# RELATÓRIO DE TESTES DE CAIXA BRANCA

## Projeto Integrador 2025 - Equipe BYTECRAFT

**Analista de Qualidade:** Matheus Junio da Silva - 5382

**Data:** 20 de setembro de 2025

**Versão:** 1.0

# 1. INTRODUÇÃO

Este relatório apresenta os resultados dos testes de caixa branca realizados no sistema ByteCraft, seguindo as diretrizes estabelecidas no Plano Geral de Testes. Os testes foram conduzidos com conhecimento total do código-fonte da aplicação, permitindo verificar o comportamento interno dos métodos, estruturas de controle e fluxos de execução.

A execução dos testes teve como objetivo identificar possíveis falhas lógicas, garantir uma boa cobertura de código e validar se a implementação atende aos padrões de codificação estabelecidos pela equipe.

# 2. ESCOPO DOS TESTES

Os testes de caixa branca foram aplicados nas seguintes classes principais do sistema:

1. **Classe Aluno**: Entidade responsável por representar os estudantes no sistema
2. **Classe Professor**: Entidade que representa os professores e suas credenciais
3. **Classe Sala**: Entidade que gerencia o relacionamento entre professores e alunos

Cada classe foi testada individualmente para validar sua estrutura interna, métodos, construtores e relacionamentos.

# 3. METODOLOGIA APLICADA

## 3.1 Técnicas de Teste Utilizadas

1. **Análise de Fluxo de Controle**: Verificação de todos os caminhos possíveis através do código
2. **Teste de Valor Limite**: Validação de comportamento com valores extremos
3. **Teste de Estruturas de Dados**: Verificação da integridade das estruturas internas
4. **Teste de Construtores**: Validação de todos os construtores disponíveis

5. **Teste de Métodos Acessores**: Verificação completa de getters e setters

### 3.2 Critérios de Validação

Os testes foram desenvolvidos considerando os padrões de codificação estabelecidos:

1. Nomenclatura PascalCase para classes
2. Nomenclatura camelCase para métodos e variáveis
3. Uso adequado de anotações Lombok
4. Implementação correta de equals e hashCode

# 4. RESULTADOS DETALHADOS

## 4.1 Classe Aluno - AlunoTest

**Total de Testes Executados:** 8

**Testes Aprovados:** 8

**Testes Reprovados:** 0

**Taxa de Sucesso:** 100%

**Funcionalidades Testadas:**

1. **Teste de Construtor com Parâmetros**: Validou a criação correta de instâncias com nome, pontuação e sala
2. **Teste de Getters e Setters**: Verificou o funcionamento adequado de todos os métodos de acesso
3. **Teste de Validação de Nome**: Confirmou a aplicação do padrão camelCase para variáveis
4. **Teste de Pontuação com Valores Limite**: Testou comportamento com valores extremos incluindo zero, negativos e máximo inteiro
5. **Teste de Equals e HashCode**: Validou que a comparação é baseada exclusivamente no campo alunoId
6. **Teste de Associação com Sala**: Verificou o relacionamento correto entre aluno e sala
7. **Teste de Estado Inicial**: Confirmou que objetos criados pelo construtor padrão possuem estado inicial adequado
8. **Cobertura de Código**: Todos os métodos públicos da classe foram testados

**Observações Técnicas:**

1. A implementação do Lombok está funcionando corretamente
2. Os construtores estão seguindo os padrões estabelecidos
3. O relacionamento com a entidade Sala está consistente

## 4.2 Classe Professor - ProfessorTest

**Total de Testes Executados:** 8

**Testes Aprovados:** 8

**Testes Reprovados:** 0

**Taxa de Sucesso:** 100%

**Funcionalidades Testadas:**

1. **Teste de Construtor com DTO**: Validou a criação através de ProfessorRequestDTO com senha criptografada
2. **Teste de Getters e Setters**: Verificou conformidade com padrão PascalCase para a classe
3. **Teste de Validação de CPF**: Confirmou o armazenamento adequado de CPF brasileiro
4. **Teste de Senha Criptografada**: Validou que senhas são armazenadas de forma segura
5. **Teste de Equals e HashCode**: Verificou que a comparação é baseada no campo CPF
6. **Teste de Associação com Sala**: Confirmou relacionamento correto entre professor e sala
7. **Teste de Estado Inicial**: Validou estado inicial do construtor NoArgsConstructor
8. **Teste de Construtor Completo**: Verificou funcionamento do construtor AllArgsConstructor

**Observações Técnicas:**

1. A lógica de criptografia de senha está sendo aplicada corretamente
2. O campo CPF está sendo usado adequadamente como identificador único
3. A integração com o sistema de autenticação está consistente

## 4.3 Classe Sala - SalaTest

**Total de Testes Executados:** 8

**Testes Aprovados:** 8

**Testes Reprovados:** 0

**Taxa de Sucesso:** 100%

**Funcionalidades Testadas:**

1. **Teste de Construtor com Professor**: Validou criação com professor associado
2. **Teste de Inicialização da Lista de Alunos**: Confirmou que a lista é inicializada automaticamente
3. **Teste de Getters e Setters**: Verificou funcionamento de todos os métodos de acesso
4. **Teste de Adição de Alunos**: Validou manipulação correta da lista de alunos

5. **Teste de Equals e HashCode**: Confirmou comparação baseada em salaId
6. **Teste de Estado Inicial**: Verificou estado inicial adequado
7. **Teste de Construtor Completo**: Validou construtor AllArgsConstructor
8. **Teste de Relacionamento Bidirecional**: Confirmou consistência entre Professor e Sala

**Observações Técnicas:**

1. A inicialização automática da lista de alunos está funcionando corretamente
2. Os relacionamentos bidirecionais estão mantendo consistência
3. A manipulação da coleção de alunos está adequada

# 5. CONFORMIDADE COM PADRÕES DE CODIFICAÇÃO

## 5.1 Análise de Conformidade

Durante os testes, foi verificada a aderência aos padrões estabelecidos no documento "Padrões de Codificação":

**Classes nomeadas em PascalCase**: Aluno, Professor, Sala

**Métodos nomeados em camelCase**: getNome, setNome, getPontuacao

**Variáveis nomeadas em camelCase**: alunoId, salaId

**Pacotes em minúsculas**: gov.ufv.bytecraft_prototipo.aluno

**Uso adequado de anotações Lombok**: @Getter, @Setter, @NoArgsConstructor

**Implementação correta de equals/hashCode**: Baseado em campos chave únicos

## 5.2 Pontos de Melhoria Identificados

1. Não foram identificadas violações significativas aos padrões estabelecidos
2. A implementação está seguindo as boas práticas definidas pela equipe
3. Os comentários nos testes seguem as diretrizes de documentação

# 6. COBERTURA DE CÓDIGO

## 6.1 Métricas de Cobertura

1. **Cobertura de Classes**: 100% (3/3 classes testadas)
2. **Cobertura de Métodos Públicos**: 100% (todos os getters, setters e construtores)
3. **Cobertura de Linhas de Código**: Estimada em 95% das linhas executáveis
4. **Cobertura de Cenários**: 100% dos cenários básicos e de exceção

## 6.2 Caminhos de Execução Testados

1. Construtores padrão e parametrizados
2. Operações de leitura e escrita de propriedades
3. Relacionamentos entre entidades
4. Comparação de objetos (equals/hashCode)
5. Manipulação de coleções

# 7. IDENTIFICAÇÃO DE DEFEITOS

## 7.1 Defeitos Críticos

**Total:** 0

Não foram identificados defeitos críticos que impeçam o funcionamento do sistema.

## 7.2 Defeitos Médios

**Total:** 0

Não foram identificados defeitos de prioridade média.

## 7.3 Defeitos Menores

**Total:** 0

Não foram identificados defeitos menores.

# 8. ANÁLISE DE RISCOS

## 8.1 Riscos Técnicos Identificados

1. **Baixo Risco**: As classes básicas estão bem estruturadas
2. **Dependências Externas**: Lombok está funcionando adequadamente
3. **Integridade de Dados**: Relacionamentos entre entidades estão consistentes

## 8.2 Recomendações

1. Manter os padrões de codificação atuais
2. Continuar utilizando testes automatizados para novas funcionalidades
3. Implementar testes de integração para validar o comportamento conjunto das classes

# 9. CONCLUSÕES

## 9.1 Avaliação Geral

Os testes de caixa branca demonstraram que o código-fonte do sistema ByteCraft está em conformidade com os padrões estabelecidos e apresenta alta qualidade estrutural. Todas as funcionalidades testadas funcionaram conforme esperado, sem identificação de defeitos.

### 9.2 Qualidade do Código

1. **Estrutura**: Bem organizada e seguindo padrões de projeto
2. **Manutenibilidade**: Alta, devido ao uso adequado de anotações e convenções
3. **Legibilidade**: Excelente, com nomenclatura clara e consistente
4. **Testabilidade**: Boa, permitindo fácil criação de casos de teste

### 9.3 Aprovação para Próxima Fase

Com base nos resultados obtidos, as classes testadas estão aprovadas para integração com outras partes do sistema. A implementação atende aos critérios de qualidade estabelecidos no Plano de Garantia de Qualidade do Produto.

## 10. RESPONSABILIDADES E APROVAÇÕES

**Executado por:** Matheus Junio da Silva - Analista de Qualidade

**Data de Execução:** 18 de setembro de 2025

**Status:** Aprovado para continuidade do desenvolvimento

Este relatório confirma que a fase de testes de caixa branca foi concluída com sucesso, validando a qualidade interna do código e sua conformidade com os padrões estabelecidos pela equipe de desenvolvimento.