

UNIVERSIDADE FEDERAL DE OURO PRETO - UFOP INSTITUTO DE CIÊNCIAS EXATAS E BIOLÓGICAS - ICEB DEPARTAMENTO DE COMPUTAÇÃO - DECOM 

ANANDA MENDES SOUZA

RÔMULO DE OLIVEIRA CARNEIRO

**SISTEMA DE GERENCIAMENTO PARA CLÍNICA ODONTOLÓGICA**

RELATÓRIO TRABALHO PRÁTICO I

OURO PRETO

2021

ANANDA MENDES SOUZA

RÔMULO DE OLIVEIRA CARNEIRO

**SISTEMA DE GERENCIAMENTO PARA CLÍNICA ODONTOLÓGICA**

RELATÓRIO TRABALHO PRÁTICO I

Relatório apresentado à disciplina

Programação orientada a objeto da matriz

curricular do Curso de Ciência da

Computação da Universidade Federal de

Ouro Preto, a ser utilizado como parte das

exigências do Trabalho Prático I – Sistema

de Gerenciamento para Clínica

Odontológica.

OURO PRETO

2021

**SUMÁRIO**

**INTRODUÇÃO 4 DESCRIÇÃO DAARQUITETURA E IMPLEMENTAÇÃO 4 DIAGRAMA UML 6 DIFICULDADES DO PROJETO 7 RECURSOS DA LINGUAGEM 7 INSTRUÇÕES E COMPILAÇÃO 8 CONCLUSÃO 9**

**1. INTRODUÇÃO**

Programação Orientada a Objetos é uma metodologia de desenvolvimento de software, cujo principal objetivo é facilitar a modelagem e desenvolvimento de sistemas, através da interação entre objetos. Sua principal vantagem é a proximidade com a forma que os seres humanos visualizam e entendem o mundo ao seu redor.

Sendo assim, a proposta deste trabalho consiste em criar um sistema de gerenciamento para uma Clínica Odontológica, considerando os conhecimentos de Programação Orientada a Objetos adquiridos durante as aulas ministradas na matéria.

A princípio foi criado o diagrama de UML para nortear a implementação do código, através da criação de classes e suas respectivas relações, sendo elas: Clínica, Pessoa, Cliente, Funcionário, Administrador, Assistente Administrativo, Usuário Geral, Dentista, Recepcionista, Agenda, Contas, Consulta. Posteriormente a isso houve a implementação do código.

**2. DESCRIÇÃO DAARQUITETURA E IMPLEMENTAÇÃO**

O código foi implementado simulando o funcionamento de uma Clínica Odontológica, onde cada classe representava um agente da aplicação, sendo elas:

● **Clínica**: Classe responsável por vincular todas as outras a uma empresa em questão.

● **Pessoa**: Classe base de todas as pessoas envolvidas no sistema da Clínica, armazenando dados básicos de cadastro, sendo eles nome e idade. Não foram considerados dados cadastrais mais complexos no intuito de a princípio dar atenção às técnicas de POO empregadas, dados como CPF e RG serão incrementados no trabalho com interface gráfica.

● **Cliente:** Classe dos pacientes que irão marcar uma consulta, possuindo assim dados referentes ao pagamento da mesma.

● **Funcionário:** Classe base de todos os trabalhadores da Clínica, herda os dados da Classe Pessoa e fornece funções de cadastro dos funcionários.

4

● **Administrador:** Classe derivada de Funcionário, com o diferencial de possuir um login e senha, responsável por administrar todo o funcionamento do sistema.

● **Assistente Administrativo:** Classe derivada de Administrador, possui permissão para acessar a Agenda e folha de ponto, receber consultas e fazer pagamento de contas.

● **Usuário Geral:** Classe que possui acesso único e exclusivo aos dados das Agendas disponíveis, podendo acessá-las ou editá-las.

● **Dentista:** Classe derivada de Funcionário, responsável por ser vinculado à Agenda e possuir um Assistente e Pacientes.

● **Recepcionista:** Classe derivada de Funcionário, tratado com usuário geral da aplicação.

● **Agenda:** Classe responsável por vincular os horários de atendimento de Dentistas a pacientes.

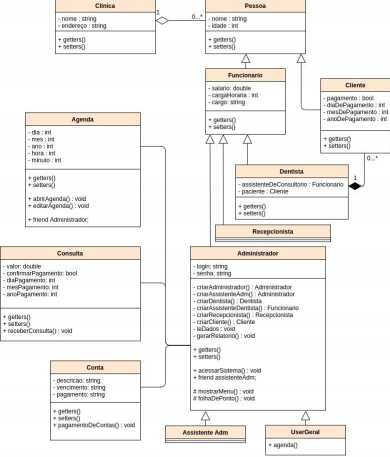
● **Contas:** Classe que armazena dados de vencimento e pagamento das contas da clínica, podendo ser acessada apenas por administradores e seus assistentes.

● **Consulta:** Classe que armazena o valor e data de pagamento das consultas marcadas, podendo ser acessada apenas por administradores e seus assistentes.

5

3. **DIAGRAMA UML**

O diagrama foi construído utilizando o software draw.io, que contém funções específicas para diagramação proposta no trabalho.

Obs: Em caso de dificuldade para visualizar terá um anexo do diagrama em pdf.

6

4. **DIFICULDADES DO PROJETO**

A principal dificuldade encontrada neste projeto foi entender a dinâmica da clínica para poder criar o diagrama UML, visto que não sabíamos quais classes deveriam existir e qual seria a relação entre elas. Para solucioná-lo foi necessário marcar uma reunião com o professor e tirar as dúvidas.

Outra dificuldade foi em relação a manipulação de funções de acordo com o tipo de usuário, por exemplo, como separar as ações que apenas um administrador pode fazer que o resto dos usuários não podem. A solução encontrada para esse problema foi verificar na entrada se o usuário seria administrador ou apenas usuário geral e colocar as funções especiais como privadas na classe Administrador.

5. **RECURSOS DA LINGUAGEM**

Para tornar o código mais funcional, compreensivo e compacto foram utilizados algumas funcionalidades disponíveis em algumas bibliotecas, sendo elas:

● **Bibliotecas:**

**<list> :** A biblioteca list foi utilizada para armazenar uma estrutura de dados de maneira dinâmica, visto que não há um limite inicial a ser traçado para seu tamanho, a exemplo o número de funcionários da clínica.

**<unistd> :** A biblioteca unistd foi utilizada para aplicar um temporizador na aplicação para adicionar um delay entre comandos para simular um sistema

**<time.h> :** A biblioteca time foi utilizada para pegar a data atual do sistema com a ajuda da struct ™ que existe nessa biblioteca. Usamos essa funcionalidade nos momentos de confirmar pagamento sendo a data em que o usuário confirmou.

**● Recursos externos:**

**Makefile :** Utilizou-se do Makefile para facilitar a compilação do código, isso porque devido a grande quantidade de arquivos seria custosa a compilação de todos eles a cada teste, ademais será utilizado arquivos de texto para armazenar algumas informações, como Folha de Ponto e Salário, dentre outras.

7

**6. INSTRUÇÕES E COMPILAÇÃO**

Para execução do trabalho no terminal foi utilizado os recursos do Makefile como foi citado anteriormente, dito isso basta digitar “make clean” para limpar os resíduos dos arquivos “.o” gerados e em seguida executar o comando “make all” para compilar o código passando por toda a estrutura listada no próprio Makefile e por fim inserir o comando “make run” para executar o programa.

Depois de iniciado, o programa solicita que o usuário especifique qual tipo de usuário ele é, usuário geral ou administrador. Escolhendo a primeira opção, o mesmo só terá acesso às funções relacionadas à agenda (abrir e editar). Em caso contrário, possuirá acesso a todas as funções do sistema, desde a agenda até criar novos usuários. Em ambos os casos o login do sistema é padrão, sendo **login:** admin e **senha:** admin.

O sistema é composto por vários menus de ações, todos são bem intuitivos e possuem comandos bem descritivos para que não gere dúvidas ao usuário que esteja interagindo com eles. Segue alguns exemplos:

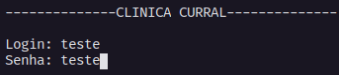
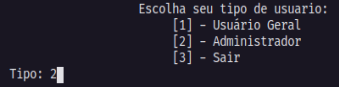
Menu inicial que especifica o tipo de usuário.

Menu de usuário geral.

8

Menu de Administrador.

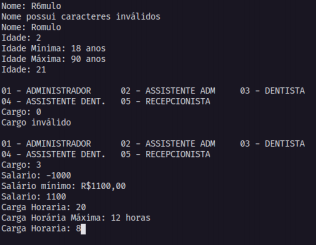
Para realizar ações relacionadas a entrada de dados foi criado validações para que o código não quebrasse, a exemplo do acesso ao sistema por administradores e criações de novos funcionários/clientes.





Teste de entrada de dados de um administrador inválido.

9



Teste de entrada de dados durante a criação de um funcionário

**7. CONCLUSÃO**

Pode-se inferir, portanto, que este trabalho se trata de uma junção de diversos conhecimentos de programação orientado a objetos adquiridos, aplicados a um sistema de gerenciamento de uma clínica odontológica.

Por meio dele podemos explorar conceitos de classes, herança, associação, polimorfismo, abstração de dados e encapsulamento de dados.

10