Universidade de Brasília

RELATÓRIO DE IMPLEMENTAÇÃO DE SISTEMA DE GERENCIAMENTO DE CLÍNICA MÉDICA

Alunos

Alberto Côrtes Cavalcante - 232014610

Caio Rocha de Oliveira - 232001371

João Guilherme - 232014039

Lucas Machado Peres Ricarte - 232014093

Professor: André Luiz Peron Martins Lanna

Disciplina: Orientação a Objetos

BRASÍLIA

2025

Universidade de Brasília

OBJETIVOS

O objetivo deste trabalho é desenvolver um sistema de gerenciamento de clínica médica em Java, aplicando os conceitos fundamentais de orientação a objetos, como modularidade, encapsulamento, herança, polimorfismo e tratamento de exceções personalizadas. O sistema deve permitir o cadastro e gerenciamento de pacientes e médicos, agendamento de consultas, prescrição de exames e medicamentos, além da gestão de pagamentos. O projeto tem como finalidade desenvolver habilidades práticas de programação orientada a objetos com base no conteúdo aprendido ao longo do semestre.

INTRODUÇÃO

A gestão de clínicas médicas envolve uma série de processos que usam programação orientada a objetos, por meio de cadastro de pacientes e médicos até o agendamento de consultas e a prescrição de exames e medicamentos. Neste contexto, a utilização de Java aliada aos conceitos de orientação a objetos, permite a criação de um sistema que pode ser de fácil implementação. Este trabalho propõe a implementação de um sistema que atenda a esses requisitos, utilizando práticas de desenvolvimento que garantam a qualidade e a eficácia do software.

METODOLOGIA

O desenvolvimento do sistema de gerenciamento de clínica médica foi uma experiência prática que permitiu aplicar, os conceitos de orientação a objetos estudados durante o curso. Utilizamos a linguagem Java (versão SDK 17) para construir um sistema modular, encapsulado e funcional, que atende aos requisitos funcionais e técnicos propostos no enunciado do projeto. A seguir, descrevo detalhadamente a metodologia adotada, as decisões tomadas durante o desenvolvimento e as justificativas para as escolhas feitas, com base no código que implementamos em grupo.

Universidade de Brasília

SUMÁRIO

1 REPOSITÓRIO DO PROJETO 3

2. DIAGRAMA UML 3

3. ANÁLISE DO PROJETO 4

4. IMPLEMENTAÇÃO 4

5. MODULARIDADE E ENCAPSULAMENTO 5

6. HERANÇA 5

7. POLIMORFISMO 5

8. TESTES E DOCUMENTAÇÃO 6

9. CONCLUSÃO 6

10. REFERÊNCIAS BIBLIOGRÁFICAS 6

Universidade de Brasília

1. REPOSITÓRIO DO PROJETO

O código-fonte completo do sistema pode ser acessado no seguinte repositório: https://github.com/Dexmachi/Trabalho-OO

- **Alberto Cavalcante: https://github.com/oalbertocavalcant**
- **Caio Rocha: https://github.com/Dexmachi/**
- **João Guilherme: https://github.com/JoaoGuilherme14**
- **Lucas Ricarte: https://github.com/Lucas-Ricarte**

2. DIAGRAMA UML

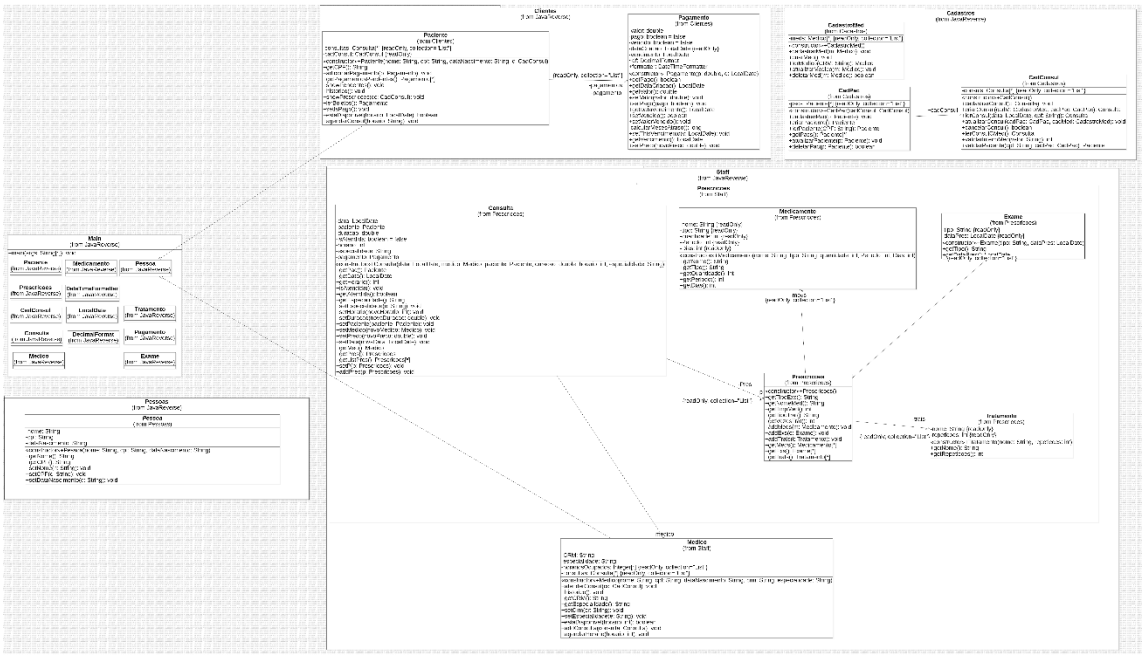

Figura 1: Diagrama em UML

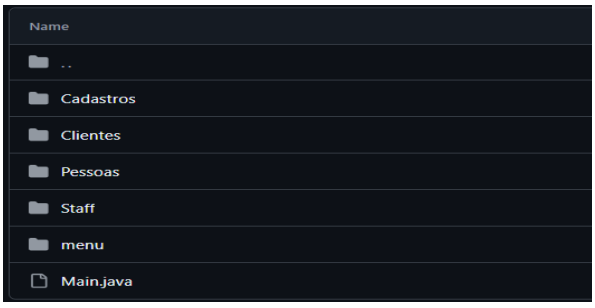

Figura 2: Estrutura de pacotes

3. ANÁLISE DO PROJETO

A primeira etapa do projeto consistiu em analisar os requisitos funcionais e técnicos para entender as necessidades do sistema. Com base nessa análise, identificamos as principais entidades que compõem o domínio da aplicação: Paciente, Médico, Consulta, Exame, Medicamento, Tratamento, Pagamento e Prescrição. Essas entidades foram modeladas como classes, e suas relações foram representadas em um diagrama de classes UML, que serviu como guia para a implementação. O diagrama UML para representar visualmente as classes, seus atributos, métodos e as associações entre elas

A escolha de utilizar um diagrama de classes foi fundamental para visualizar as associações entre os objetos e garantir que todas as relações necessárias fossem implementadas corretamente. Por exemplo, a classe Consulta está associada a um Paciente e a um Médico, além de possuir uma lista de Exames e Medicamentos prescritos. Essa modelagem permitiu que o sistema fosse construído, com cada classe tendo responsabilidades bem definidas.

As principais classes identificadas foram:

Paciente: Representa os pacientes da clínica, com atributos como nome, CPF, data de nascimento e histórico médico.

Médico: Representa os médicos da clínica, com atributos como nome, CPF, CRM, especialidade e histórico de consultas.

Consulta: Representa as consultas agendadas, com atributos como data, horário, duração, status, paciente associado, médico responsável, lista de exames e medicamentos prescritos.

Exame: Representa os exames prescritos, com atributos como tipo, data de prescrição, data de realização, resultado e custo.

Medicamento: Representa os medicamentos prescritos, com atributos como nome, tipo, quantidade, período e dias de tratamento.

Pagamento: Representa os pagamentos associados às consultas e exames, com atributos como valor, data de criação, data de vencimento e status de pagamento.

4. IMPLEMENTAÇÃO

A implementação do sistema foi realizada com foco em modularidade, encapsulamento, herança e polimorfismo, seguindo os princípios da orientação a objetos. O código foi organizado em pacotes para separar as responsabilidades e facilitar a manutenção. Por exemplo, o pacote Cadastros contém as classes responsáveis pelo cadastro de pacientes, médicos, consultas e prescrições, enquanto o pacote Clientes contém as classes relacionadas aos pacientes e pagamentos.

5. MODULARIDADE E ENCAPSULAMENTO

O encapsulamento foi uma das práticas mais importantes adotadas durante o desenvolvimento. Todos os atributos das classes foram definidos como privados, e o acesso a eles foi controlado através de métodos públicos (getters e setters). Isso garantiu que os dados só pudessem ser modificados de forma controlada, seguindo as regras de negócio estabelecidas. Por exemplo, o histórico médico de um paciente só pode ser modificado através de métodos da classe Paciente, nunca diretamente. Essa abordagem protegeu a integridade dos dados e facilitou a manutenção do código.

A modularidade foi alcançada através da separação das responsabilidades em diferentes classes e pacotes. Por exemplo, a classe CadConsul é responsável por gerenciar o cadastro de consultas, enquanto a classe CadPac gerencia o cadastro de pacientes. Essa divisão de responsabilidades permitiu que o código fosse mais organizado e de fácil compreensão:

Cadastros: Contém as classes responsáveis pelo cadastro de pacientes, médicos, consultas e prescrições.

Clientes: Contém as classes relacionadas aos pacientes e pagamentos.

Staff: Contém as classes relacionadas aos médicos e prescrições.

Menu: Contém a classe responsável pela interface de interação com o usuário.

Pessoas: Contém a classe base Pessoa, que é herdada por Paciente e Médico.

Todos os atributos das classes foram definidos como privados, e o acesso a eles foi controlado através de métodos públicos (getters e setters). Isso garantiu que os dados só pudessem ser modificados de forma controlada, seguindo as regras de negócio estabelecidas.

6. HERANÇA

A herança foi utilizada para reutilizar código e estabelecer uma hierarquia entre as classes. A classe Pessoa foi criada como uma classe base, contendo atributos comuns como nome, CPF e data de nascimento. As classes Paciente e Médico herdam de Pessoa, aproveitando esses atributos e métodos comuns.

7. POLIMORFISMO

O polimorfismo foi aplicado através da sobrescrita de métodos e do uso de generics. Por exemplo, a classe Consulta possui métodos que podem ser sobrescritos para adicionar comportamentos específicos. Além disso, as listas de exames e medicamentos foram

implementadas utilizando generics, permitindo a manipulação de diferentes tipos de objetos de forma genérica.

8. TESTES E DOCUMENTAÇÃO

Durante a implementação, foram realizados testes manuais para validar o funcionamento do sistema. Cada funcionalidade foi testada individualmente, garantindo que os requisitos fossem atendidos. Além disso, foram realizados testes de integração para verificar a interação entre as diferentes classes e pacotes.

A documentação do sistema foi realizada através de comentários no código, explicando o propósito de cada classe e método. Além disso, foi criado um relatório em PDF contendo o diagrama de classes UML, a explicação das associações, heranças e polimorfismos aplicados, e a justificativa para as exceções customizadas.

9. CONCLUSÃO

O desenvolvimento do sistema de gerenciamento de clínica médica foi uma oportunidade para aplicar os conceitos de orientação a objetos de forma prática. A modularidade, encapsulamento, herança e polimorfismo foram essenciais para criar um sistema organizado, de fácil manutenção e expansão. O tratamento de exceções personalizadas garantiu que o sistema fosse robusto e capaz de lidar com situações de erro de forma adequada. O diagrama de classes UML foi uma ferramenta valiosa para a compreensão e comunicação do design do sistema.

Este projeto demonstrou a importância de uma abordagem estruturada e orientada a objetos para o desenvolvimento de sistemas complexos, garantindo que os requisitos funcionais e técnicos fossem atendidos de forma eficiente.

10. REFERÊNCIAS BIBLIOGRÁFICAS

StarUML Documentation. Disponível em: https://staruml.io/