# Sistema de Gerenciamento Hospitalar

Documentação Técnica do Projeto

Curso de Ciência da Computação
Universidade Federal de Juiz de Fora

## Sumário

# 1 Introdução

Este documento apresenta a documentação técnica completa do **Sistema de Gerenciamento Hospitalar**, desenvolvido em **Java** utilizando o paradigma de **Programação Orientada a Objetos**, interface gráfica com **Swing** e gerenciamento de dependências via **Maven**.

O sistema tem como objetivo informatizar processos hospitalares essenciais, como:

- Cadastro e autenticação de usuários;
- Agendamento e gerenciamento de consultas;
- Controle de prontuários médicos;
- Emissão de documentos médicos;
- Persistência de dados em formato JSON.

# 2 Tecnologias Utilizadas

- **Java 17**
- **Maven**
- **Swing (GUI)**
- **Jackson** (serialização JSON)
- **JUnit 5** (testes automatizados)

# 3 Arquitetura do Sistema

O sistema segue uma arquitetura modular, organizada em pacotes com responsabilidades bem definidas:

- `usuario`: entidades do domínio de usuários;
- `sistema`: núcleo de regras de negócio;
- `view`: interface gráfica (GUI);
- `excessoes`: tratamento centralizado de erros;
- `utilitarios`: enums e classes auxiliares;
- `usuario.validacoes`: serviços de validação;
- `usuario.userDB`: persistência de dados.

# 4 Modelo de Usuários

## 4.1 Classe Abstrata Usuario

A classe `Usuario` representa a abstração base para todos os tipos de usuários do sistema. Ela define atributos comuns e validações obrigatórias no construtor.

- nome
- cpf
- senha
- email
- tipo (enum `TipoUsuario`)

O uso de anotações do **Jackson** permite polimorfismo na serialização JSON.

## 4.2 Paciente

Representa o paciente do hospital.

Principais funcionalidades:

- Agendamento e histórico de consultas;
- Prontuário médico;
- Controle de internação;
- Armazenamento de documentos médicos.

## 4.3 Medico

Representa o médico do hospital.

Atributos relevantes:

- CRM
- Especialidade
- Expediente semanal
- Duração padrão das consultas

O médico possui regras de disponibilidade baseadas em:

- Dia da semana;

- Horário de expediente;
- Consultas já agendadas;
- Estado ativo/inativo.

### 4.4 Secretaria

Usuário responsável pelo apoio administrativo, podendo:

- Cadastrar usuários;
- Agendar consultas;
- Interagir diretamente com o núcleo do hospital.

## 5 Sistema Hospital

A classe `Hospital` atua como o **núcleo do sistema**, sendo responsável por:

- Armazenar usuários;
- Gerenciar consultas;
- Validar regras de negócio;
- Emitir documentos médicos;
- Persistir dados em arquivo JSON.

Todas as regras críticas de agendamento passam por essa classe.

## 6 Consultas Médicas

A classe `Consulta` representa um atendimento médico agendado.

Cada consulta possui:

- Paciente e médico associados;
- Data e horário;
- Especialidade;
- Prontuário (após atendimento).

O sistema identifica automaticamente consultas já realizadas comparando a data atual.

# 7 Prontuário Médico

O prontuário armazena informações clínicas do paciente, como:

- Doença;
- Status clínico;
- Histórico de atendimento.

Está diretamente vinculado à consulta.

# 8 Documentos Médicos

O sistema permite a emissão de:

- Receita médica;
- Atestado;
- Exames.

Todos os documentos herdam de `DocumentoMedico` e geram conteúdo textual padronizado.

# 9 Interface Gráfica

A interface foi desenvolvida com **Swing**, separando responsabilidades entre telas e eventos.

Principais telas:

- TelaLogin
- TelaCadastro
- TelaMedico
- TelaPaciente
- TelaRecepcionista

Cada ação do usuário é tratada por classes do pacote `view.eventos`.

# 10 Persistência de Dados

A persistência é realizada via arquivos JSON utilizando a biblioteca **Jackson**.

A classe `RepositorioDeUsuario` é responsável por:

- Carregar usuários do arquivo;
- Salvar alterações;
- Buscar usuários por CPF.

# 11 Tratamento de Exceções

O sistema possui exceções personalizadas, como:

- UsuarioInexistente
- UsuarioJaExistente
- SenhaIncorretaException
- DataIndisponivel
- ProfissionalIndisponivel

Isso garante mensagens claras e controle adequado de erros.

# 12 Testes Automatizados

Os testes foram desenvolvidos com **JUnit 5**.

Cobrem:

- Criação de usuários;
- Validação de login;
- Disponibilidade de médicos;
- Agendamento de consultas.

Os testes garantem confiabilidade e manutenção segura do sistema.

# 13 Execução do Projeto

Para executar o projeto:

```
mvn clean package
mvn exec:java
```

O ponto de entrada do sistema é a classe `Main`.