

# **RELATÓRIO** **Programação de Sistemas de Informação**

# **CURSO TÉCNICO DE PROGRAMAÇÃO E GESTÃO DE SISTEMAS INFORMÁTICOS**

Professor : Breno Sousa

Alunos: Daniel Alexandre, Marcos Catito

Nº Alunos:

26/10/2025

O relatório encontra-se em condições para ser apresentado \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ciclo de Formação 2023/2026  
Ano Letivo 2025/2026

*INDICE*

[AGRADECIMENTOS 1](#_Toc212396292)

[INTRODUÇÃO 1](#_Toc212396293)

[CAPÍTULO I –- Caracterização do Trabalho 2](#_Toc212396294)

[Caraterização do problema 2](#_Toc212396295)

[Linguagem de programação utilizada 3](#_Toc212396296)

[Capítulo II – O Desenvolvimento do Sistema do hospital 4](#_Toc212396297)

[Sobre as Classes 4](#_Toc212396298)

[Utilização de Módulos 12](#_Toc212396299)

[Execução do sistema 15](#_Toc212396300)

[Capítulo III - CONCLUSÃO 17](#_Toc212396301)

*INDICE DE FIGURAS*

[Fig 1 – Rascunho das classes 2](#_Toc212396029)

[Fig 2 – Rascunho das classes segunda parte 3](#_Toc212396030)

[Fig 3 – Logo da linguagem de programação 3](#_Toc212396031)

[Fig 4 – imagem da classe pessoa com método classe abstrata 4](#_Toc212396032)

[Fig 5 – Getter e o setter do nome 4](#_Toc212396033)

[Fig 6 – Getter e o setter para idade 5](#_Toc212396034)

[Fig 7 – Getter e o setter do numero\_utente 5](#_Toc212396035)

[Fig 8 – Funções do 6](#_Toc212396036)

[Fig 9 – Mensagens para serem exibidas 6](#_Toc212396037)

[Fig 10 – Getter e o setter do salário 6](#_Toc212396038)

[Fig 11 – As funções para as mensagens 7](#_Toc212396039)

[Fig 12 – Mensagens da classe funcionários 7](#_Toc212396040)

[Fig 13 – Funções da classe médico 8](#_Toc212396041)

[Fig 14 – Getter e o setter do turno 8](#_Toc212396042)

[Fig 15 – Pagamento com base no turno 9](#_Toc212396043)

[Fig 16 – Getter e o setter para o setor 9](#_Toc212396044)

[Fig 17 – Registo da hora e cálculo do pagamento 9](#_Toc212396045)

[Fig 18 – Pagamento bónus do enfermeiro chefe 10](#_Toc212396046)

[Fig 19 – Classe sala 11](#_Toc212396047)

[Fig 20 – Getter e o setter dos atributos da classe slas 11](#_Toc212396048)

[Fig 21 – Funções da sala cirurgia 11](#_Toc212396049)

[Fig 22 – Getter e o setter do atributo medico responsável 12](#_Toc212396050)

[Fig 23 – Funções da classe sala consulta 12](#_Toc212396051)

[Fig 24 – Lista dos arquivos 13](#_Toc212396052)

[Fig 25 – Import dos arquivos para o menu 13](#_Toc212396053)

[Fig 26 – Listas 14](#_Toc212396054)

[Fig 27 – Menu de seleção 14](#_Toc212396055)

[Fig 28 – Opção para inserir o paciente 14](#_Toc212396056)

[Fig 29 – Menu do sistema 15](#_Toc212396057)

[Fig 30 – Inserção dos dados do paciente 15](#_Toc212396058)

[Fig 31 – Apresentação dos pacientes 15](#_Toc212396059)

[Fig 32 – Paciente a ser atendido 16](#_Toc212396060)

[Fig 33 – histórico do paciente 16](#_Toc212396061)

AGRADECIMENTOS

Gostaríamos de agradecer ao Professor por ter nos dado este desafio, aos nossos país pelo o apoio dado e aos nossos colegas que auxiliaram em algumas dúvidas.

INTRODUÇÃO

Este relatório, relata o desenvolvimento de um trabalho dado no período de aulas, fazendo a caraterização e daquilo que era o trabalho a ser feito ao longo desse período, contendo a explicação do devido código do sistema, a linguagem de programação que foi utilizada e a exibição da execução de alguns comandos do sistema a serem apresentadas.

No primeiro capítulo faz a caraterização do trabalho

* Caracteriza-se o problema;
* Linguagem de programação utilizada.

No segundo, descreve o código do sistema

* Explicação das classes;
* Explicação da utilização dos módulos;
* Execução de alguns comandos.

CAPÍTULO I –- Caracterização do Trabalho

Neste primeiro capítulo, iremos caraterizar o trabalho que foi desenvolvido ao longo do período de aulas na resolução de um problema proposto.

**Em que consistia o problema?**

O seguinte problema consistia no desenvolvimento de um sistema para um hospital, onde o sistema cumpriria os métodos de atendimento ao paciente, lista dos pacientes, ter o registo dos funcionários e por fim fazer a gestão das salas.

Antes de caraterizar o problema, foram definidos alguns parâmetros para o desenvolvimento do trabalho, sendo definido 7 horas para o desenvolvimento do trabalho e o trabalho seria cumprido em duplas.

## Caraterização do problema

Com base no problema, veio a seguinte pergunta em nós, como seria um sistema para um hospital e que tipo de métodos para executá-los seriam necessários?

Com algum auxílio disponibilizado pelo professor, no nosso problema, tínhamos a estrutura construída de como seria a implementação em código. Esta estrutura era basicamente um mapa de como o sistema seria desenvolvido e como funcionaria.

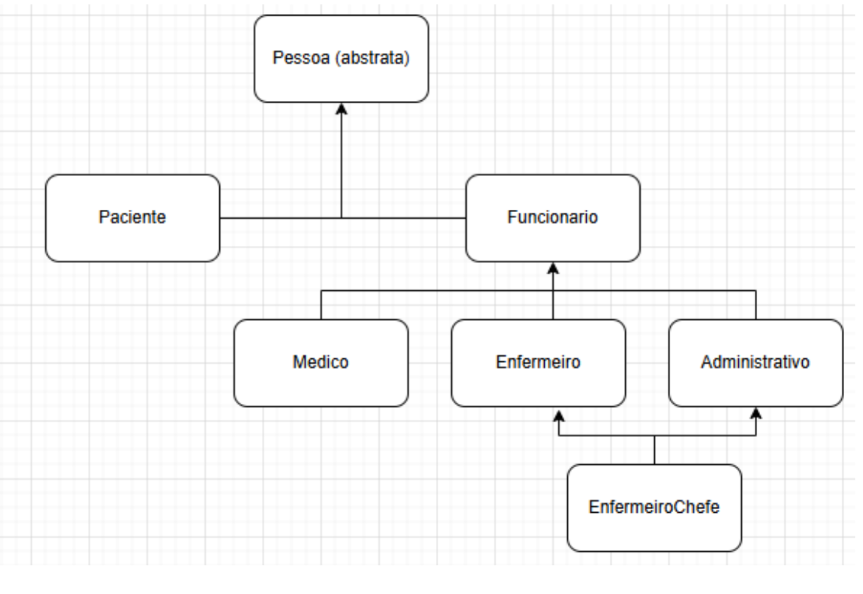


Fig – Rascunho das classes

Na imagem seguinte, temos o caminho de como as classes seriam desenvolvidas e de como serviriam as suas heranças. A seguir a este rascunho há outro, onde temos o esquema das classes da sala, sala consulta e a sala cirurgia.

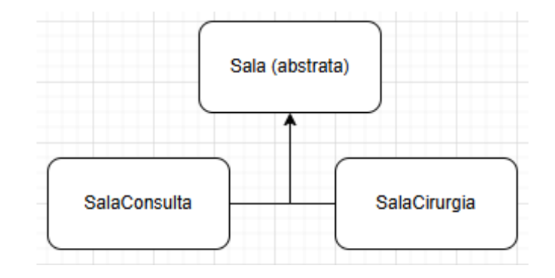


Fig – Rascunho das classes segunda parte

Depois dos esquemas, tínhamos algumas sugestões dadas para o desenvolvimento do sistema.

## Linguagem de programação utilizada

Lembrando que todo desenvolvimento foi feito através da linguagem de

programação Python. Para melhor compreender o desenvolvimento, falaremos um pouco da linguagem de programação Python.



Fig – Logo da linguagem de programação

Python é uma ampla linguagem de programação conhecida por sua facilidade de aprendizagem, sintaxe clara e concisa, e sua ampla variedade de aplicações, incluindo desenvolvimento web, data science, machine learning, automação de tarefas, entre outros.

Assim concluindo a explicação de como o trabalho foi caraterizado, daremos entrada ao próximo capítulo onde iremos descrever a parte do desenvolvimento em código.

Capítulo II – O Desenvolvimento do Sistema do hospital

Depois de compreendido o que era o trabalho e a sua caraterização, neste segundo capítulo, iremos descrever o código por trás da execução do sistema do hospital.

## Sobre as Classes

Para começar, foi feito a utilização de múltiplas classes, sendo no total 11, das quais cada uma teve a sua devida função começando pela primeira delas:

Classe pessoa – a classe pessoa, é uma classe abstrata onde não tem uma função direta, ou seja, esta classe serviria apenas para herdar as suas funções a outras classes.

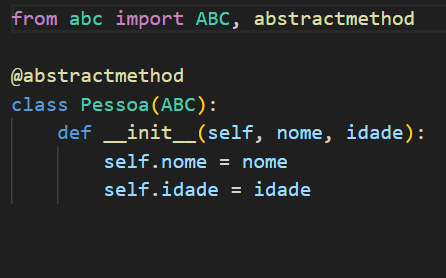
Pelo que se vê na imagem, primeiro é importado uma instrução do abc (abstract base classes) servindo como um decorador, onde está função obriga que as classes que herdarem os seus atributos, terão de os definir.

Fig – imagem da classe pessoa com método classe abstrata

Seguidamente, faz-se as classes paciente e funcionário, estás classes serão as classes que herdam os atributos da classe pessoa, sendo as classes obrigadas a definir o nome e a idade, atributos que fazem parte da classe pessoa.

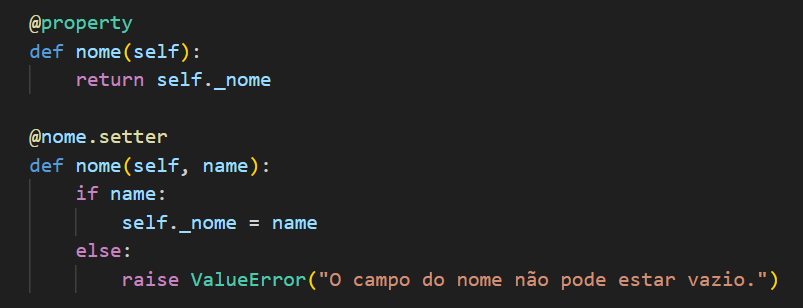


Fig – Getter e o setter do nome

Como pode se comprovar na imagem, para definir o nome do paciente ou do funcionário, primeiro é feito a instrução getter para ter o acesso ao atributo nome e em seguida o setter para definir o atributo.

Dentro do getter do nome faz-se uma função onde retorna o atributo e no setter aplica-se uma função onde se o nome não for atribuído, o utilizador recebe mensagem de erro.

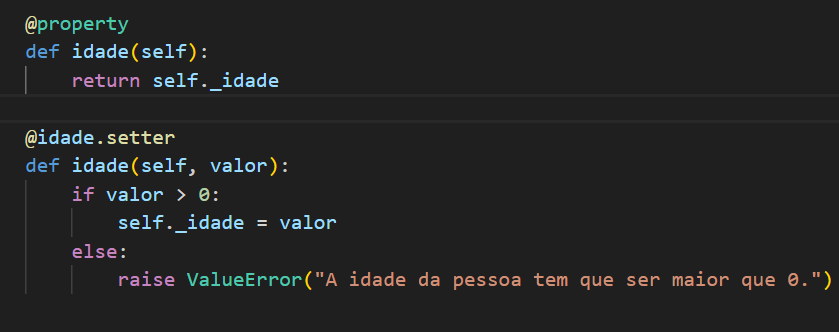


Fig – Getter e o setter para idade

No atributo idade, faz-se a mesma coisa que o atributo nome, só que no seu setter a função feita, serve para impedir que o utilizador não atribuí números menores que zero, pois a idade não pode ser negativa.

Agora que os atributos nome e idade da classe pessoa foram definidos para serem atribuídos, serão definidos dois atributos para classe paciente sendo o número de utentes e o atributo histórico que servirá como uma lista.

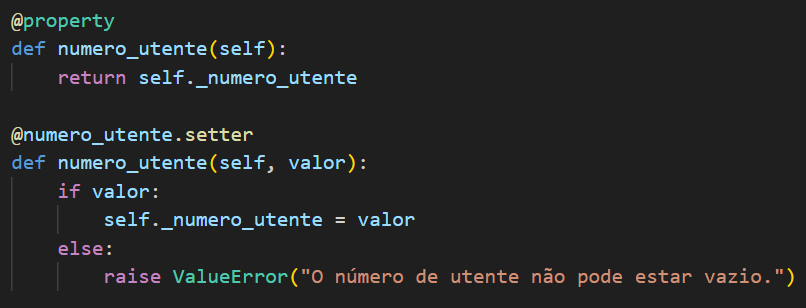


Fig – Getter e o setter do numero\_utente

Depois de ter o atributo utentes, faz-se um getter para fazer o retorno do atributo e o setter que tem a função de atribuir o valor do utente, caso seja definido um espaço vazio, o sistema dará uma mensagem de erro.

Seguidamente, utilizando a lista (o atributo histórico) e criam-se as seguintes funções, onde a função adicionar\_registo e a função mostrar\_historico são essências para mostrar o histórico do registo de cada paciente.

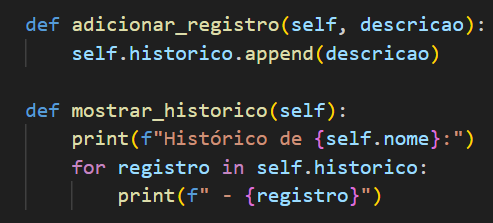


Fig – Funções do

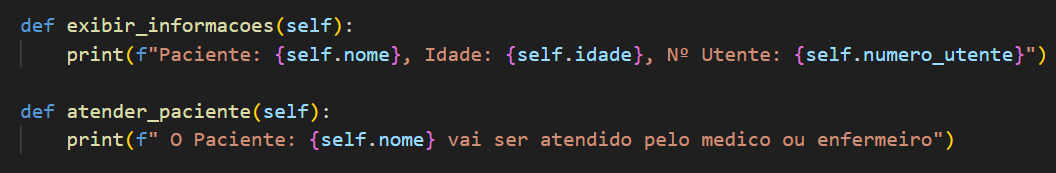


Fig – Mensagens para serem exibidas

Após as funções do histórico, temos mais duas funções que exibiram as informações do paciente e o atendimento do paciente.

Agora, saindo da classe paciente, partiremos para classe funcionário que também herda os atributos nome e idade da classe pessoa, contendo os atributos salário e o cargo.

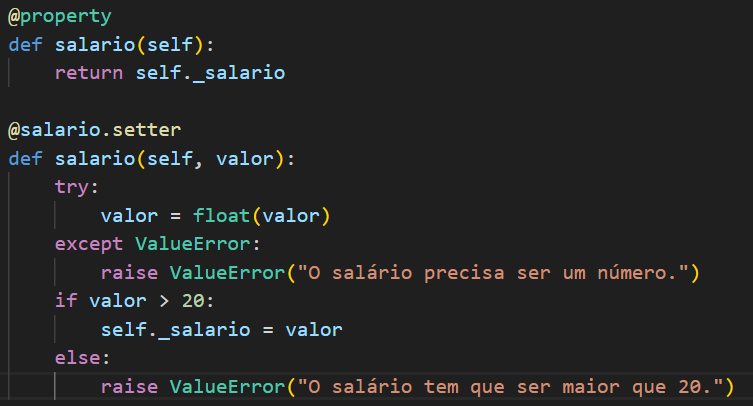


Fig – Getter e o setter do salário

Aqui tem-se o getter e o setter do salário, onde o valor do salário é definido como tipo Float, ou seja, onde aceita apenas números com vírgula.

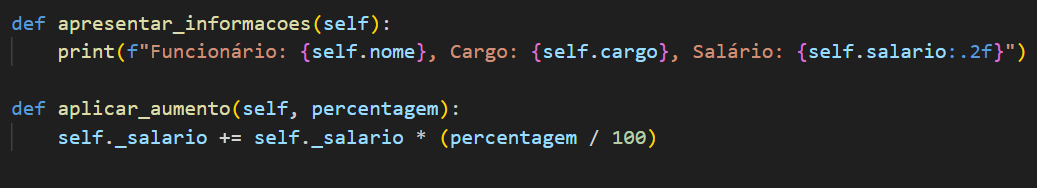


Fig – As funções para as mensagens

E em seguida as funções para exibirem as mensagens da classe dos funcionários.

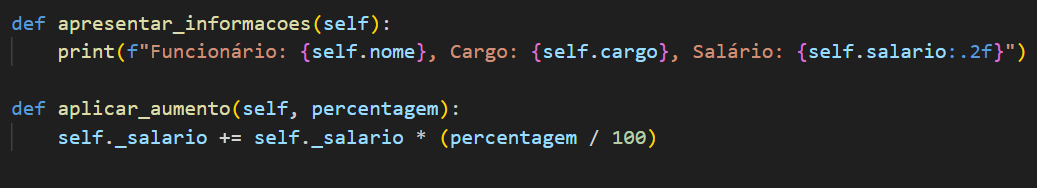


Fig – Mensagens da classe funcionários

Agora que as classes paciente e funcionários estão definidas e herdam atributos da classe abstrata pessoa, teremos uma herança múltipla, onde as classes como médico, enfermeiro e administrativo herdam os atributos da classe funcionário, isso significa que estas classes também herdam os atributos da classe abstrata, mas com um, porém, a função dos atributos da classe abstrata mantém-se só executadas na classe funcionário.

Começando pela classe médico, é atribuído a especialidade e uma lista paciente que serviria para reconhecer o número de pacientes atendidos. Onde a lista do paciente é utilizada para

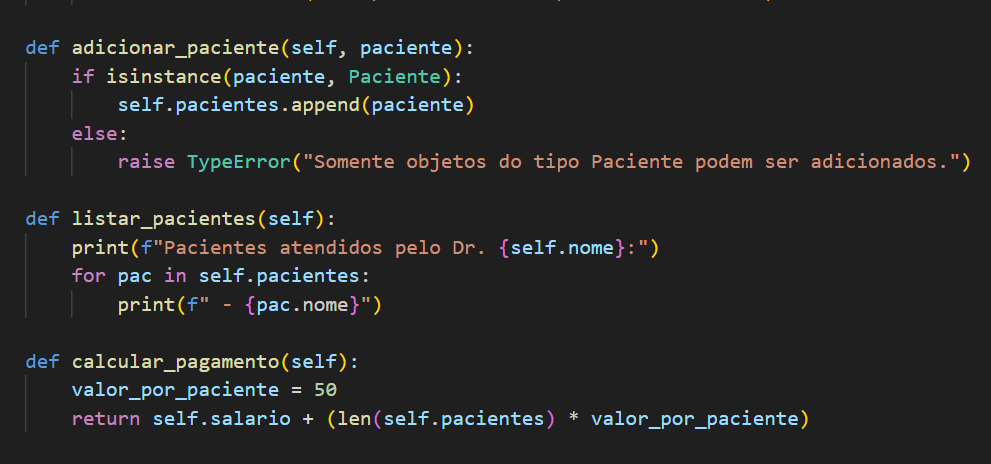


Fig – Funções da classe médico

Na seguinte imagem, comprova-se as funções que a classe médico cumpre, sendo a primeira (adicionar\_paciente) que serve para fazer adição de apenas o objeto paciente na lista pacientes, a segunda função (listar\_pacientes) serve para exibir os pacientes que foram atendidos pelo nome do doutor e a última função (calcular\_pagamento) faz o cálculo do salário.

Explicando agora a classe enfermeiro, está classe é similar há classe médico, mas contém algumas mudanças, tendo como atributo o turno e novamente uma lista de pacientes definida com as mesmas funções que da classe médico.

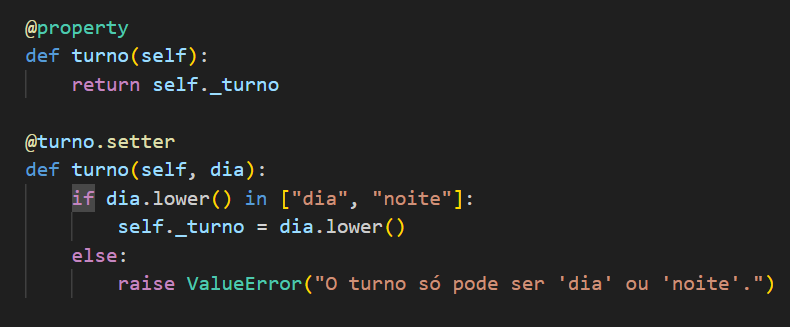


Fig – Getter e o setter do turno

Para o atributo turno, é feito o seu getter para o retorno do atributo e em seguida o setter para a sua atribuição entro o turno do dia e o da noite.

**Importante**: dia.lower é uma instrução utilizada apenas para transformar as letras em minúsculas para evitar comparações entre as maiúsculas. Depois verifica se o dia e a noite estão na lista.

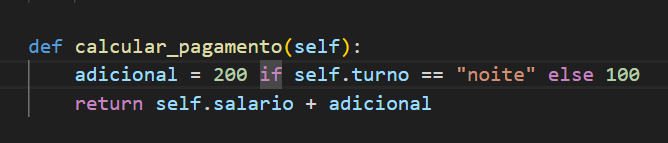


Fig – Pagamento com base no turno

Para a classe enfermeiro temos o tipo de pagamento baseado no seu turno, assim se o enfermeiro tiver um salário base, receberá um bónus.

Na classe administrativo, foram atribuídos setor e as horas de trabalho, tendo um getter e um setter para o atributo setor com a função de retornar e atribuir o setor de cada administrador.

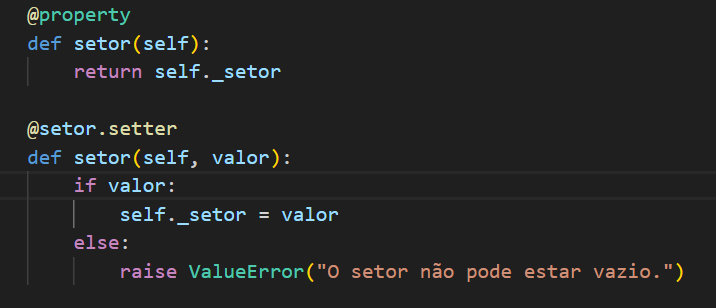


Fig – Getter e o setter para o setor

Em seguida cria-se duas funções para o atributo hora, onde a primeira função regista a hora e verifica se a hora não é menor que zero e a segunda função faz o cálculo do valor a ser pago ao administrador, sendo o valor pela hora cumprida do trabalho.

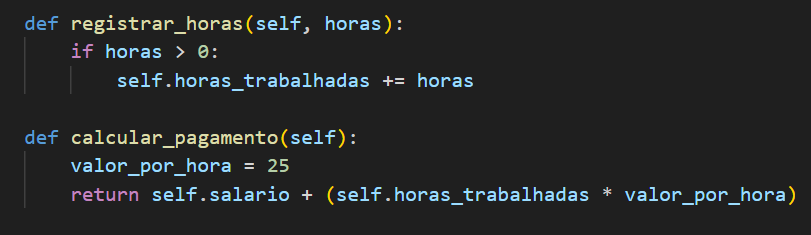


Fig – Registo da hora e cálculo do pagamento

Está classe talvez seja uma das classes mais confusas, mas explicaremos o porque da sua utilidade, por mais que a classe enfermeiro chefe aparenta ser uma classe igual a classe enfermeiro, está classe para além de possuir um cargo diferente, herda elementos da casse administrativos e da classe enfermeiro.

Mas então, qual é a diferença entre ambas classes (enfermeiro chefe e o enfermeiro)?

A resposta para tal pergunta é simples até, pois o enfermeiro chefe é como se fosse o coordenador dos enfermeiros, para além disso, a classe enfermeiro chefe tem um serviço administrativo por herdar elementos da classe administrativo.

Agora, no sistema, o enfermeiro chefe opera mais ou menos como o enfermeiro e o medico tendo o seu pagamento bónus acima dos outros funcionários.

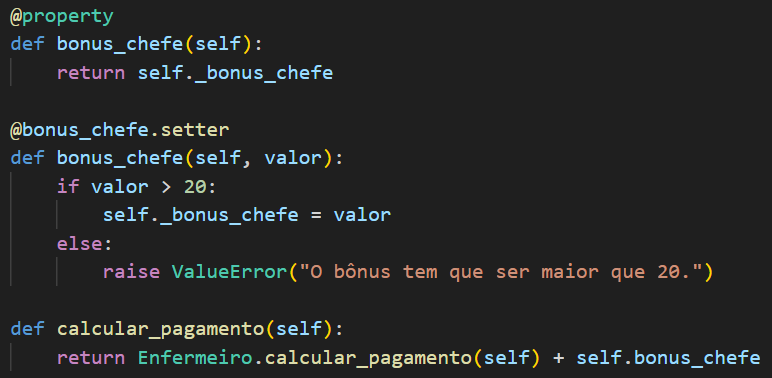


Fig – Pagamento bónus do enfermeiro chefe

Após as classes do paciente, funcionário, médico, enfermeiro, administrativo e o enfermeiro chefe parece tudo acabado, mas ainda não, pois, para o sistema funcionar muito bem, precisamos saber as salas também.

Para isso foram criadas três classes, sendo a classe sala, a classe sala cirurgia e a sala consulta. A classe sala seria a classe abstrata onde seus atributos serão herdados pelas outras classes.

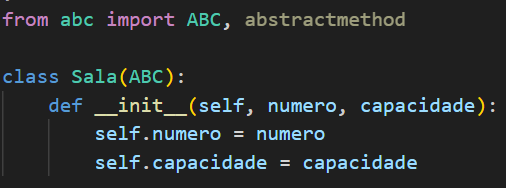


Fig – Classe sala

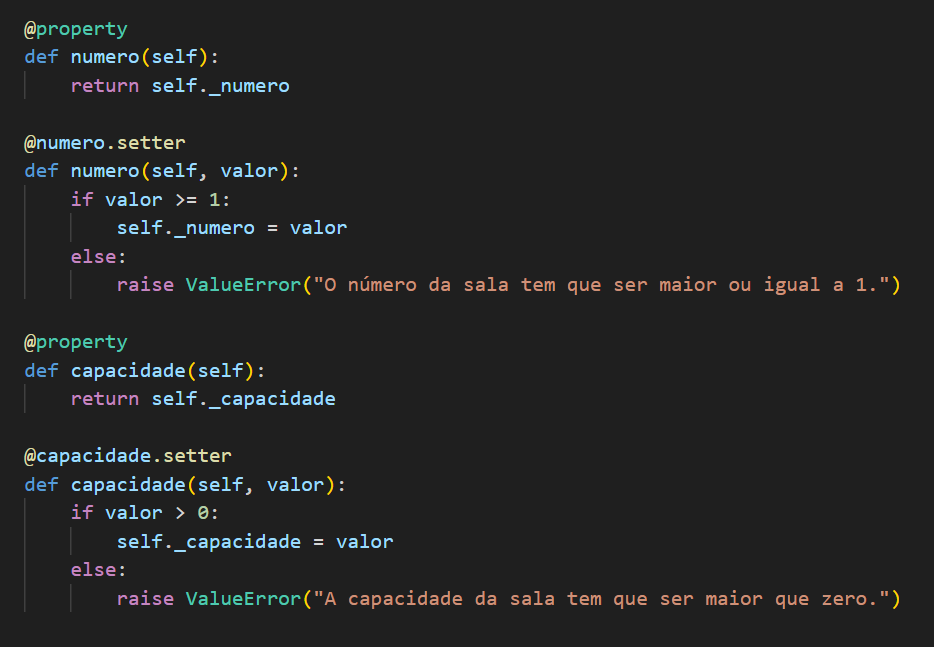


Fig – Getter e o setter dos atributos da classe slas

Em seguida na classe cirurgia e a sala consulta, é feito o getter e o setter para os atributos número (número da sala) e capacidade vindos da classe abstrata. Para além disso, é definido o atributo equipamentos.

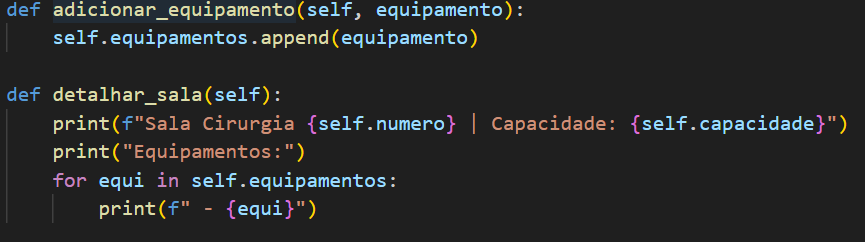


Fig – Funções da sala cirurgia

Na imagem, demonstra a função para adição dos equipamentos da sala e a outra função para exibir os detalhes da sala.

Por último, a classe consulta recebe a herança da classe sala e diferente da classe sala cirurgia, recebe os atributos de medico o que fica responsável e tem uma lista dos pacientes.

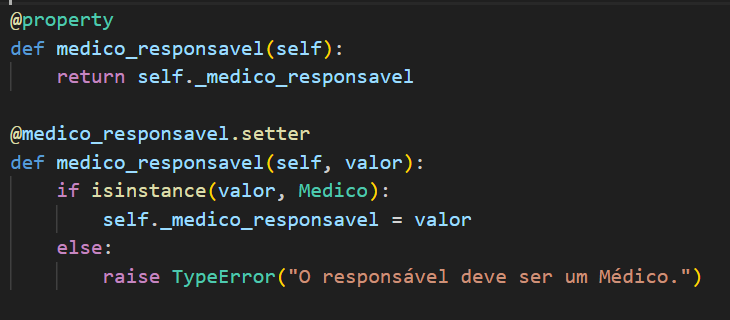


Fig – Getter e o setter do atributo medico responsável

Faz-se um getter e um setter para o atributo, seguidamente faz-se uma função de agendar a consulta, onde se verifica a capacidade d sala.

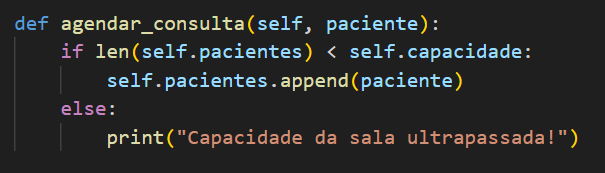


Fig – Funções da classe sala consulta

## Utilização de Módulos

Após a descrição das funções de cada classe, falaremos da utilização de módulos no sistema do hospital. Módulos em Python seriam como arquivos que podem ser importados, tendo como função no sistema do hospital organizar separadamente os arquivos de cada classe.

Neste sentido, os arquivos criados no sistema, fazem a utilização dos módulos para que as funções de cada arquivo sejam importadas para cada um deles.

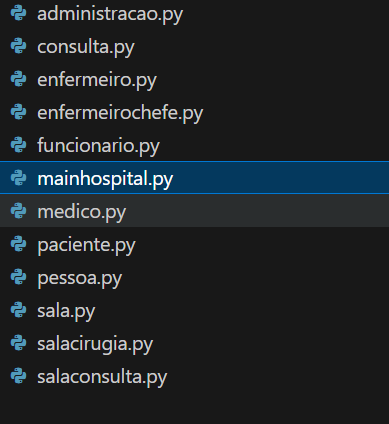


Fig – Lista dos arquivos

Para fazer a importação de cada arquivo, utiliza-se a linha do comando from nome do arquivo import nome da classe.

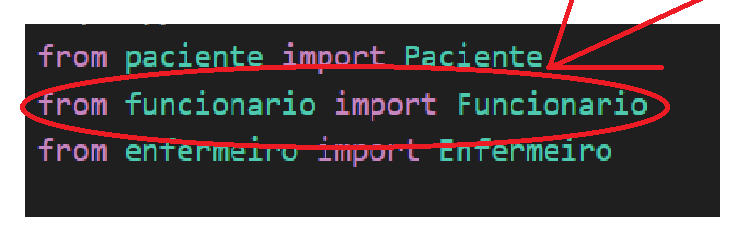


Fig – Import dos arquivos para o menu

Na imagem comprova-se a importação dos arquivos paciente, funcionário, enfermeiro para o main hospital onde ocorre a execução do menu do sistema. Seguidamente é feito três listas para guardarem os dados do paciente, funcionário e o enfermeiro.

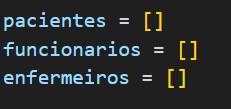


Fig – Listas

Dentro do main hospital, é feito um menu com o comando while true, onde existem ao todo 12 opções a serem selecionadas pelo utilizador, cada opção selecionada executará uma função. Na imagem abaixo, cada uma das 12 opções cumprem a função de inserir os dados do paciente, o atendimento, ver o paciente do enfermeiro, inserção dos dados do funcionário, entre outras.

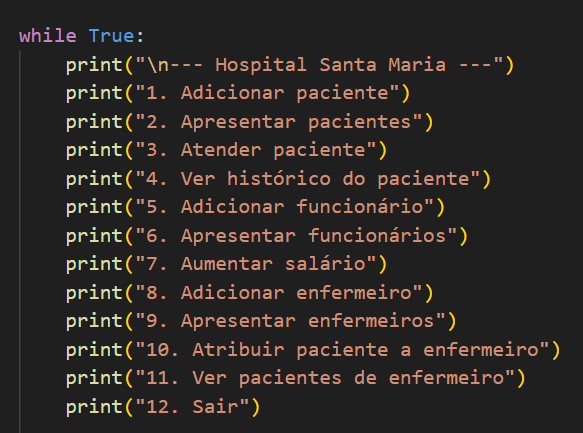


Fig – Menu de seleção

Depois é feita uma sequência de se (if) para cada opção, destacando a primeira onde se faz a inserção dos dados do paciente.

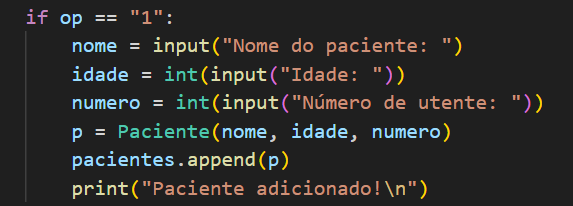


Fig – Opção para inserir o paciente

## Execução do sistema

* Execução do menu

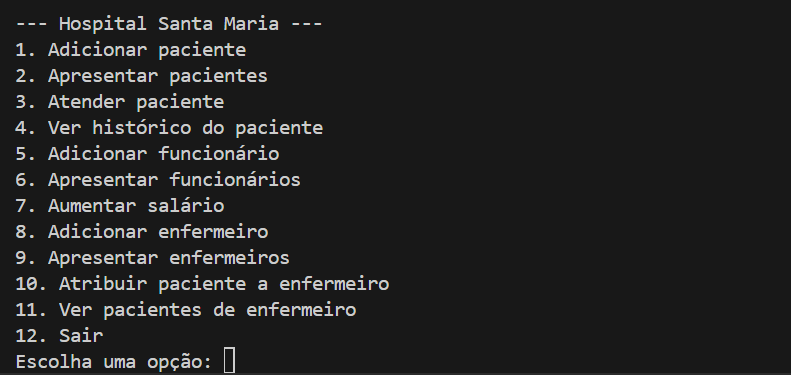


Fig – Menu do sistema

* A inserção dos dados do paciente

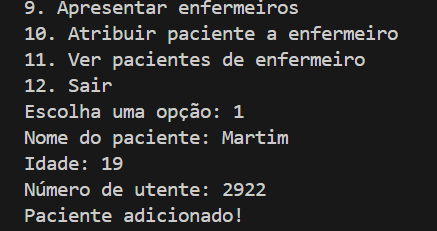


Fig – Inserção dos dados do paciente

* Apresentar o paciente

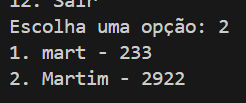


Fig – Apresentação dos pacientes

* Atendimento do paciente

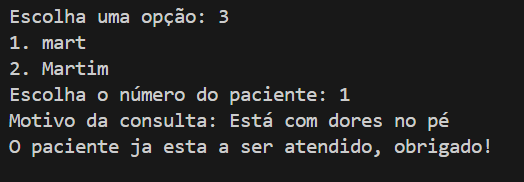


Fig – Paciente a ser atendido

* Histórico do paciente

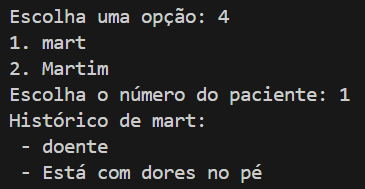


Fig – histórico do paciente

Antes concluir, é importante dizer que todas as partes do código que foram explicadas, são as partes em que se achou importante terem uma explicação das quais eram suas funções e a sua utilidade no sistema.

Capítulo III - CONCLUSÃO

Podemos concluir que este projeto ajudou nos na compreensão das heranças e os módulos, deu para perceber que com a utilização dos módulos é possível organizar o código em vários arquivos sem depender de um apenas e depois ter uma confusão entre onde começa e acaba o código, sobre as heranças entendemos o modo de utilização múltiplas heranças e como classes abstratas podem fazer a herança dos seus atributos sem ter qualquer função direta no código.

O sistema do hospital, concluído, agora com as suas funções disponíveis para ser utilizados. Este projeto serviu de um aprendizado que nos ajudará no futuro com as nossas competências com linguagem de programação Python.