente de Estoque inicia o fluxo ao solicitar a funcionalidade de cadastro. A Interface do Usuário (Sistema) responde exibindo um formulário para preenchimento dos dados do produto. Após o preenchimento, o Gerente de Estoque envia as informações. A Interface do Usuário passa esses dados para o `ControladorProduto`, que é responsável pela lógica de negócio. O `ControladorProduto` cria uma nova instância do objeto `Produto` com os dados fornecidos. Em seguida, o `ControladorProduto` interage com o Banco de Dados (ou a camada de persistência, como um DAO) para salvar as informações do novo produto. O Banco de Dados retorna uma confirmação de sucesso para o `ControladorProduto`. Finalmente, o `ControladorProduto` informa a Interface do Usuário sobre o sucesso da operação, que então exibe uma mensagem de confirmação ao Gerente de Estoque.



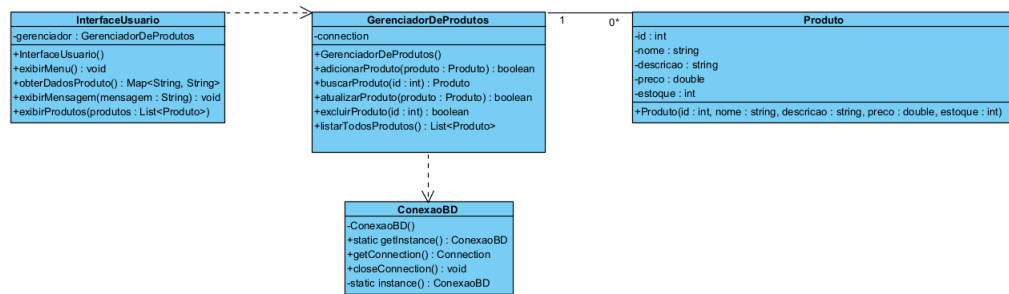
Fonte: Autoria própria (2025).

2.4 Diagrama de Classes

ConexaoBD (Singleton) garante uma única conexão com o banco de dados para todo o sistema. As classes se relacionam da seguinte forma: a InterfaceUsuario depende da GerenciadorDeProdutos para operar; a GerenciadorDeProdutos gerencia os objetos Produto; e a GerenciadorDeProdutos utiliza a ConexaoBD para acessar o banco de dados. O diagrama fornece uma visão clara da organização do código e da colaboração entre os componentes.

2.5 Modelo Entidade-Relacionamento (ER)

Este Modelo Entidade-Relacionamento (ER) ilustra a estrutura do banco de dados do sistema, com foco nas informações persistentes. O modelo é composto por uma única Entidade: PRODUTOS. Esta representa a tabela fundamental que armazena todos os dados dos produtos. Inclui atributos como id (chave primária e identificador único), nome, descricao, preco e estoque, cada um com seu tipo de dado e restrições de nulidade. Devido à simplicidade do escopo, este diagrama ER não apresenta relacionamentos entre múltiplas entidades, focado inteiramente na organização dos dados da tabela de produtos. Ele serve como blueprint para a criação da base de dados do sistema.



Fonte: Autoria própria (2025).

2.6 Uso de Padrões de Projeto

Os padrões de projeto a seguir foram aplicados e são visíveis na modelagem do sistema:

Padrão Singleton

Finalidade: Garante uma única instância para otimizar recursos.

Onde ver: Implementado na classe **ConexaoBD** (construtor privado e método `getInstance()`) no Diagrama de Classes.

Padrão de Camadas (Arquitetura)

Finalidade: Organizar o sistema em responsabilidades claras (separação de preocupações).

Onde ver: A estrutura em camadas (Apresentação, Lógica/Dados, Infraestrutura) é evidente nas relações entre **InterfaceUsuario**, **GerenciadorDeProdutos** e **ConexaoBD** no Diagrama de Classes.

Produto			
	id	integer(10)	U
	nome	varchar(255)	
	descricao	varchar(255)	
	preco	decimal(10, 2)	
	estoque	integer(10)	

Fonte: Autoria própria (2025).

2.2 Validação e Testes

Esta seção apresenta os resultados obtidos nos testes unitários, de funcionalidade e de carga, fornecendo uma análise sobre o comportamento e desempenho do sistema.

2.2.1 Testes Unitários

O Teste Unitário foca na verificação isolada de componentes. Nos 27 testes executados, 4 falharam. As falhas estão ligadas a validação do modelo Produto, especificamente permitindo preços e estoque negativos, e indicando problemas em validações múltiplas, o que exige revisão das regras de negócio.

```

Codeception PHP Testing Framework v5.3.2 https://stand-with-ukraine.pp.ua

Unit Tests (27) -----
x ContactFormTest: Email is sent on contact (0.09s)
+ LoginFormTest: Login no user (0.01s)
+ LoginFormTest: Login wrong password (0.02s)
+ LoginFormTest: Login correct (0.01s)
+ ProdutosTest: Validation success (0.02s)
+ ProdutosTest: Nome required validation fail (0.01s)
x ProdutosTest: Preço negative validation fail (0.02s)
x ProdutosTest: Estoque negative validation fail (0.01s)
x ProdutosTest: Multiple validation fails (0.01s)
+ ProdutosTest: Preço not numeric validation fail (0.01s)
+ UserTest: Find user by id (0.00s)
+ UserTest: Find user by access token (0.00s)
+ UserTest: Find user by username (0.00s)
+ UserTest: Validate user (0.00s)
+ AlertTest: Single error message (0.05s)
+ AlertTest: Multiple error messages (0.00s)
+ AlertTest: Single danger message (0.00s)
+ AlertTest: Multiple danger messages (0.00s)
+ AlertTest: Single success message (0.00s)
+ AlertTest: Multiple success messages (0.00s)
+ AlertTest: Single info message (0.00s)
+ AlertTest: Multiple info messages (0.00s)
+ AlertTest: Single warning message (0.00s)
+ AlertTest: Multiple warning messages (0.00s)
+ AlertTest: Single mixed messages (0.00s)
+ AlertTest: Multiple mixed messages (0.00s)
+ AlertTest: Flash integrity (0.01s)
-----
Time: 00:00.467, Memory: 26.00 MB

There were 4 failures:
1) ContactFormTest: Email is sent on contact
Test tests\unit\models\ContactFormTest.php:testEmailIsSentOnContact
Failed asserting that a boolean is not empty.
#1 C:\Users\miguel\OneDrive\Área de Trabalho\trabalho-prog3-yii2\tests\unit\models\ContactFormTest.php:27
#2 C:\Users\miguel\OneDrive\Área de Trabalho\trabalho-prog3-yii2\vendor\bin\codecept:119
2) ProdutosTest: Preço negative validation fail
Test tests\unit\models\ProdutosTest.php:testPrecoNegativeValidationFail
A validação deveria falhar com preço negativo.
Failed asserting that true is false.
#1 C:\Users\miguel\OneDrive\Área de Trabalho\trabalho-prog3-yii2\tests\unit\models\ProdutosTest.php:84
#2 C:\Users\miguel\OneDrive\Área de Trabalho\trabalho-prog3-yii2\vendor\bin\codecept:119
3) ProdutosTest: Estoque negative validation fail
Test tests\unit\models\ProdutosTest.php:testEstoqueNegativeValidationFail
A validação deveria falhar com estoque negativo.
Failed asserting that true is false.
#1 C:\Users\miguel\OneDrive\Área de Trabalho\trabalho-prog3-yii2\tests\unit\models\ProdutosTest.php:99
#2 C:\Users\miguel\OneDrive\Área de Trabalho\trabalho-prog3-yii2\vendor\bin\codecept:119
4) ProdutosTest: Multiple validation fails
Test tests\unit\models\ProdutosTest.php:testMultipleValidationFails
Deveria haver um erro para o campo preço.
Failed asserting that an array has the key 'preco'.
#1 C:\Users\miguel\OneDrive\Área de Trabalho\trabalho-prog3-yii2\tests\unit\models\ProdutosTest.php:117
#2 C:\Users\miguel\OneDrive\Área de Trabalho\trabalho-prog3-yii2\vendor\bin\codecept:119

FAILURES!
Tests: 27, Assertions: 106, Failures: 4.

```

Fonte: Autoria própria (2025).

2.2.2 Testes Funcionais

O Teste de Funcionalidade avalia se as funcionalidades correspondem aos requisitos. Das 13 funcionalidades testadas, 4 falharam, como mostram as figuras abaixo. As falhas incluem problemas no envio de formulário de contato, na exibição de mensagens de erro para dados inválidos, na atualização (erro 404) e na exclusão de produtos (botão 'Delete' não encontrado).

```

[...]
```

Fonte: Autoria própria (2025).

```

[...]
```

Fonte: Autoria própria (2025).

```

[...]
```

Fonte: Autoria própria (2025).

```

[...]
```

Fonte: Autoria própria (2025).


```

@unittest:
Vtl-log: [yii\db\Connection::open] Opening DB connection: mysql:host=localhost;dbname=yilbrazil_test
html: C:\Users\luciano\Documents de Trabalho\trabalho-prog-yii\tests_output\functional\ProductsTest.atualizarProduto.fail.html
Response: C:\Users\luciano\Documents de Trabalho\trabalho-prog-yii\tests_output\functional\ProductsTest.atualizarProduto.fail.html
5) ProductsTest: Deletar um produto
FAIL: C:\Users\luciano\Documents de Trabalho\trabalho-prog-yii\tests_output\functional\ProductsTest.atualizarProduto.fail.html
@unittest:
Vtl-log: [yii\db\Connection::open] Opening DB connection: mysql:host=localhost;dbname=yilbrazil_test
html: C:\Users\luciano\Documents de Trabalho\trabalho-prog-yii\tests_output\functional\ProductsTest.atualizarProduto.fail.html
Response: C:\Users\luciano\Documents de Trabalho\trabalho-prog-yii\tests_output\functional\ProductsTest.atualizarProduto.fail.html
6) ProductsTest: Deletar um produto
FAIL: C:\Users\luciano\Documents de Trabalho\trabalho-prog-yii\tests_output\functional\ProductsTest.atualizarProduto.fail.html
@unittest:
Vtl-log: [yii\db\Connection::open] Opening DB connection: mysql:host=localhost;dbname=yilbrazil_test
@unittest:
Vtl-log: [yii\db\Connection::open] Opening DB connection: mysql:host=localhost;dbname=yilbrazil_test
html: C:\Users\luciano\Documents de Trabalho\trabalho-prog-yii\tests_output\functional\ProductsTest.atualizarProduto.fail.html
Response: C:\Users\luciano\Documents de Trabalho\trabalho-prog-yii\tests_output\functional\ProductsTest.atualizarProduto.fail.html
@unittest:
Vtl-log: [yii\db\Connection::open] Opening DB connection: mysql:host=localhost;dbname=yilbrazil_test
@unittest:
@unittest:
@unittest:
@unittest:
@unittest:
@unittest:
@unittest:
Vtl-log: [yii\db\Connection::open] Opening DB connection: mysql:host=localhost;dbname=yilbrazil_test

```

Fonte: Autoria própria (2025).

```

html: C:\Users\luciano\Documents de Trabalho\trabalho-prog-yii\tests_output\functional\ProductsTest.atualizarProduto.fail.html
html: C:\Users\luciano\Documents de Trabalho\trabalho-prog-yii\tests_output\functional\ProductsTest.atualizarProduto.fail.html
Response: C:\Users\luciano\Documents de Trabalho\trabalho-prog-yii\tests_output\functional\ProductsTest.atualizarProduto.fail.html
5) ProductsTest: Deletar um produto
Test: TestsFunctionalProductsTest.php:deletarProduto
Stop: Click "Delete"
Fail: Link or Button by name or CSS or XPath element with "Delete" was not found.

Scenario Steps:
4. $I->click("Delete") at tests\functional\ProductsTest.php:130
3. $I->waitForPageToLoad.php:produtos?id=6127) at tests\functional\ProductsTest.php:122
2. $I->haveRecord("app\models\Produto", ["name" => "teste", "preco" => 120, "estoque" => 1]) at tests\functional\ProductsTest.php:118
1. $I->haveRecord("app\models\Produto", []) at tests\functional\ProductsTest.php:25

@unittest:
Vtl-log: [yii\db\Connection::open] Opening DB connection: mysql:host=localhost;dbname=yilbrazil_test
html: C:\Users\luciano\Documents de Trabalho\trabalho-prog-yii\tests_output\functional\ProductsTest.deletarProduto.fail.html
Response: C:\Users\luciano\Documents de Trabalho\trabalho-prog-yii\tests_output\functional\ProductsTest.deletarProduto.fail.html
FAILURES!
Tests: 19, Assertions: 27, Failures: 6
F:\C:\Users\luciano\Documents de Trabalho\trabalho-prog-yii\

```

Fonte: Autoria própria (2025).

2.2.3 Testes de Carga

O Teste de Carga tem como objetivo avaliar o desempenho do sistema sob diferentes níveis de utilização e identificar possíveis gargalos ou limitações. Para este trabalho, o teste foi planejado com as configurações apresentadas abaixo, utilizando 100 usuários virtuais simulando requisições por um período de 10 segundos para rampa de carga, com 10 iterações de loop.

Thread Group

Name:

Comments:

Action to be taken after a Sampler error

☒ Continue
 ☐ Start Next Thread Loop
 ☐ Stop Thread
 ☐ Stop Test
 ☐ Stop Test Now

Thread Properties

Number of Threads (users):

Ramp-up period (seconds):

Loop Count: ☐ Infinite

☒ Same user on each iteration
☐ Delay Thread creation until needed
☐ Specify Thread lifetime

Duration (seconds):

Startup delay (seconds):

Fonte: Autoria própria (2025).

Contudo, os resultados apontam para desafios significativos de desempenho: o tempo médio de resposta de 7.665 milissegundos é consideravelmente elevado, e o desvio padrão de 1.791,56 milissegundos indica uma inconsistência notável na experiência do usuário. A vazão observada foi de 11.3 requisições por segundo. Tais métricas sugerem a necessidade de otimização para melhorar a agilidade e a previsibilidade do sistema sob carga.

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
HTTP Request	1000	7665	2080	9202	1791.56	0.00%	11.3/sec	477.01	1.53	43370.0
TOTAL	1000	7665	2080	9202	1791.56	0.00%	11.3/sec	477.01	1.53	43370.0

Fonte: Autoria própria (2025).

3. CONCLUSÃO

Este relatório apresentou a fase de modelagem de software para o sistema de gerenciamento de produtos, detalhando seus requisitos funcionais e estruturais através de diagramas UML e ER. A aplicação de padrões de projeto demonstrou a preocupação com a organização e manutenibilidade do código. A fase de validação e testes, embora tenha revelado algumas falhas importantes nos testes unitários e funcionais (especialmente relacionadas a validações e interação com a interface), forneceu insights valiosos sobre o comportamento do sistema. Adicionalmente, os testes de carga evidenciaram a necessidade de otimização de desempenho, com tempos de resposta elevados e inconsistentes sob simulação de uso.

<https://www.youtube.com/watch?v=I8mFWieYke8>