

UNIVERSIDADE ESTADUAL DO NORTE FLUMINENSE  
LABORATÓRIO DE ENGENHARIA E EXPLORAÇÃO DE PETRÓLEO

SIMULADOR PARA ESTIMATIVA DE PROPRIEDADES  
PETROFÍSICAS A PARTIR DE PERFIS GEOFÍSICOS DE POÇOS

POLLYANA FERREIRA DA SILVA  
ÉRICA MELLO LISBÔA  
Prof. André Duarte Bueno

MACAÉ - RJ  
Janeiro - 2015

# Sumário

<b>1</b>	<b>Introdução</b>	<b>1</b>
1.1	Escopo do problema . . . . .	1
1.2	Objetivos . . . . .	2
<b>2</b>	<b>Especificação</b>	<b>3</b>
2.1	Especificação do programa - descrição dos requisitos . . . . .	3
2.2	Especificação do software - requisitos . . . . .	3
2.2.1	Nome do sistema/produto . . . . .	3
2.2.2	Requisitos funcionais . . . . .	4
2.2.3	Requisitos não funcionais . . . . .	4
2.3	O que são os casos de uso do software? . . . . .	4
2.4	Casos de uso do software . . . . .	4
2.4.1	Diagrama de caso de uso geral . . . . .	5
2.4.2	Diagrama de caso de uso específico . . . . .	5
<b>3</b>	<b>Elaboração</b>	<b>6</b>
3.1	Análise de domínio . . . . .	6
3.2	Perfilagem . . . . .	7
3.2.1	Perfil de Raio Gama (GR) . . . . .	9
3.2.2	Perfil de Densidade (RHOB) . . . . .	9
3.2.3	Perfil de Resistividade - Indução (ILD) . . . . .	10
3.2.4	Perfil Neutrônico (NPHI) . . . . .	11
3.2.5	Perfil Sônico (DT) . . . . .	11
3.3	Identificação de pacotes – assuntos . . . . .	11
3.4	Diagrama de pacotes – assuntos . . . . .	12
<b>4</b>	<b>AOO – Análise Orientada a Objeto</b>	<b>14</b>
4.1	Diagramas de classes . . . . .	14
4.1.1	Dicionário de classes . . . . .	16
4.2	Diagrama de sequência – eventos e mensagens . . . . .	21
4.2.1	Diagrama de sequência geral . . . . .	21
4.2.2	Diagrama de sequência específico . . . . .	22

---

4.3	Diagrama de comunicação – colaboração . . . . .	23
4.4	Diagrama de máquina de estado . . . . .	24
4.5	Diagrama de atividades . . . . .	25
<b>5</b>	<b>Projeto</b>	<b>26</b>
5.1	Projeto do sistema . . . . .	26
5.2	Projeto orientado a objeto – POO . . . . .	27
5.3	Diagrama de componentes . . . . .	28
5.4	Diagrama de implantação . . . . .	29
<b>6</b>	<b>Implementação</b>	<b>31</b>
6.1	Código fonte . . . . .	31
<b>7</b>	<b>Teste</b>	<b>58</b>
7.1	Teste 1: Interface . . . . .	58
7.2	Teste 2: Plotar Perfis . . . . .	58
7.3	Teste 2: Cálculo da Argilosidade . . . . .	59
7.4	Teste 3: Cálculo da Porosidade . . . . .	62
7.5	Teste 4: Cálculo da Saturação . . . . .	63
7.6	Teste 5: Ver Litologia . . . . .	64
<b>8</b>	<b>Documentação</b>	<b>66</b>
8.1	Documentação do usuário . . . . .	66
8.1.1	Como rodar o software . . . . .	66
8.2	Documentação para desenvolvedor . . . . .	67
8.2.1	Dependências . . . . .	67
8.2.2	Documentação usando doxygen . . . . .	69
<b>9</b>	<b>Referências Bibliográficas</b>	<b>72</b>

# Capítulo 1

## Introdução

No presente projeto de engenharia desenvolve-se o “Simulador para Estimativa de Propriedade Petrofísicas a partir de Perfis Geofísicos de Poços”, um software aplicado a engenharia de petróleo e que utiliza o paradigma da orientação a objetos, utilizando a linguagem C++.

A importância desse trabalho está relacionada com a otimização da perfuração para prospecção de petróleo. Além disso, levando em conta que no Laboratório de Engenharia e Exploração de Protróleo (LENEP) da Universidade Estadual do Norte Fluminense (UNF) não há softwares para realização dos cálculos que quantificam essas propriedades, o presente trabalho possibilita os cálculos das mesmas, podendo, portanto, ser usado para fins acadêmicos.

### 1.1 Escopo do problema

Este trabalho utiliza as técnicas de perfilagem geofísica com o propósito de qualificar reservatórios para identificar o topo e a base do reservatório. A caracterização do reservatório tem como objetivo central a obtenção das propriedades petrofísicas das rochas constituintes do reservatório tais como espessura, litologia, porosidade, permeabilidade e saturação de água/hidrocarboneto, para este fim são utilizados os perfis geofísicos de poço: Raios Gama (GR), Neutrônico (NPHI), Resistividade (ILD), Sônico (DT), Densidade (RHOB) e Caliper (CAL).

O perfil de poço é a prática de efetuar um registro detalhado das formações geológicas atravessadas por um poço.

A criação do software com este fim específico, objetiva facilitar a interpretação dos perfis de poços, de modo a otimizar a produção de hidrocarbonetos e auxiliar na engenharia de Petróleo.

## 1.2 Objetivos

Os objetivos deste projeto de engenharia são:

- Objetivo geral:
  - Desenvolver um software que calcula as propriedades petrofísicas ao longo do poço a partir de equações da física;
- Objetivos específicos:
  - Plotar os perfis;
  - Calcular a porosidade;
  - Calcular a argilosidade;
  - Calcular a saturação de óleo e água;
  - Plotar as propriedades calculadas.

# Capítulo 2

## Especificação

Apresenta-se neste capítulo do projeto de engenharia a concepção, a especificação do sistema a ser modelado e desenvolvido.

### 2.1 Especificação do programa - descrição dos requisitos

Este programa tem a finalidade de calcular a saturação dos fluidos, a porosidade e a argilosidade da rocha, e os apresentar através de gráficos, podendo também plotar os perfis utilizados e o gráfico identificando as litologias.

O projeto a ser desenvolvido consiste em um software que pede ao usuário a opção que o mesmo deseja obter. Dependendo da opção escolhida o software pedirá que entre com algumas propriedades para que os cálculos possam ser efetuados. Assim, a partir da opção escolhida e/ou da propriedade calculada, o software retornará ao usuário o resultado em forma de gráfico.

O software será desenvolvido utilizando o conceito de programação orientada a objeto (POO) e sua interface será em modo texto.

### 2.2 Especificação do software - requisitos

#### 2.2.1 Nome do sistema/produto

<b>Nome</b>	SEPP
<b>Componentes principais</b>	Sistema para cálculo das propriedades petrofísicas.
<b>Missão</b>	Cálcular e plotar as propriedades petrofísicas.

### 2.2.2 Requisitos funcionais

Apresenta-se a seguir os requisitos funcionais

<b>RF-01</b>	O usuário deverá ter liberdade para escolher o que deseja calcular.
--------------	---

<b>RF-02</b>	Deve permitir o carregamento de arquivos.
--------------	---

<b>RF-03</b>	O usuário poderá plotar seus resultados em um gráfico.
--------------	--

### 2.2.3 Requisitos não funcionais

<b>RNF-01</b>	O programa deverá ser multi-plataforma, podendo ser executado em <i>Windows</i> , <i>GNU/Linux</i> ou <i>Mac</i> .
---------------	--

## 2.3 O que são os casos de uso do software?

O diagrama de casos de uso é uma representação visual dos casos de uso. É o diagrama mais simples da UML, sendo utilizado para demonstrar os cenários de uso do sistema pelos usuários, os quais ao verem esses diagramas terão uma visão geral do sistema.

Um *caso de uso* descreve um ou mais cenários de uso do software, exemplos de uso, como o sistema interage com usuários externos (atores). Ademais, ele deve representar uma seqüência típica de uso do software (a execução de determinadas tarefas-padrão). Também deve representar as exceções, casos em que o usuário comete algum erro, em que o sistema não consegue realizar as tarefas solicitadas.

## 2.4 Casos de uso do software

A Tabela 2.1 mostra os itens a serem incluídos na descrição do caso de uso.

Tabela 2.1: Exemplo de caso de uso.

Nome do caso de uso:	Cálculo de uma propriedade petrofísica.
Resumo/descrição:	Cálculo de uma propriedade a partir do perfil geofísico.
Etapas:	<ol style="list-style-type: none"><li>1. Criar objeto simulador;</li><li>2. Escolher a propriedade a ser calculada;</li><li>3. Entrar com as propriedades solicitadas;</li><li>4. Calcular a propriedade;</li><li>5. Gerar gráficos;</li><li>6. Analisar resultados.</li></ol>
Cenários alternativos:	Um cenário alternativo envolve a plotagem dos três perfis geofísicos que o software utiliza.

### 2.4.1 Diagrama de caso de uso geral

O diagrama de caso de uso geral da Figura 2.1 mostra o usuário entrando com os dados, calculando a propriedade e analisando resultados.

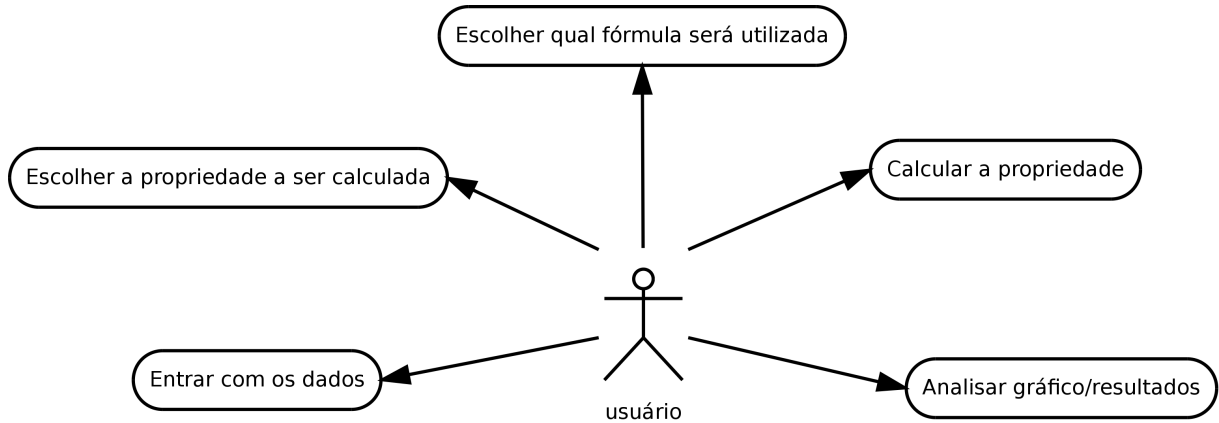


Figura 2.1: Diagrama de caso de uso: geral

### 2.4.2 Diagrama de caso de uso específico

A Figura 2.2 mostra o diagrama de caso de uso específico do método calcular saturação de Archie que considera o grau de argilosidade da rocha. Este diagrama evidencia a relação entre o usuário e o software ao longo da simulação.

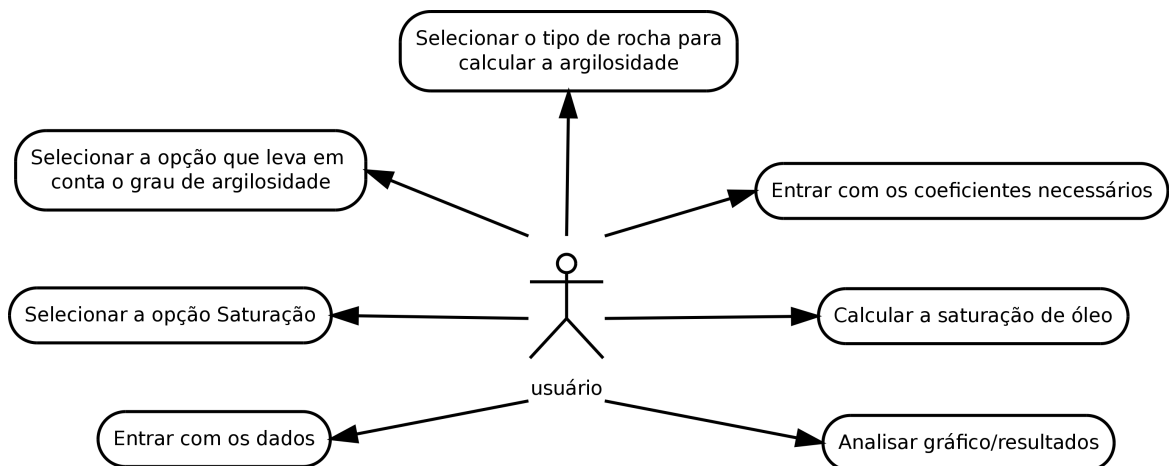


Figura 2.2: Diagrama de caso de uso: específico - Cálculo da Saturação



# Capítulo 3

## Elaboração

Depois da definição dos objetivos, é necessário um processo de elaboração do software a ser desenvolvido. São definidas as áreas de atuação do software e as equações utilizadas para a solução do problema. Por fim, monta-se o diagrama de pacotes para mostrar os assuntos relacionados.

### 3.1 Análise de domínio

É importante compreender a abrangência do sistema a ser desenvolvido, o que possibilita o desenvolvimento do diagrama de pacotes, que representa as áreas abordadas pelo software.

A Figura 3.1 ilustra as diferentes áreas relacionadas ao software. Observe que está relacionado com a Petrofísica, pois é a área que trata das propriedades físicas de minerais e rochas (incluindo composição matricial e espaço poroso, além dos fluidos que as percorrem); com a Geologia, interessada na litologia; com a Engenharia de Poço, que envolve um conjunto de técnicas para desenvolver projetos que visem a exploração sem prejuízo econômico.

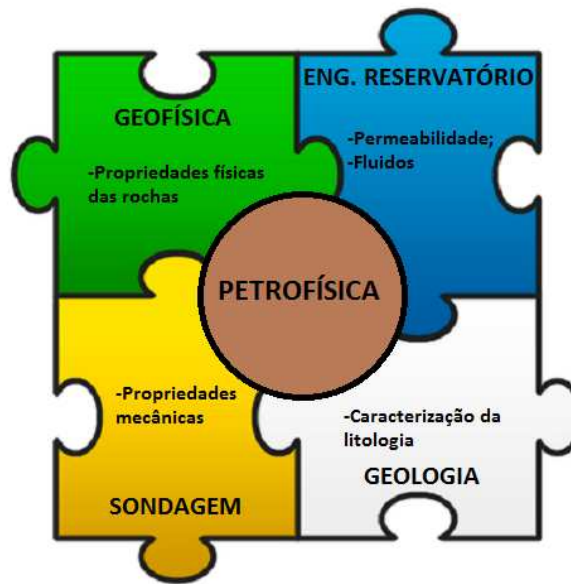


Figura 3.1: Elaboração e análise de domínio

## 3.2 Perfilagem

A perfilagem trata do levantamento de diferentes perfis referentes a um determinado poço. O perfil é obtido a partir de ferramentas (elétricas, eletromagnéticas e radiativas) que são descidas no poço, os valores de resposta são captados e armazenados em arquivos digitais. Estes arquivos são processados, em tempo real, e depois usados em softwares específicos, como o que estamos desenvolvendo. A Figura 3.2 mostra os perfis do Campo de Namorados.

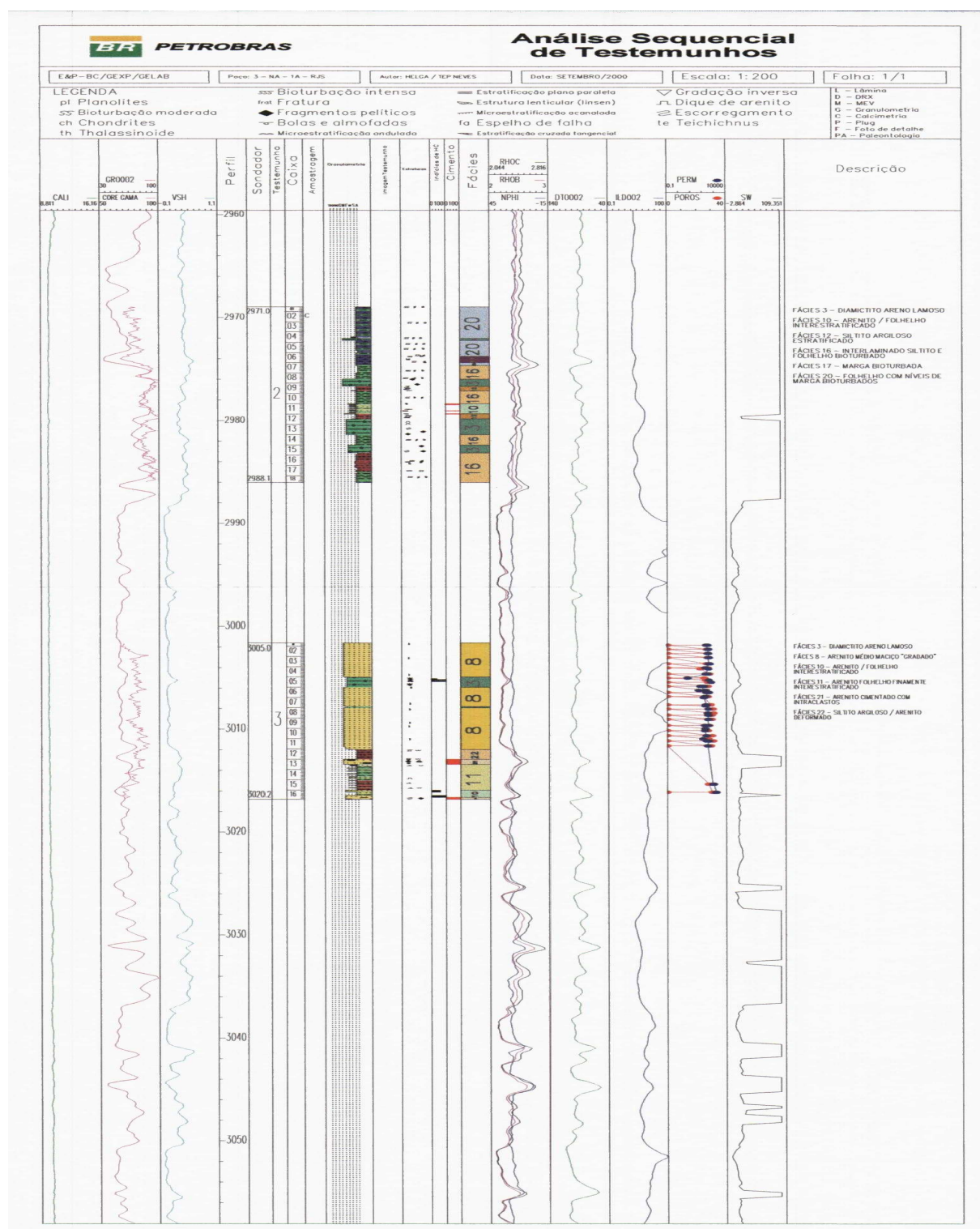


Figura 3.2: Perfil do Poço NA01A do Campo de Namorados/imagem cedida pela Petrobras à UENF-LENEP

Existem vários tipos de perfis que são utilizados com o objetivo de avaliar as formações geológicas quanto a ocorrência de uma jazida comercial de hidrocarbonetos.

Nas próximas seções apresentaremos os perfis: raio gama, resistividade, densidade, neutrônico e sônico. Entretanto, os dois últimos não são utilizados no software.

### 3.2.1 Perfil de Raio Gama (GR)

Os elementos radioativos presentes nas rochas emitem uma radiação natural que são detectáveis por contadores de cintilação (ou cintilômetros), por câmaras de ionização ou, ocasionalmente, por um contador Geiger-Müller. O perfil de Raio Gama detecta a radioatividade total da formação geológica, e a unidade é expressa em API (American Petroleum Institute).

O perfil de Raio Gama é utilizado para obter, quantitativamente, o volume de argila. Sabendo-se que o perfil de Raios Gama reflete a proporção de folhelho ou argila de uma camada, pode-se utilizá-lo como um indicador do teor de folhelho ou argilosidade das rochas (NERY, 2004).

A partir da leitura deste perfil pode-se calcular primeiramente o índice de raios gama, que pode ser interpretado como o próprio volume de argila, através da seguinte fórmula:

$$V_{sh} = IGR = \frac{GR_{log} - GR_{min}}{GR_{max} - GR_{min}} \quad (3.1)$$

Onde:

$V_{sh}$  = volume de argila [API/API];

$IGR$  = índice de raio gama [API/API];

$GR_{log}$  = raio gama lido na formação [API];

$GR_{max}$  = raio gama máximo da formação [API];

$GR_{min}$  = raio gama mínimo da formação [API];

Podemos calcular, também, o volume de argila segundo Rider (2002), no qual as equações para os cálculos de  $V_{sh}$  das rochas consolidadas (anteriores ao terciário) e das rochas inconsolidadas (terciário) são, respectivamente:

$$V_{sh} = 0.33(2^{2*IGR} - 1) \quad (3.2)$$

e

$$V_{sh} = 0.083(2^{3.7*IGR} - 1) \quad (3.3)$$

### 3.2.2 Perfil de Densidade (RHOB)

O valor de densidade global [ $g/cm^3$ ] é determinado através da colisão de raios gama artificiais, utilizando uma fonte de Cobalto (Co) ou Césio (Cs), com os elementos da formação, criando o fenômeno conhecido como espalhamento de Compton (STEVANATO, 2011).

O perfil de densidade de raio gama é um registro contínuo do valor de densidade global da rocha (matriz sólida e fluidos presentes).

Quantitativamente, este perfil é utilizado no cálculo da porosidade, qualitativamente, este perfil é um indicador litológico, que pode ser utilizado para identificar certos minerais,

ajudar na avaliação do conteúdo de matéria orgânica da rocha geradora e também na identificação de regiões de alta pressão e de fraturas (RIDER, 2002).

A equação utilizada para calcular a porosidade é:

$$\phi = \frac{\rho - \rho_{\min}}{\rho_{\max} - \rho_{\min}} \quad (3.4)$$

onde:

$\phi$  = porosidade [ $m^3/m^3$ ];

$\rho$  = densidade lida [ $g/cm^3$ ];

$\rho_{\min}$  = densidade mínima [ $g/cm^3$ ];

$\rho_{\max}$  = densidade máxima [ $g/cm^3$ ];

### 3.2.3 Perfil de Resistividade - Indução (ILD)

A medição da resistividade [ $ohm.m$ ] é realizada através da ferramenta de indução, que é responsável pela medição da condutividade (inverso da resistividade). A sonda de indução é constituída por duas bobinas, uma transmissora onde é aplicada uma corrente constante de alta frequência e uma receptora. A corrente aplicada à bobina transmissora gera um campo eletromagnético ao redor da ferramenta, o que por sua vez induz correntes na formação. As correntes induzidas criam um campo eletromagnético secundário que induz uma corrente alternada na bobina receptora (RIDER, 2011).

O perfil de resistividade consiste na medição da resistividade da formação, que é a propriedade física que impede o fluxo de corrente elétrica.

A saturação da água pode ser determinada a partir dos dados fornecidos por este perfil. Com este perfil pode-se determinar também a saturação de hidrocarbonetos do reservatório. Para estes cálculos deve-se ter o conhecimento prévio da resistividade da água, da porosidade e do expoente de cimentação e saturação. Esses parâmetros relacionam-se através da equação de Archie:

$$R_t = \left( \frac{a * R_w}{\phi^m * S_w^n} \right) \quad (3.5)$$

Onde:

$R_t$  = resistividade total [ $ohm.m$ ];

$R_w$  = resistividade da água [ $ohm.m$ ];

$S_w$  = saturação da água [ $m^3/m^3$ ];

$\phi$  = porosidade [ $m^3/m^3$ ];

$a$  = coeficiente de tortuosidade [ $m^3/m^3$ ];

$n$  = coeficiente de saturação [ $m^3/m^3$ ];

$m$  = coeficiente de cimentação [ $m^3/m^3$ ];

Os parâmetros  $a$ ,  $m$  e  $n$  podem ser obtidos em laboratório, da experiência da área ou dos próprios perfis.

Caso queira considerar o volume de argila da litologia, calcula-se saturação da água através da equação de Archie modificada:

$$\frac{1}{R_t} = \frac{\phi^m * S_w^n}{a * R_w} + \frac{V_{sh} * S_w}{R_{sh}} \quad (3.6)$$

Onde:

$R_{sh}$  = resistividade da argila [*ohm.m*];

Para obter a saturação de óleo, utiliza-se a seguinte equação:

$$S_h = 1 - S_w \quad (3.7)$$

Onde:

$S_h$  = saturação de hidrocarboneto [ $m^3/m^3$ ];

### 3.2.4 Perfil Neutrônico (NPHI)

O perfil de neutrons ou Perfil neutrônico mede uma radioatividade, induzida artificialmente, por meio de bombardeio das rochas com nêutrons de alta energia ou velocidade. A unidade é expressa em termos de unidade de porosidade neutrão, a qual é relacionada com o índice de hidrogênio presente na formação (RIDER, 2002).

Este perfil é utilizado, quantitativamente, na medição da porosidade e qualitativamente, como um discriminador entre gás e óleo. Pode ser também utilizado na identificação de litologia em conjunto com o perfil de densidade (RIDER, 2002).

### 3.2.5 Perfil Sônico (DT)

O perfil sônico ou perfil acústico mede a diferença nos tempos de trânsito de uma onda mecânica através das rochas. Segundo Rider (2002), a capacidade de transmissão da onda sonora na formação geológica está diretamente relacionada com a litologia, textura e porosidade da mesma. A unidade de medida é expressa em microssegundos por pé de formação ( $\mu s/ft$ ).

Este perfil é utilizado para estimativas de porosidade, correlação poço a poço, estimativas de grau de compactação das rochas ou estimativas das constantes elásticas, detecção de fraturas e apoio à sísmica para a elaboração do sismograma sintético (THOMAS, 2001).

## 3.3 Identificação de pacotes – assuntos

Apresentados os conceitos físico-matemáticos relacionados ao software desenvolvido, podemos montar nossa lista de assuntos (pacotes):

- Perfil: representa os perfis obtidos (Resistividade, Raio Gama e Densidade);

- Propriedades Petrofísicas: serve para calcular as propriedades como a porosidade, a saturação e a argilosidade, através das informações do pacote Perfil;
- Gráficos: um softwares externo (Gnuplot) plota os gráficos dos perfis e das propriedades petrofísicas;
- Simulador: faz a interligação entre os pacotes.

### 3.4 Diagrama de pacotes – assuntos

A Figura 3.3 mostra o diagrama de pacotes do software (SEPP).

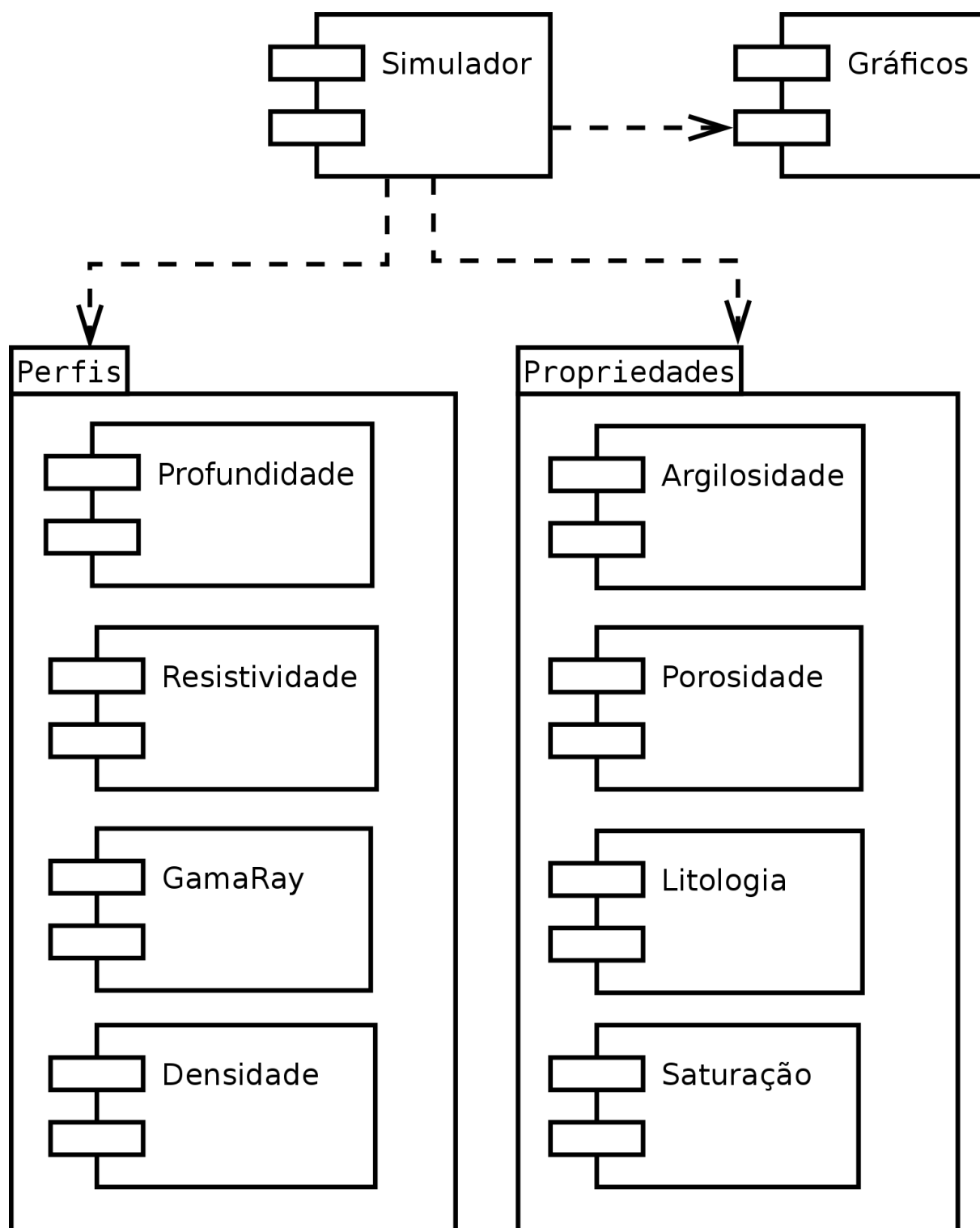


Figura 3.3: Diagrama de Pacotes



# Capítulo 4

## AOO – Análise Orientada a Objeto

A terceira etapa do desenvolvimento de um projeto de engenharia, no nosso caso um software aplicado a engenharia de petróleo, é a AOO - Análise Orientada a Objeto. A AOO utiliza algumas regras para identificar os objetos de interesse, as relações entre os pacotes, as classes, os atributos, os métodos, as heranças, as associações, as agregações, as composições e as dependências.

O modelo de análise deve ser conciso, simplificado e deve mostrar o que deve ser feito, não se preocupando como isso será realizado.

O resultado da análise é um conjunto de diagramas que identificam os objetos e seus relacionamentos.

### 4.1 Diagramas de classes

O diagrama de classes é apresentado na Figura 4.1.

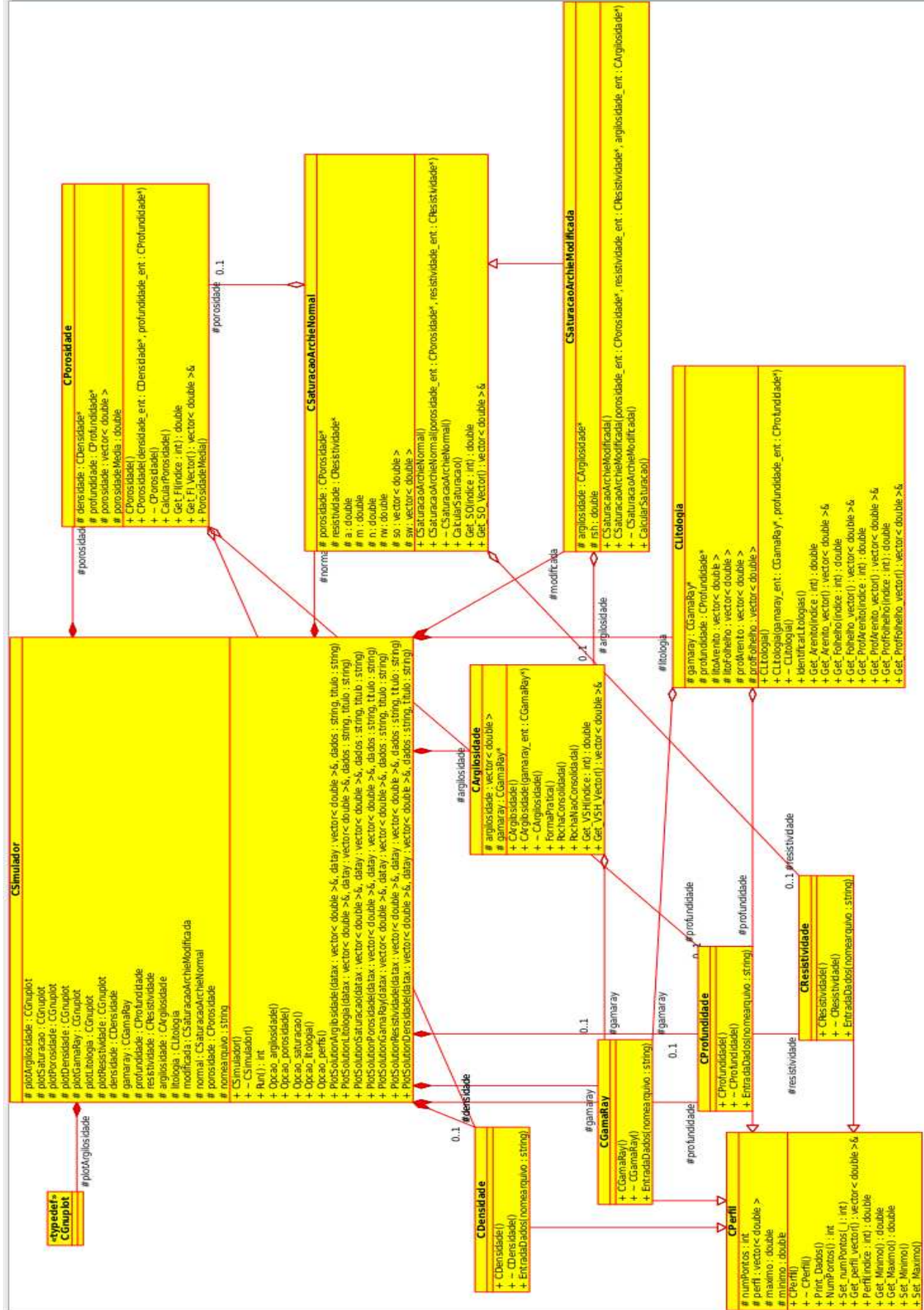


Figura 4.1: Diagrama de classes

### 4.1.1 Dicionário de classes

- Classe CSimulador: representa a simulação do programa como um todo.
  - atributo plotArgilosidade: representa o gráfico da argilosidade;
  - atributo plotSaturacao: representa o gráfico da saturação;
  - atributo plotPorosidade: representa o gráfico da porosidade;
  - atributo plotDensidade: representa o gráfico da densidade;
  - atributo plotGamaRay: representa o gráfico da radiação Gama;
  - atributo plotLitologia: representa o gráfico que demonstra as áreas de arenito e folhelho;
  - atributo plotResistividade: representa o gráfico da resistividade total;
  - atributo densidade: representa a classe CDensidade;
  - atributo gamaray: representa a classe CGamaRay;
  - atributo profundidade: representa a classe CProfundidade;
  - atributo resistividade: representa a classe CResistividade;
  - atributo argilosidade: representa a classe CArgilosidade;
  - atributo litologia: representa a classe CLitologia;
  - atributo modificada: representa a classe CSaturacaoArchieModificada;
  - atributo normal: representa a classe CSaturacaoArchieNormal;
  - atributo porosidade: representa a classe CPorosidade;
  - atributo nomearquivo: representa o nome do arquivo a ser importado;
  - método CSimulador (): método construtor da classe;
  - método ~CSimulador (): método destrutor da classe;
  - método Run (): funciona como um menu.
  - método Opcao\_argilosidade (): possui as funções necessárias para calcular e plotar a argilosidade;
  - método Opcao\_porosidade (): possui as funções necessárias para calcular e plotar a porosidade;
  - método Opcao\_saturacao (): possui as funções necessárias para calcular e plotar a saturação;
  - método Opcao\_litologia (): possui as funções necessárias para mostrar as litologias ;
  - método Opcao\_perfil (): possui as funções necessárias para mostrar os perfis;

- método `PlotSolutionArgilosidade ()`: método que utiliza o gnuplot para plotar a argilosidade;
  - método `PlotSolutionLitologia ()`: método que utiliza o gnuplot para plotar as litologias;
  - método `PlotSolutionSaturacao ()`: método que utiliza o gnuplot para plotar a saturação de hidrocarboneto;
  - método `PlotSolutionPorosidade ()`: método que utiliza o gnuplot para plotar a porosidade;
  - método `PlotSolutionGamaRay ()`: método que utiliza o gnuplot para plotar a radiação gama;
  - método `PlotSolutionResistividade ()`: método que utiliza o gnuplot para plotar a resistividade;
  - método `PlotSolutionDensidade ()`: método que utiliza o gnuplot para plotar a densidade;
- Classe `CPerfil`: possui os atributos e os métodos comuns a todos os perfis.
    - atributo `numPontos`: guarda o numero de pontos dos vetores dos perfis;
    - atributo `perfil`: armazena os valores dos perfis;
    - atributo `maximo`: armazena o valor máximo do vetor perfil;
    - atributo `minimo`: armazena o valor mínimo do vetor perfil;
    - método `CPerfil ()`: método construtor da classe;
    - método `~CPerfil ()`: método destrutor da classe;
    - método `Print_Dados ()`: chama o método de cada perfil que imprime os dados na tela;
    - método `NumPontos ()`: retorna o número de pontos;
    - método `Set_numPontos ()`: armazena o número de pontos;
    - método `Get_Maximo ()`: retorna o valor máximo do vetor perfil;
    - método `Get_Minimo ()`: retorna o valor mínimo do vetor perfil;
    - método `Set_Maximo ()`: armazena o valor máximo do vetor perfil;
    - método `Set_Minimo ()`: armazena o valor mínimo do vetor perfil;
    - método `Get_perfil_vector ()`: retorna o valor máximo do vetor perfil;
  - Classe `CDensidade`: é uma classe que armazena os valores de densidade do poço.
    - método `CDensidade ()`: método construtor da classe;

- método `~CDensidade ()`: método destrutor da classe;
- método `EntradaDados ()`: inicializa o vetor com os valores de densidade utilizando arquivo de extensão `.dat`;
- Classe `CGamaRay`: é uma classe que armazena os valores de radiação Gama do poço.
  - método `CGamaRay ()`: método construtor da classe;
  - método `~CGamaRay ()`: método destrutor da classe;
  - método `EntradaDados ()`: inicializa o vetor de radiação Gama, com os valores do arquivo de extensão `.dat`;
- Classe `CProfundidade`: é uma classe que armazena os valores da profundidade medidas do poço.
  - método `CProfundidade ()`: método construtor da classe;
  - método `~CProfundidade ()`: método destrutor da classe;
  - método `EntradaDados ()`: inicializa o vetor de profundidade, com os valores do arquivo de extensão `.dat`;
- Classe `CResistividade`: é uma classe que armazena os valores da resistividade do poço.
  - método `CResistividade ()`: método construtor da classe;
  - método `~CResistividade ()`: método destrutor da classe;
  - método `EntradaDados ()`: inicializa o vetor de resistividade, com os valores do arquivo de extensão `.dat`;
- Classe `CArgilosidade`: é a classe que contém os cálculos necessário para obter a argilosidade e a armazena.
  - atributo `argilosidade`: vetor que armazena os valores da argilosidade calculada;
  - atributo `gamaray`: representa um ponteiro para acessar os dados da classe `CGamaRay`;
  - método `CArgilosidade ()`: método construtor da classe;
  - método `CArgilosidade (CGamaRay* gamaray_ent)`: construtor sobrecarregado da classe;
  - método `~CArgilosidade ()`: método destrutor da classe;
  - método `FormaPratica ()`: método que calcula a argilosidade utilizando a fórmula simples;

- método RochaConsolidada(): método que calcula a argilosidade utilizando a fórmula para rochas consolidadas;
- método RochaNãoConsolidada (): método que calcula a argilosidade utilizando a fórmula para rochas não consolidadas;
- método Get\_VSH (): retorna um valor de argilosidade de determinada posição;
- método Get\_VSH\_Vector (): retorna uma referência do vetor de argilosidade;
- Classe CPorosidade: mostra a porosidade das formações.
  - atributo porosidade: é um vetor das porosidades ;
  - atributo profundidade: é um ponteiro para a classe CProfundidade ;
  - atributo densidade: é um ponteiro para a classe CDensidade ;
  - atributo porosidadeMedia: é um atributo que armazena a porosidade média;
  - método CPorosidade (): método construtor da classe;
  - método CPorosidade (CDensidade\* densidade\_ent, CProfundidade\* profundidade\_ent): construtor sobrecarregado da classe;
  - método ~CPorosidade (): método destrutor da classe;
  - método CalcularPorosidade (): método que calcula a porosidade;
  - método PorosidadeMedia (): é um método que retorna a porosidade média;
  - método Get\_FI (): retorna um valor de porosidade de determinada posição;
  - método Get\_FI\_Vector (): retorna uma referência do vetor de porosidade;
- Classe CSaturacaoArchieNormal: representa a saturação de óleo e de água ao longo do poço, sem considerar a argilosidade.
  - atributo sw: é um vetor que armazena os valores da saturação de água;
  - atributo so: é um vetor que armazena os valores da saturação de óleo;
  - atributo a: representa o valor do coeficiente de tortuosidade;
  - atributo n: representa o valor do coeficiente de saturação;
  - atributo m: representa o valor do coeficiente de cimentação;
  - atributo rw: representa o valor da resistividade da água;
  - atributo porosidade: representa um ponteiro para acessar a classe CPorosidade;
  - atributo resistividade: representa um ponteiro para acessar a classe CResistividade;
  - método CSaturacaoArchieNormal (): método construtor da classe;

- método CSaturacaoArchieNormal ( CPorosidade\* porosidade\_ent, CResistividade\* resistividade\_ent): método construtor sobrecarregado da classe;
  - método ~CSaturacaoArchieNormal (): método destrutor da classe;
  - método CalcularSaturacao (): método que calcula a saturação, através da Lei de Archie Normal;
  - método Get\_SO (): retorna um valor de saturação de determinada posição;
  - método Get\_SO\_Vector (): retorna uma referência do vetor de saturação;
- Classe CSaturacaoArchieModificada: representa a saturação de óleo e de água ao longo do poço, levando em consideração o grau de argilosidade da rocha. Considera-se que todos os atributos e métodos da classe mãe, CSaturacaoArchieNormal, são herdados.
    - atributo rsh: armazena em um valor da resistividade média da argila contida na formação;
    - atributo argilosidade: é um ponteiro para a classe CArgilosidade;
    - método CSaturacaoArchieModificada (): método construtor da classe;
    - método ~CSaturacaoArchieModificada (): método destrutor da classe;
    - método CSaturacaoArchieModificada ( CPorosidade\* porosidade\_ent, CResistividade\* resistividade\_ent, CArgilosidade\* argilosidade\_ent): método construtor sobrecarregado da classe;
    - método CalcularSaturacao (): método que calcula a saturação através da Lei de Archie Modificada;
- Classe CLitologia: mostra as regiões de folhelho e arenito, ao longo do poço.
    - atributo litoArenito: vetor que armazena os valores de radiação Gama referentes a região de arenito;
    - atributo litoFolhelho: vetor que armazena os valores de radiação Gama referentes a região de folhelho;
    - atributo profArenito: vetor que armazena os valores de profundidade referentes a região de arenito;
    - atributo profFolhelho: vetor que armazena os valores de profundidade referentes a região de folhelho;
    - atributo gamaray: é um ponteiro para a classe CGamaRay;
    - atributo profundidade: é um ponteiro para a classe CProfundidade;
    - método CLitologia (): método construtor da classe;

- método CLitologia (CGamaRay\* gamaray, CProfundidade\* profundidade): método construtor sobrecarregado da classe;
- método ~CLitologia (): método destrutor da classe;
- método IndentificarLitologia (): método que separa o vetor de radiação Gama em diferentes vetores; um vetor referente ao arenito e outro ao folhelho;
- método Get\_Arenito (): retorna um valor específico do vetor de arenito;
- método Get\_Arenito\_Vector (): retorna uma referência do vetor de arenito;
- método Get\_Folhelho (): retorna um valor específico do vetor de folhelho;
- método Get\_Folhelho\_Vector (): retorna uma referência do vetor de folhelho;
- método Get\_ProfArenito (): retorna um valor específico do vetor de profundidade do arenito;
- método Get\_ProfFolhelho (): retorna um valor específico do vetor de profundidade do folhelho;
- método Get\_ProfArenito\_Vector (): retorna uma referência do vetor de profundidade de arenito;
- método Get\_ProfFolhelho\_Vector (): retorna uma referência do vetor de profundidade do folhelho;

## 4.2 Diagrama de sequência – eventos e mensagens

O diagrama de sequência enfatiza a troca de eventos e mensagens e sua ordem temporal. Contém informações sobre o fluxo de controle do software. Costuma ser montado a partir de um diagrama de caso de uso e estabelece o relacionamento dos atores (usuários e sistemas externos) com alguns objetos do sistema.

### 4.2.1 Diagrama de sequência geral

O diagrama de sequência, Figura 4.2, demonstra a sequência de como o software faz para plotar os Perfis petrofísicos. Esta é a função mais básica do sistema, por isso escolheu-se esta sequência para ser representada no diagrama.



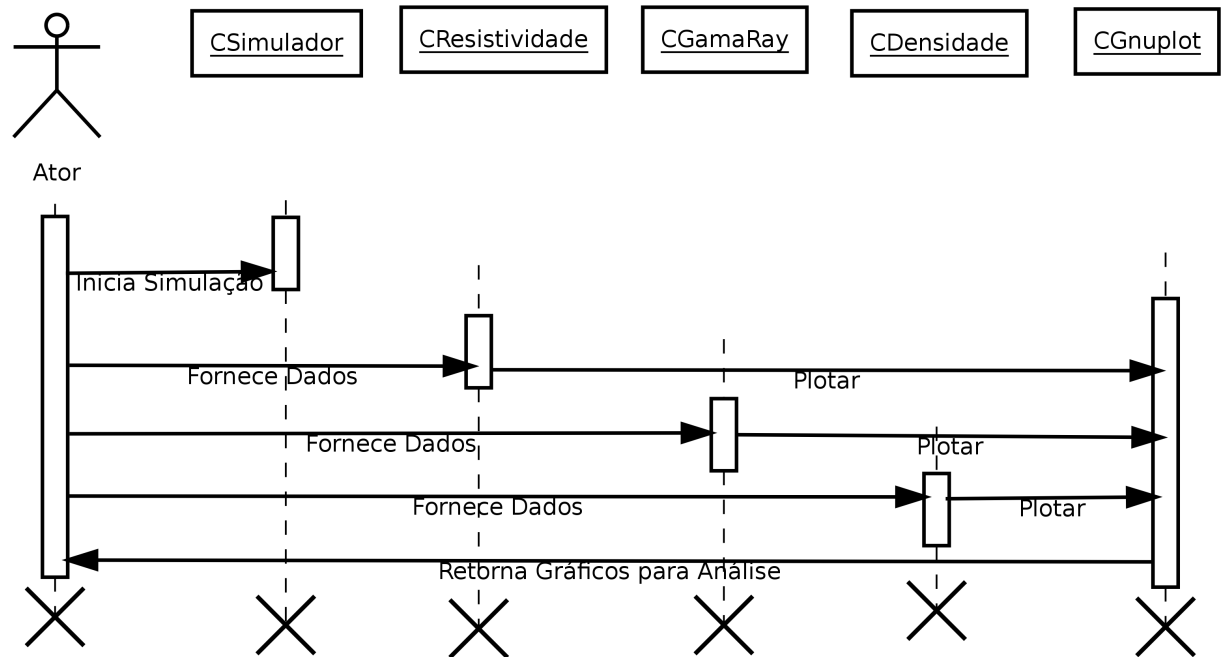


Figura 4.2: Diagrama de sequência: geral

### 4.2.2 Diagrama de sequência específico

O diagrama de sequência representado pela Figura 4.3 descreve como o sistema funciona caso o usuário deseje calcular a argilosidade da formação. Note que esta função se interliga, principalmente, com a classe `CGamaRay`, pois essa apresenta as informações necessárias para realização do cálculo.

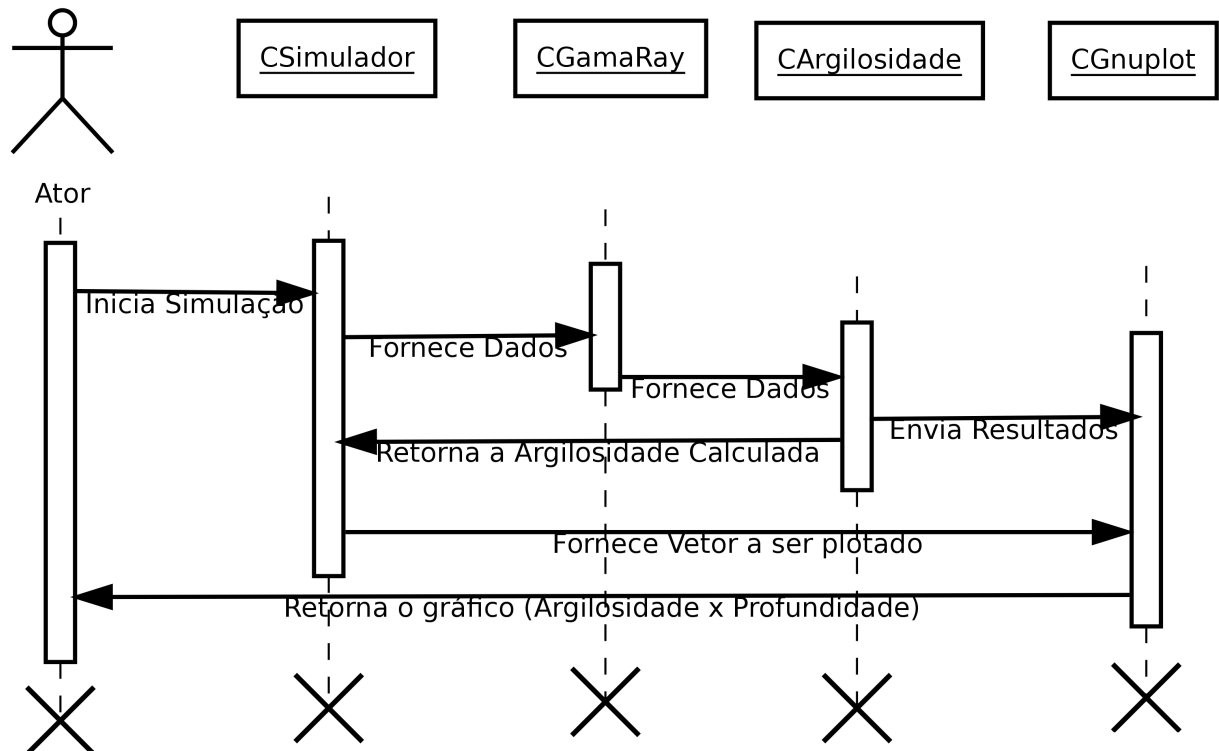


Figura 4.3: Diagrama de sequência: específico – Cálculo da Argilosidade

### 4.3 Diagrama de comunicação – colaboração

No diagrama de comunicação o foco é a interação e a troca de mensagens e dados entre os objetos.

Veja na Figura 4.3 o diagrama de comunicação geral do software.

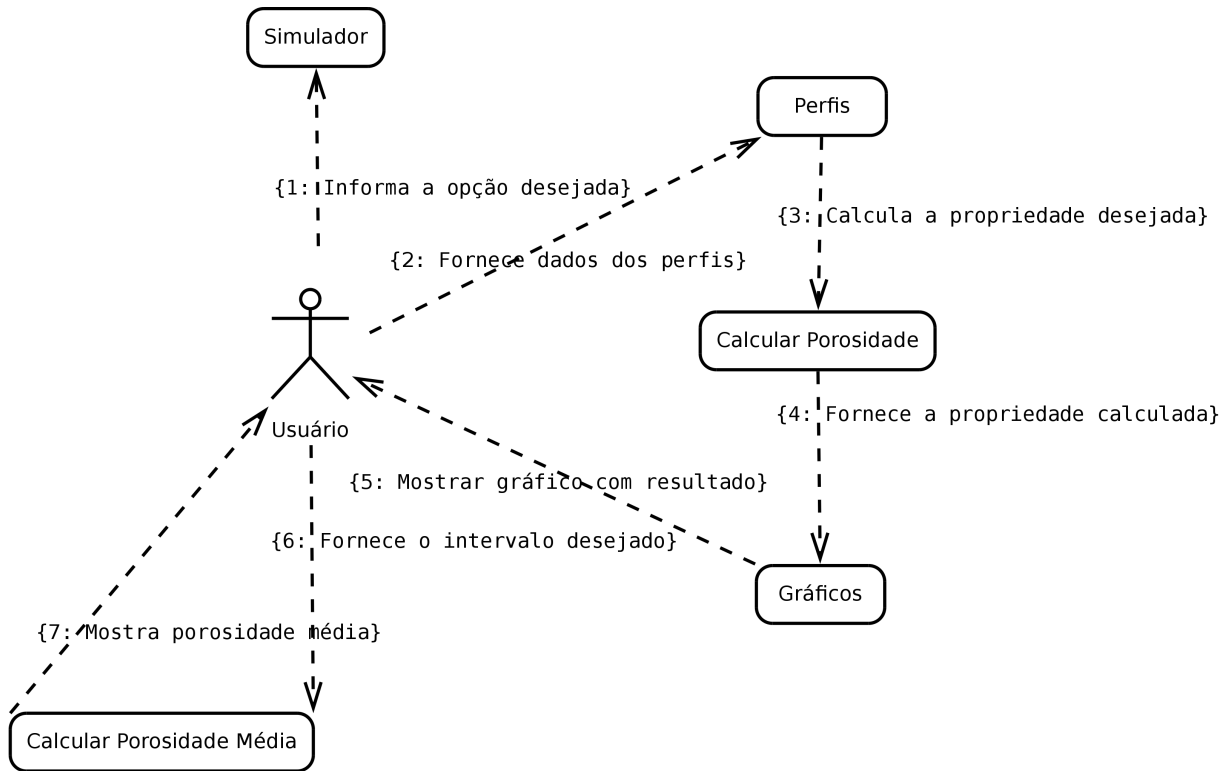


Figura 4.4: Diagrama de comunicação: Cálculo da Porosidade

## 4.4 Diagrama de máquina de estado

Um diagrama de máquina de estado representa os diversos estados que o objeto assume e os eventos que ocorrem ao longo de sua vida ou mesmo ao longo de um processo (histórico do objeto). É usado para modelar aspectos dinâmicos do objeto.

A Figura 4.5 demonstra os estados que a classe CSimulador apresenta durante a execução do software.

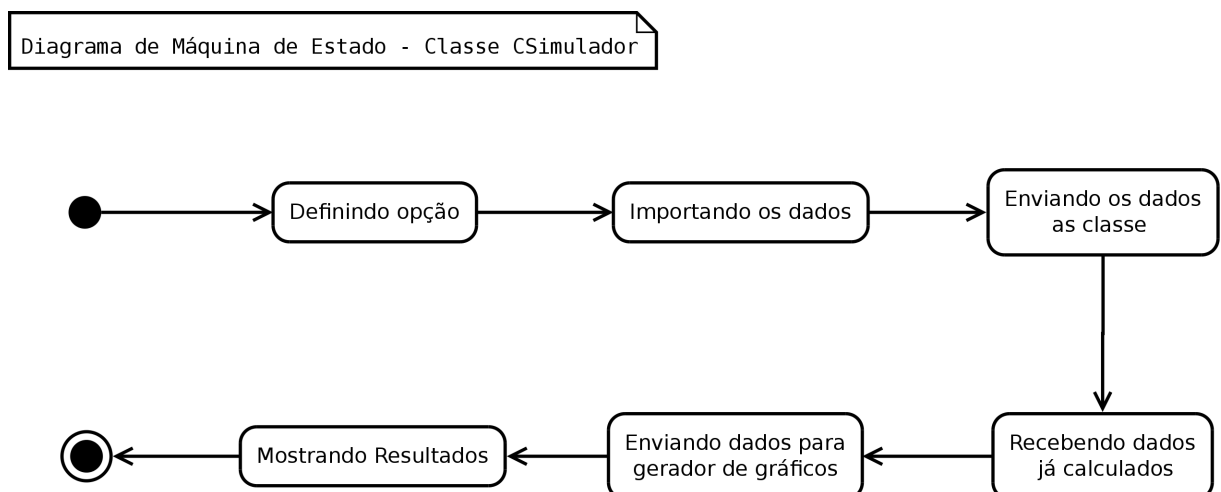


Figura 4.5: Diagrama de máquina de estado: classe CSimulador

## 4.5 Diagrama de atividades

Veja na Figura 4.6 o diagrama de atividades correspondente ao método `CalcularSaturacao` da classe `CSaturacaoArchieNormal`. Observe que a uma estrutura de repetição para preencher todo o vetor de saturação de água e o de óleo.

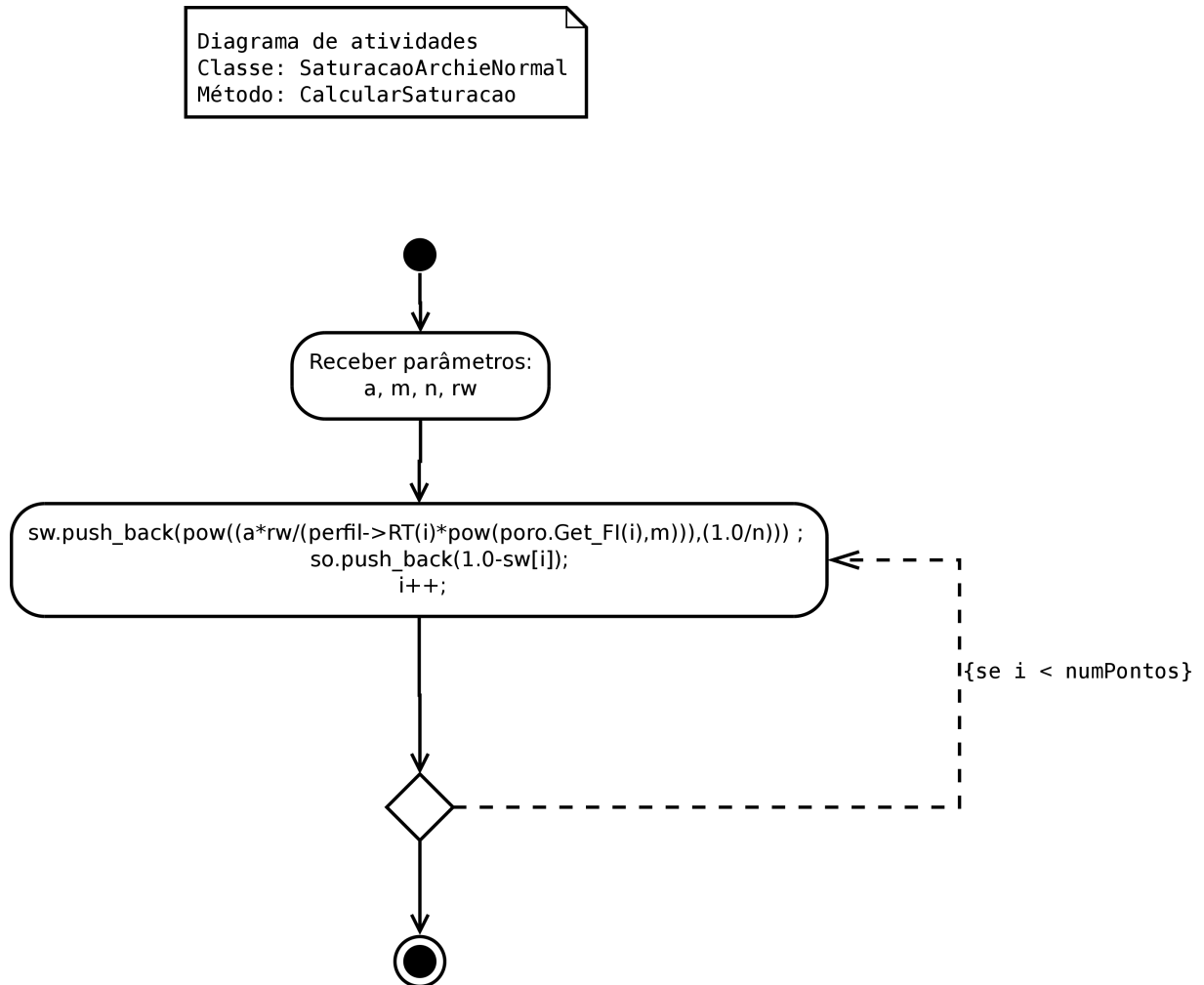


Figura 4.6: Diagrama de atividades: classe `CSaturacaoArchieNormal` – método: `CalcularSaturacao`

# Capítulo 5

## Projeto

Neste capítulo do projeto de engenharia veremos questões associadas ao projeto do sistema, incluindo protocolos, recursos, plataformas suportadas, implicações nos diagramas feitos anteriormente, diagramas de componentes e implantação. Na segunda parte revisamos os diagramas levando em conta as decisões do projeto do sistema.

### 5.1 Projeto do sistema

Depois da análise orientada a objeto desenvolve-se o projeto do sistema, o qual envolve etapas como a definição dos protocolos, da interface API, o uso de recursos, a subdivisão do sistema em subsistemas, a alocação dos subsistemas ao hardware e a seleção das estruturas de controle, a seleção das plataformas do sistema, das bibliotecas externas, dos padrões de projeto, além da tomada de decisões conceituais e políticas que formam a infraestrutura do projeto.

Deve-se definir padrões de documentação, padrões de retorno e de parâmetros em métodos, características da interface do usuário e características de desempenho.

Para o desenvolvimento do projeto é necessário se preocupar com itens como:

#### 1. Protocolos

- A única intercomunicação é entre o software desenvolvido e o componente externo, Gnuplot. Este plotará todos os gráficos que o usuário desejar;
- O software terá entrada de dados importando arquivos com extensão .dat e receberá dados via teclado;
- A interface utilizada será em modo texto.

#### 2. Recursos

- O presente software utilizará HD, o processador, o teclado, a memória, a tela e os demais componentes do computador.

### 3. Controle

- Neste projeto o controle será sequencial;
- Neste projeto todos os cálculos necessitam de estruturas de repetição, pois são feitos ao longo da profundidade do poço. É independente do tempo.

### 4. Plataformas

- O software é multiplataforma, funciona no Windows e GNU/Linux;
- A linguagem utilizada é C++;
- O software acessa a biblioteca externa CGuplot que permite o acesso ao gerador de gráficos Gnuplot.

## 5.2 Projeto orientado a objeto – POO

O projeto orientado a objeto é a etapa posterior ao projeto do sistema. Baseia-se na análise, mas considera as decisões do projeto do sistema. Acrescenta a análise desenvolvida e as características da plataforma escolhida (hardware, sistema operacional e linguagem de programação). Passa pelo maior detalhamento do funcionamento do software, acrescentando atributos e métodos que envolvem a solução de problemas específicos não identificados durante a análise.

Envolve a otimização da estrutura de dados e dos algoritmos, a minimização do tempo de execução, de memória e de custos. Existe um desvio de ênfase para os conceitos da plataforma selecionada.

Exemplo: na análise você define que existe um método para salvar um arquivo em disco, define um atributo nomeDoArquivo, mas não se preocupa com detalhes específicos da linguagem. Já no projeto, você inclui as bibliotecas necessárias para acesso ao disco, cria um objeto específico para acessar o disco, podendo, portanto, acrescentar novas classes àquelas desenvolvidas na análise.

### Efeitos do projeto no modelo estrutural

- Novas classes e associações oriundas das bibliotecas selecionadas e da linguagem escolhida devem ser acrescentadas ao modelo.
  - Neste projeto foi feito associações, composições e heranças que podem ser visualizadas no diagrama de classe, Figura 4.1.

### Efeitos do projeto no modelo dinâmico

- Os diagramas de sequência e comunicação foram modificados a medida que a implementação foi desenvolvida.

### Efeitos do projeto nos métodos

- O diagrama de classe, foi completamente modificado, heranças foram desconstruídas e associações e composições foram feitas, Figura 4.1;
- Todos os métodos foram alterados;
- De maneira geral os métodos devem ser divididos em dois tipos: i) tomada de decisões, métodos políticos ou de controle; devem ser claros, legíveis, flexíveis e usam polimorfismo. ii) realização de processamentos, podem ser otimizados e em alguns casos o polimorfismo deve ser evitado.
- Neste projeto todos os métodos estão correspondendo às respectivas responsabilidades.

### Efeitos do projeto nas heranças

- Além da herança que havia entre CSaturacaoArquieModificada e CSaturacaoArquieNormal outras foram acrescentadas, as classes CDensidade, CGamaRay, CProfundidade e CResistividade herdaram a classe CPerfil.

## 5.3 Diagrama de componentes

O diagrama de componentes mostra a forma como os componentes do software se relacionam, suas dependências. Inclui itens como: componentes, subsistemas, executáveis, nós, associações, dependências, generalizações, restrições e notas. Exemplos de componentes são bibliotecas estáticas, bibliotecas dinâmicas, dlls, componentes Java, executáveis, arquivos de disco, código-fonte.

Veja na Figura 5.1 um exemplo de diagrama de componentes. A geração da biblioteca depende dos arquivos de classe de extensão .h e .cpp. O software executável a ser gerado depende da biblioteca gerada, dos arquivos da biblioteca, dos arquivos desta e do banco de dados.

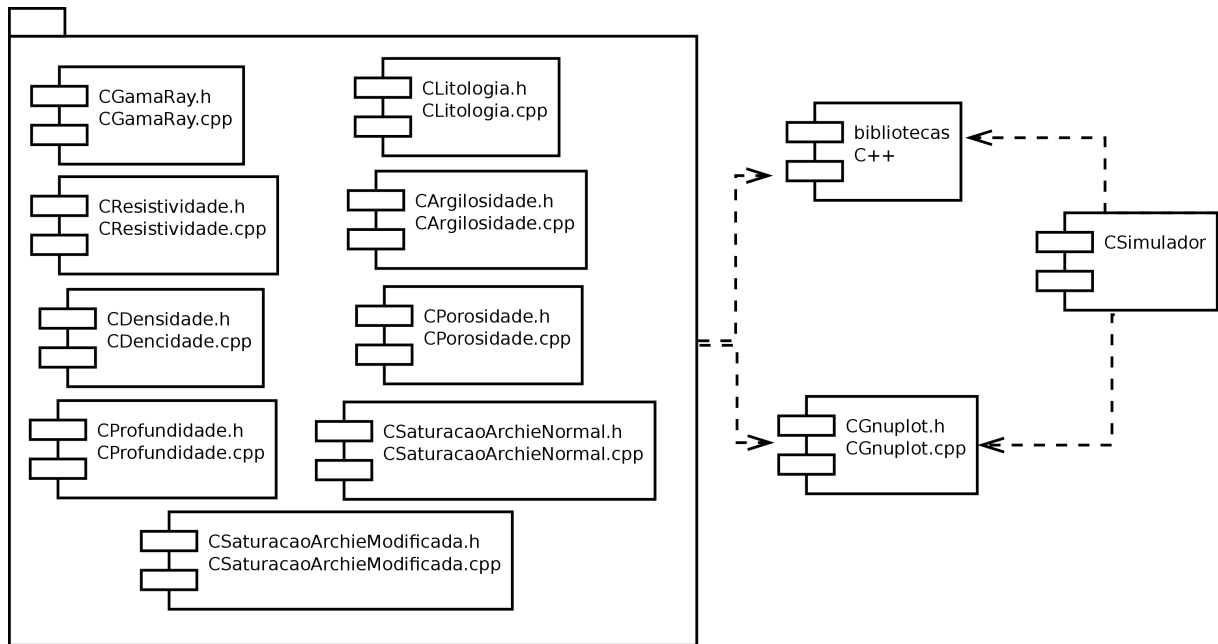


Figura 5.1: Diagrama de componentes

## 5.4 Diagrama de implantação

O diagrama de implantação é um diagrama que inclui relações entre o sistema e o hardware e deve incluir elementos necessários para que o sistema seja colocado em funcionamento: computador, periféricos, processadores, dispositivos, nós, relacionamentos de dependência, associação, componentes, subsistemas, restrições e notas.

Veja na Figura 5.2 um exemplo de diagrama de implantação do software. Primeiramente, os perfis registram as propriedades e a profundidade, enviando os dados para um computador na superfície. Esses arquivos são gerados em formato .dat. O software importa os dados deste arquivo e na execução precisa de um monitor para mostrar os resultados.



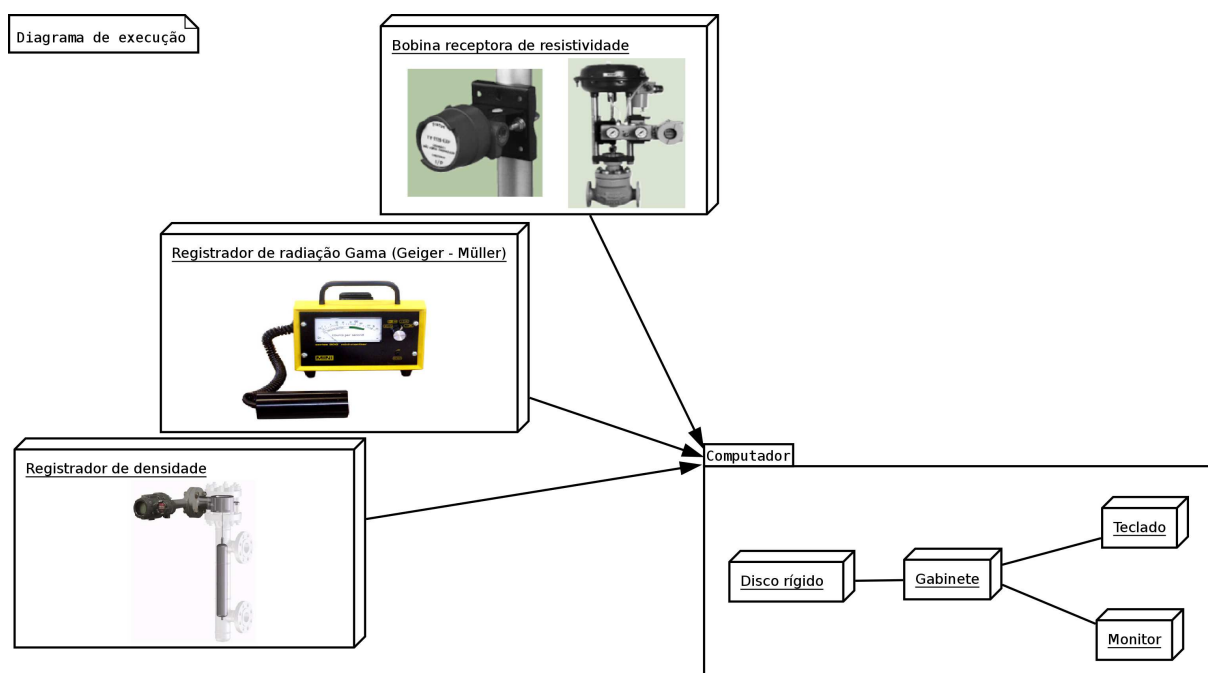


Figura 5.2: Diagrama de execução

# Capítulo 6

## Implementação

Neste capítulo do projeto de engenharia apresentamos os códigos fonte que foram desenvolvidos.

### 6.1 Código fonte

Apresenta-se a seguir um conjunto de classes (arquivos .h e .cpp) além do programa main.

Apresenta-se na listagem 6.1 o arquivo com código da classe CSimulador.

Listing 6.1: Arquivo de cabeçalho da classe CSimulador.h

```
1 #ifndef CSimulador_H
2 #define CSimulador_H
3
4 #include "CDensidade.h"
5 #include "CGamaRay.h"
6 #include "CProfundidade.h"
7 #include "CResistividade.h"
8
9 #include "CArtilosidade.h"
10 #include "CLitologia.h"
11 #include "CSaturacaoArchieModificada.h"
12 #include "CSaturacaoArchieNormal.h"
13 #include "CPorosidade.h"
14
15 #include "CGnuplot.h"
16
17 using namespace std;
18
19 class CSimulador
20 {
21
22 protected:
23     CGnuplot plotArtilosidade;
```

```

24  CGnuplot plotSaturacao;
25  CGnuplot plotPorosidade;
26  CGnuplot plotDensidade;
27  CGnuplot plotGamaRay;
28  CGnuplot plotLitologia;
29  CGnuplot plotResistividade;
30
31
32  CDensidade densidade;
33  CGamaRay gamaray;
34  CProfundidade profundidade;
35  CResistividade resistividade;
36
37  CArgilosidade argilosidade;
38  CLitologia litologia;
39  CSaturacaoArchieModificada modificada;
40  CSaturacaoArchieNormal normal;
41  CPorosidade porosidade;
42
43  string nomearquivo;
44
45 public:
46  CSimulador();
47  ~CSimulador() { };
48  int Run();
49  void Opcao_argilosidade();
50  void Opcao_porosidade();
51  void Opcao_saturacao();
52  void Opcao_litologia();
53  void Opcao_perfis();
54  void PlotSolutionArgilosidade(vector<double> &datax, vector<double> &
    datay, string dados, string titulo);
55  void PlotSolutionLitologia(vector<double> &datax, vector<double> &
    datay, string dados, string titulo);
56  void PlotSolutionSaturacao(vector<double> &datax, vector<double> &
    datay, string dados, string titulo);
57  void PlotSolutionPorosidade(vector<double> &datax, vector<double> &
    datay, string dados, string titulo);
58  void PlotSolutionGamaRay(vector<double> &datax, vector<double> &datay,
    string dados, string titulo);
59  void PlotSolutionResistividade(vector<double> &datax, vector<double> &
    datay, string dados, string titulo);
60  void PlotSolutionDensidade(vector<double> &datax, vector<double> &
    datay, string dados, string titulo);
61  //void Config_Plot(string dados, CGnuplot plot);
62 };
63
64 #endif

```

Apresenta-se na listagem 6.2 o arquivo de implementação da classe CSimulador.

Listing 6.2: Arquivo de implementação da classe CSimulador.cpp

```

65 #include "CSimulador.h"
66
67 CSimulador :: CSimulador() {
68
69     porosidade = CPorosidade (&densidade, &profundidade);
70     argilosidade = CArgilosidade(&gamaray);
71     litologia = CLitologia (&gamaray , &profundidade);
72     normal = CSaturacaoArchieNormal (&porosidade,&resistividade);
73     modificada = CSaturacaoArchieModificada (&porosidade,&resistividade,&
        argilosidade);
74
75
76 }
77
78 int CSimulador :: Run ()
79 {
80     cout<<"_
        =====
        " <<endl;
81     cout<<"_Universidade_Estadual_do_Norte
        _Fluminense_"<<endl;
82     cout<<"_Simulador_Para_Estimativa_de_Propriedades_Petrofisicas_
        _a_partir_de_Perfis_Geofisicos_de_Pocos"<<endl;
83     cout<<"_ERICA_MELLO_
        LISBOA"<<endl;
84     cout<<"_POLLYANA_
        FERREIRA_DA_SILVA_"<<endl;
85     cout<<"_
        =====
        " <<endl;
86
87
88     char opcao;
89
90
91     cout << "Entre_com_o_nome_do_arquivo:_"<< endl;
92     cin >> nomearquivo;
93     nomearquivo = nomearquivo + ".dat";
94     densidade.EntradaDados(nomearquivo);
95     profundidade.EntradaDados(nomearquivo);
96     gamaray.EntradaDados(nomearquivo);
97     resistividade.EntradaDados(nomearquivo);
98     // o loop só parará quando uma das opcoes for correta.
99     do{
100         do {

```

```

101     cout << "Entre com a opção desejada" << endl;
102     cout << "1- Argilosidade" << endl;
103     cout << "2- Porosidade" << endl;
104     cout << "3- Saturacao" << endl;
105     cout << "4- Ver litologia" << endl;
106     cout << "5- Plotar perfis" << endl;
107     cout << "0- Quit" << endl;
108     cin >> opcao;
109
110     if ((opcao != '1') && (opcao != '2') && (opcao != '3') && (opcao != '4')
        && (opcao != '5') && (opcao != '0'))
111     { cout<< "Opcao invalida!"<< endl;};
112
113
114     }while ((opcao != '1') && (opcao != '2') && (opcao != '3') && (opcao !=
        '4') && (opcao != '5') && (opcao != '0') );
115
116
117     // quando a opção inserida for a correta ocorrerá uma das seguintes
        opções
118     switch (opcao)
119     {
120         case '1': Opcao_argilosidade();
121             break;
122
123         case '2': Opcao_porosidade();
124             break;
125
126         case '3': Opcao_saturacao();
127             break;
128
129         case '4': Opcao_litologia();
130             break;
131
132         case '5': Opcao_perfis();
133             break;
134
135         case '0': return 0;
136     };
137     }while ((opcao != '0')) ;
138
139 }
140
141 void CSimulador :: Opcao_argilosidade()
142 {
143     char opcao;
144     do{
145         cout << "Para calculo da argilosidade entre com o tipo de rocha" <<

```

```

        endl;
146  cout << "1-ParaRochaConsolidada" << endl;
147  cout << "2-ParaRochaNaoConsolidada" << endl;
148  cout << "3-FormulaqueNaoconsideraograu_deconsolidacao" << endl
        ;
149  cin >> opcao;
150  if ((opcao != '1') && (opcao != '2') && (opcao != '3'))
151  { cout<< "Opcao_invalida!"<< endl;
152    opcao='4';
153  }
154  else {opcao=opcao;};
155
156  }while (opcao=='4' );
157
158
159  // quando a opção inserida for a correta ocorrerá uma das seguintes
        opções
160  switch (opcao)
161  {
162      case '1': argilosidade.RochaConsolidada();
163          break;
164      case '2': argilosidade.RochaNaoConsolidada();
165          break;
166      case '3': argilosidade.FormaPratica();
167          break;
168  }
169
170  PlotSolutionArgilosidade(argilosidade.Get_VSH_Vector(), profundidade.
        Get_perfil_vector(),"Argilosidade", "Argilosidade_vs._Profundidade"
        );
171 }
172
173 void CSimulador :: Opcao_litologia()
174 {
175     litologia.IdentificarLitologias();
176     PlotSolutionLitologia(litologia.Get_Arenito_vector(), litologia.
        Get_ProfArenito_vector(),"Gama_Ray", "Gama_Ray_Arenito_vs._
        Profundidade");
177     PlotSolutionLitologia(litologia.Get_Folhelho_vector(), litologia.
        Get_ProfFolhelho_vector(),"Gama_Ray","Gama_Ray_Folhelho_vs._
        Profundidade");
178     cin.get();
179     cout << "\n";
180 }
181
182 void CSimulador :: Opcao_porosidade()
183 {
184     porosidade.CalcularPorosidade();

```

```

185 PlotSolutionPorosidade(porosidade.Get_FI_Vector(), profundidade.
    Get_perfil_vector(), "Porosidade", "Porosidade_vs._Profundidade");
186 porosidade.PorosidadeMedia();
187 cin.get();
188 cout << "\n";
189 }
190
191 void CSimulador :: Opcao_saturacao()
192 {
193     char opcao;
194     cout<< "Entre com a opcao desejada" <<endl;
195     cout<< "1-Saturacao de Archie" << endl;
196     cout<< "2-Saturacao de Archie com a Argilosidade" <<endl;
197     cin>> opcao;
198
199     porosidade.CalcularPorosidade();
200
201     if (opcao=='1')
202     {
203         normal.CalcularSaturacao();
204         PlotSolutionSaturacao(normal.Get_S0_Vector(), profundidade.
            Get_perfil_vector(), "Saturacao de Archie", "Saturacao_vs._
            Profundidade");
205         cin.get();
206         cout << "\n";
207     }
208     else
209     {
210         char opcao;
211         do{
212             cout << "Para calculo da argilosidade entre com o tipo de rocha" <<
                endl;
213             cout << "1-Para Rocha Consolidada" << endl;
214             cout << "2-Para Rocha Nao Consolidada" << endl;
215             cout << "3-Formula que Nao considera o grau de consolidacao" << endl
                ;
216             cin >> opcao;
217             if ((opcao != '1') && (opcao != '2') && (opcao != '3'))
218             { cout<< "Opcao invalida!"<< endl;
                opcao='4';
219             }
220             else {opcao=opcao;};
221
222         }while (opcao=='4' );
223
224
225
226     // quando a opção inserida for a correta ocorrerá uma das seguintes
        opções

```

```

227  switch (opcao)
228  {
229      case '1': argilosidade.RochaConsolidada();
230          break;
231      case '2': argilosidade.RochaNaoConsolidada();
232          break;
233      case '3': argilosidade.FormaPratica();
234          break;
235  }
236  modificada.CalcularSaturacao();
237  PlotSolutionSaturacao(modificada.Get_SO_Vector(), profundidade.
      Get_perfil_vector(),"Saturacao_de_Archie_com_a_Argilosidade", "
      Saturacao_vs_Profundidade");
238  cin.get();
239  cout << "\n";
240 };
241
242 }
243
244
245 void CSimulador :: Opcao_perfis()
246 {
247     PlotSolutionGamaRay(gamaray.Get_perfil_vector(), profundidade.
        Get_perfil_vector(),"Gama_Ray","Gama_Ray_vs_Profundidade");
248     PlotSolutionResistividade(resistividade.Get_perfil_vector(),
        profundidade.Get_perfil_vector(),"Resistividade","Resistividade_vs.
        Profundidade");
249     PlotSolutionDensidade(densidade.Get_perfil_vector(), profundidade.
        Get_perfil_vector(),"Densidade","Densidade_vs_Profundidade");
250     cin.get();
251     cout << "\n";
252 }
253
254
255
256 void CSimulador :: PlotSolutionArgilosidade(vector<double> &datax,
        vector<double> &datay, string dados, string titulo)
257 {
258     plotArgilosidade.set_style("lines");
259     int i= 0;
260     double in = profundidade.Perfil(i);
261     i = profundidade.NumPontos();
262     double on = profundidade.Perfil(i-2);
263     plotArgilosidade.set_yrange(on,in);
264     plotArgilosidade.set_grid();
265     plotArgilosidade.set_xlabel(dados);
266     plotArgilosidade.set_ylabel("Profundidade");
267     plotArgilosidade.set_pointsize(1.2);

```



```
268     plotArgilosidade.plot_xy(datax , datay, titulo );
269 }
270
271 void CSimulador :: PlotSolutionPorosidade(vector<double> &datax, vector<
    double> &datay, string dados, string titulo)
272 {
273     plotPorosidade.set_style("lines");
274     int i= 0;
275     double in = profundidade.Perfil(i);
276     i = profundidade.NumPontos();
277     double on = profundidade.Perfil(i-2);
278     plotPorosidade.set_yrange(on,in);
279     plotPorosidade.set_grid();
280     plotPorosidade.set_xlabel(dados);
281     plotPorosidade.set_ylabel("Profundidade");
282     plotPorosidade.set_pointsize(1.2);
283     plotPorosidade.plot_xy(datax , datay, titulo );
284 }
285
286 void CSimulador :: PlotSolutionSaturacao(vector<double> &datax, vector<
    double> &datay, string dados, string titulo)
287 {
288     plotSaturacao.set_style("lines");
289     plotSaturacao.set_xrange(0.0, 1.0);
290     int i= 0;
291     double in = profundidade.Perfil(i);
292     i = profundidade.NumPontos();
293     double on = profundidade.Perfil(i-2);
294     plotSaturacao.set_yrange(on,in);
295     plotSaturacao.set_grid();
296     plotSaturacao.set_xlabel(dados);
297     plotSaturacao.set_ylabel("Profundidade");
298     plotSaturacao.set_pointsize(1.2);
299     plotSaturacao.plot_xy(datax , datay, titulo );
300 }
301
302 void CSimulador :: PlotSolutionGamaRay(vector<double> &datax, vector<
    double> &datay, string dados, string titulo)
303 {
304     plotGamaRay.set_style("lines");
305     int i= 0;
306     double in = profundidade.Perfil(i);
307     i = profundidade.NumPontos();
308     double on = profundidade.Perfil(i-2);
309     plotGamaRay.set_yrange(on,in);
310     plotGamaRay.set_grid();
311     plotGamaRay.set_xlabel(dados);
312     plotGamaRay.set_ylabel("Profundidade");
```

```

313     plotGamaRay.set_pointsize(1.2);
314     plotGamaRay.plot_xy(datax , datay, titulo );
315 }
316
317 void CSimulador :: PlotSolutionResistividade(vector<double> &datax,
        vector<double> &datay, string dados, string titulo)
318 {
319     int i= 0;
320     double in = profundidade.Perfil(i);
321     i = profundidade.NumPontos();
322     double on = profundidade.Perfil(i-2);
323     plotResistividade.set_yrange(on,in);
324     const double base = 10;
325     plotResistividade.set_xlogscale (base);
326     plotResistividade.set_style("lines");
327     plotResistividade.set_grid();
328     plotResistividade.set_xlabel(dados);
329     plotResistividade.set_ylabel("Profundidade");
330     plotResistividade.set_pointsize(1.2);
331     plotResistividade.plot_xy(datax , datay, titulo );
332 }
333
334 void CSimulador :: PlotSolutionDensidade(vector<double> &datax, vector<
        double> &datay, string dados, string titulo)
335 {
336     plotDensidade.set_style("lines");
337     int i= 0;
338     double in = profundidade.Perfil(i);
339     i = profundidade.NumPontos();
340     double on = profundidade.Perfil(i-2);
341     plotDensidade.set_yrange(on,in);
342     plotDensidade.set_grid();
343     plotDensidade.set_xlabel(dados);
344     plotDensidade.set_ylabel("Profundidade");
345     plotDensidade.set_pointsize(1.2);
346     plotDensidade.plot_xy(datax , datay, titulo );
347 }
348
349 void CSimulador :: PlotSolutionLitologia(vector<double> &datax, vector<
        double> &datay, string dados, string titulo)
350 {
351     plotLitologia.set_style("points");
352     int i= 0;
353     double in = profundidade.Perfil(i);
354     i = profundidade.NumPontos();
355     double on = profundidade.Perfil(i-2);
356     plotLitologia.set_yrange(on,in);
357     plotLitologia.set_grid();

```

```

358     plotLitologia.set_xlabel(dados);
359     plotLitologia.set_ylabel("Profundidade");
360     plotLitologia.set_pointsize(1.2);
361     plotLitologia.plot_xy(datax , datay, titulo );
362 }

```

Apresenta-se na listagem 6.3 o arquivo com código da classe CPerfil.

Listing 6.3: Arquivo de cabeçalho da classe CPerfil.h

```

363 #ifndef CPerfil_H
364 #define CPerfil_H
365 #include <iostream>
366 #include <vector>
367 #include <string>
368
369 using namespace std;
370
371 //Esta classe serve para armazenar os valores das profundidades e dos
perfis de GamaRay, Resistividade e Densidade.
372
373 class CPerfil
374 {
375 {
376
377 protected:
378     int numPontos;
379     vector <double> perfil;
380     double maximo;
381     double minimo;
382
383
384 public:
385
386     CPerfil() {}; //Construtor Default
387     ~CPerfil() {}; //Destrutor Default
388
389     void Print_Dados();
390     int NumPontos () {return numPontos;};
391     void Set_numPontos (int _i) {numPontos=_i;};
392     vector <double> & Get_perfil_vector() {return perfil;};
393     double Perfil (int indice) {return perfil[indice];};
394
395     double Get_Minimo() {return minimo;};
396     double Get_Maximo() {return maximo;};
397     void Set_Minimo();
398     void Set_Maximo();
399
400
401 };

```

```

402
403 #endif

```

Apresenta-se na listagem 6.4 o arquivo de implementação da classe CPerfil.

Listing 6.4: Arquivo de implementação da classe CPerfil.cpp

```

404 #include "CPerfil.h"
405 #include "CGamaRay.h"
406 #include "CResistividade.h"
407 #include "CDensidade.h"
408 #include "CProfundidade.h"
409 #include <iostream>
410 #include <fstream>
411 #include <vector>
412 #include <string>
413
414
415 using namespace std;
416
417
418 void CPerfil :: Print_Dados ()
419 {
420
421     cout << "Imprimindo o vetor na tela" << endl;
422     for (int i=0; i< numPontos ; i++)
423     {cout << i << "          " << perfil[i]<< endl;};
424 }
425
426 void CPerfil :: Set_Maximo()
427 {
428     double temporaria = 0.0;
429     for (int i =1 ; i < numPontos; i++)
430     {
431         if (perfil[i]>temporaria)
432         { maximo=perfil[i];
433           temporaria=maximo;};
434
435     };
436 }
437
438 void CPerfil :: Set_Minimo()
439 {
440     double temporaria = 1000.0;
441     for (int i =1 ; i < numPontos; i++)
442     {
443         if (perfil[i]<temporaria)
444         {minimo=perfil[i]; temporaria=minimo;};
445     };
446 }

```

Apresenta-se na listagem 6.5 o arquivo com código da classe CProfundidade.

Listing 6.5: Arquivo de cabeçalho da classe CProfundidade.h

```

447 #ifndef CProfundidade_H
448 #define CProfundidade_H
449 #include "CPerfil.h"
450 #include <vector>
451 #include <string>
452
453 using namespace std;
454
455 class CProfundidade : public CPerfil
456 {
457     public:
458         CProfundidade() {};           //construtor default
459         ~CProfundidade() {};          //destrutor default
460         void EntradaDados(string nomearquivo);
461 };
462
463 #endif

```

Apresenta-se na listagem 6.6 o arquivo de implementação da classe CProfundidade.

Listing 6.6: Arquivo de implementação da classe CProfundidade.cpp

```

465 #include "CProfundidade.h"
466 #include <iostream>
467 #include <vector>
468 #include <cmath>
469 #include <fstream>
470 #include <string>
471
472 using namespace std;
473
474 void CProfundidade :: EntradaDados (string nomearquivo)
475 {
476     perfil.clear();
477     ifstream fin;
478     fin.open (nomearquivo.c_str());
479     int i = 0 ;
480     double tmp;
481     cout << "Profundidade_□importada"<<endl;
482     do {
483         fin>>tmp;
484         perfil.push_back(tmp);
485         fin>>tmp;
486         fin>>tmp;
487         fin>>tmp;
488         fin>>tmp;
489         fin>>tmp;

```

```

490 } while(!fin.eof());
491 Set_numPontos(perfil.size());
492 Set_Maximo();
493 Set_Minimo();
494 }

```

Apresenta-se na listagem 6.7 o arquivo com código da classe CGamaRay.

Listing 6.7: Arquivo de cabeçalho da classe CGamaRay.h

```

495 #ifndef CGamaRay_H
496 #define CGamaRay_H
497 #include "CPerfil.h"
498 #include <vector>
499 #include <iostream>
500 #include <string>
501
502 /* Esta classe representa os valores das medidas do perfil de raio gama
    **/
503
504 using namespace std;
505
506 class CGamaRay : public CPerfil
507 {
508 public:
509     CGamaRay() {};           //construtor default
510     ~CGamaRay() {};         //destrutor default
511     void EntradaDados(string nomearquivo);
512 };
513
514 #endif

```

Apresenta-se na listagem 6.8 o arquivo de implementação da classe CGamaRay.

Listing 6.8: Arquivo de implementação da classe CGamaRay.cpp

```

515 #include "CGamaRay.h"
516 #include <iostream>
517 #include <vector>
518 #include <cmath>
519 #include <fstream>
520 #include <string>
521
522
523 using namespace std;
524
525 void CGamaRay :: EntradaDados (string nomearquivo)
526 { perfil.clear();
527   ifstream fin;
528   fin.open (nomearquivo.c_str());
529   int i = 0 ;

```

```

530 double tmp;
531 cout << "Gama_Ray_importado"<<endl;
532 do
533 {
534     if (fin.eof()) break;
535     else
536     {
537         fin>>tmp;
538         fin>>tmp;
539         perfil.push_back(tmp);
540         fin>>tmp;
541         fin>>tmp;
542         fin>>tmp;
543         fin>>tmp;
544         i++;};
545
546 }while(!fin.eof()) ;
547
548 Set_numPontos(i);
549 Set_Maximo();
550 Set_Minimo();
551 }

```

Apresenta-se na listagem 6.9 o arquivo com código da classe CResistividade.

Listing 6.9: Arquivo de cabeçalho da classe CResistividade.h

```

552 #ifndef CResistividade_H
553 #define CResistividade_H
554 #include "CPerfil.h"
555 #include <vector>
556 #include <string>
557
558 //rt = resistividade total ; rw = resistividade da água (dado input)
559 /* Esta classe representa os valores das medidas do perfil de
560 resistividade**/
561
562 using namespace std;
563
564 class CResistividade : public CPerfil
565 {
566 public:
567     CResistividade() {}; //construtor default
568     ~CResistividade() {}; //destrutor default
569     void EntradaDados (string nomearquivo);
570 };
571 #endif

```

Apresenta-se na listagem 6.10 o arquivo de implementação da classe CResistividade.

Listing 6.10: Arquivo de implementação da classe CResistividade.cpp

```

574#include "CResistividade.h"
575#include <iostream>
576#include <vector>
577#include <cmath>
578#include <fstream>
579#include <string>
580
581
582/*
583using namespace std;*/
584
585void CResistividade :: EntradaDados (string nomearquivo)
586{
587    perfil.clear();
588    ifstream fin;
589    fin.open (nomearquivo.c_str());
590    int i = 0 ;
591    double tmp;
592    cout << "Resistividade_Importada"<<endl;
593    do
594    { if (fin.eof()) break;
595      else
596      {
597          fin>>tmp;
598          fin>>tmp;
599          fin>>tmp;
600          fin>>tmp;
601          perfil.push_back(tmp);
602          fin>>tmp;
603          fin>>tmp;
604          i++;};
605    }while(!fin.eof()) ;
606    Set_numPontos(i);
607    Set_Maximo();
608    Set_Minimo();
609
610}

```

Apresenta-se na listagem 6.11 o arquivo com código da classe CDensidade.

Listing 6.11: Arquivo de cabeçalho da classe CDensidade.h

```

611#ifndef CDensidade_H
612#define CDensidade_H
613#include "CPerfil.h"
614#include <vector>
615#include <iostream>
616#include <string>

```



```

617
618 // ro = densidade
619
620 using namespace std;
621
622 class CDensidade : public CPerfil
623 {
624     public:
625         CDensidade() {};           // construtor default
626         ~CDensidade() {};         // destrutor default
627         void EntradaDados(string nomearquivo);
628 };
629 #endif

```

Apresenta-se na listagem 6.12 o arquivo de implementação da classe CDensidade.

Listing 6.12: Arquivo de implementação da classe CDensidade.cpp

```

630 #include "CDensidade.h"
631 #include <iostream>
632 #include <fstream>
633 #include <vector>
634 #include <string>
635 // #include <cmath>
636
637
638 using namespace std;
639
640 void CDensidade::EntradaDados(string nomearquivo)
641 {
642     perfil.clear();
643     ifstream fin;
644     fin.open (nomearquivo.c_str());
645     int i = 0 ;
646     double tmp;
647     cout << "Densidade_Importada"<<endl;
648     do
649     {
650         if (fin.eof()) break;
651         else
652         {
653             {fin>>tmp;
654             fin>>tmp;
655             fin>>tmp;
656             fin>>tmp;
657             fin>>tmp;
658             fin>>tmp;
659             perfil.push_back(tmp);
660             i++;};
661         }while(!fin.eof()) ;
662     }
663     Set_numPontos(i);
664     Set_Maximo();
665     Set_Minimo();

```

662 }

Apresenta-se na listagem 6.13 o arquivo com código da classe CLitologia.

Listing 6.13: Arquivo de cabeçalho da classe CLitologia.h

```

663 #ifndef CLitologia_H
664 #define CLitologia_H
665 #include <vector>
666 #include "CGamaRay.h"
667 #include "CProfundidade.h"
668
669
670 /* Esta classe plota o grafico de GamaRay especificando onde está cada
    litologia */
671
672
673 class CLitologia
674 {
675     protected:
676         CGamaRay* gamaray;
677         CProfundidade* profundidade;
678         vector <double> litoArenito;
679         vector <double> litoFolhelho;
680         vector <double> profArenito;
681         vector <double> profFolhelho;
682
683     public:
684         CLitologia(){};          //Construtor Default
685         CLitologia(CGamaRay* gamaray_ent, CProfundidade* profundidade_ent);
686         ~CLitologia(){};        //Destrutor Default
687         void IdentificarLitologias();
688         double Get_Arenito(int indice) {return litoArenito[indice];};
689         vector <double> & Get_Arenito_vector () {return litoArenito;};
690         double Get_Folhelho(int indice) {return litoFolhelho[indice];};
691         vector <double> & Get_Folhelho_vector () {return litoFolhelho;};
692         double Get_ProfArenito(int indice) {return profArenito[indice];};
693         vector <double> & Get_ProfArenito_vector () {return profArenito;};
694         double Get_ProfFolhelho(int indice) {return profFolhelho[indice];};
695         vector <double> & Get_ProfFolhelho_vector () {return profFolhelho;};
696
697 };
698 #endif

```

Apresenta-se na listagem 6.14 o arquivo de implementação da classe CLitologia.

Listing 6.14: Arquivo de implementação da classe CLitologia.cpp

```

699 #include "CLitologia.h"
700 // #include "CGamaRay.h"
701 #include <vector>

```

```

702
703 using namespace std;
704
705 CLitologia :: CLitologia(CGamaRay* gamaray_ent, CProfundidade*
    profundidade_ent)
706 {
707     profundidade = profundidade_ent;
708     gamaray = gamaray_ent;
709 }
710
711 void CLitologia :: IdentificarLitologias()
712 {
713     litoArenito.clear();
714     profArenito.clear();
715     litoFolhelho.clear();
716     profFolhelho.clear();
717     for (int i = 0; i < gamaray->NumPontos(); i++)
718     {
719         if (gamaray->Perfil(i) < 90.0)
720         {
721             litoArenito.push_back(gamaray->Perfil(i));
722             profArenito.push_back(profundidade->Perfil(i));
723         }
724         else
725         {
726             litoFolhelho.push_back(gamaray->Perfil(i));
727             profFolhelho.push_back(profundidade->Perfil(i));
728         }
729     }
730 }

```

Apresenta-se na listagem 6.15 o arquivo com código da classe CArgilosidade.

Listing 6.15: Arquivo de cabeçalho da classe CArgilosidade.h

```

731 #ifndef CArgilosidade_H
732 #define CArgilosidade_H
733 #include <vector>
734 #include "CGamaRay.h"
735
736
737 //Esta classe guarda o valor da argilosidade calculada de acordo com a
formula desejada, referente aos três métodos de cálculo
738
739
740 class CArgilosidade
741 {
742     protected:
743         vector <double> argilosidade;
744         CGamaRay* gamaray;

```

```

745
746 public:
747     CArgilosidade(){};           //Construtor Default
748     CArgilosidade(CGamaRay* gamaray_ent);
749     ~CArgilosidade(){};         //Destrutor Default
750     //Calcula a argilosidade usando os valores maximo e mínimo de Raio
        Gama
751     void FormaPratica();
752     //Calcula a argilosidade usando uma fórmula específica para rochas
        consolidadas
753     void RochaConsolidada();
754     //Calcula a argilosidade usando uma fórmula específica para rochas
        não consolidadas
755     void RochaNaoConsolidada();
756     double Get_VSH(int indice){return argilosidade[indice];};
757     vector <double> & Get_VSH_Vector() {return argilosidade;};
758 };
759 #endif

```

Apresenta-se na listagem 6.16 o arquivo de implementação da classe CArgilosidade.

Listing 6.16: Arquivo de implementação da classe CArgilosidade.cpp

```

760 #include "CArgilosidade.h"
761 #include <iostream>
762 #include <vector>
763 #include <cmath>
764
765
766 using namespace std;
767
768 CArgilosidade :: CArgilosidade(CGamaRay* gamaray_ent)
769 {
770     gamaray = gamaray_ent;
771 }
772
773 void CArgilosidade :: FormaPratica ()
774 {
775     argilosidade.clear();
776     //loop para calcular a argilosidade em cada profundidade e preencher
        no vetor argilosidade
777     for (int i= 0; i< gamaray->NumPontos() ; i++)
778     { argilosidade.push_back((gamaray->Perfil(i) - gamaray->Get_Minimo())
        /(gamaray->Get_Maximo() - gamaray->Get_Minimo()));};
779 }
780
781
782 void CArgilosidade :: RochaConsolidada ()
783 {
784     argilosidade.clear();

```

```

785 //loop para calcular a argilosidade em cada profundidade e preencher
    no vetor argilosidade
786 for (int i= 0; i< gamaray->NumPontos() ; i++)
787 { argilosidade.push_back((gamaray->Perfil(i) - gamaray->Get_Minimo())
    /(gamaray->Get_Maximo() - gamaray->Get_Minimo()));
788   argilosidade[i]=( 0.33 *((pow(2,2*argilosidade[i])) - 1));
789 };
790 }
791
792
793 void CArgilosidade :: RochaNaoConsolidada ()
794 {
795   argilosidade.clear();
796   //loop para calcular a argilosidade em cada profundidade e preencher
    no vetor argilosidade
797   for (int i= 0; i< gamaray->NumPontos() ; i++)
798   { argilosidade.push_back((gamaray->Perfil(i) - gamaray->Get_Minimo())
    /(gamaray->Get_Maximo() - gamaray->Get_Minimo()));
799     argilosidade[i]=(0.083 * pow(2,3.7*argilosidade[i])-0.083) ;
800   };
801 }

```

Apresenta-se na listagem 6.17 o arquivo com código da classe CPorosidade.

Listing 6.17: Arquivo de cabeçalho da classe CPorosidade.h

```

802 #ifndef CPorosidade_H
803 #define CPorosidade_H
804 #include <vector>
805 #include "CDensidade.h"
806 #include "CProfundidade.h"
807
808
809 //FI - porosidade
810 /* Esta classe calcula a porosidade através dos valores máximos e
    mínimos da densidade, utilizando para isso dados da classe CDensidade
    . */
811 using namespace std;
812
813 class CPorosidade
814 {
815   protected:
816     CDensidade * densidade;
817     CProfundidade* profundidade;
818     vector <double> porosidade;
819     double porosidadeMedia;
820
821   public:
822     CPorosidade() {};
```

//Construtor Default

```

824     CPorosidade(CDensidade* densidade_ent, CProfundidade*
           profundidade_ent);
825     ~CPorosidade() {};           //Destrutor Default
826     void CalcularPorosidade ();           //Calcula a porosidade
827     double Get_FI(int indice) {return porosidade[indice];};
828     vector <double> & Get_FI_Vector () {return porosidade;};
829     void PorosidadeMedia ();
830
831 };
832
833 #endif

```

Apresenta-se na listagem 6.18 o arquivo de implementação da classe CPorosidade.

Listing 6.18: Arquivo de implementação da classe CPorosidade.cpp

```

834 #include "CPorosidade.h"
835 #include "CPorosidade.h"
836 #include <iostream>
837 #include <vector>
838 #include <cmath>
839
840
841 using namespace std;
842
843 CPorosidade :: CPorosidade(CDensidade* densidade_ent, CProfundidade*
           profundidade_ent)
844 {
845     densidade = densidade_ent;
846     profundidade = profundidade_ent;
847 }
848
849 void CPorosidade :: CalcularPorosidade ()
850 {
851     porosidade.clear();
852     for(int i=0; i< densidade->NumPontos(); i++)
853     {
854         if (densidade->Get_Maximo() == densidade->Perfil(i))
855             porosidade.push_back(0.05);
856         else
857             porosidade.push_back((densidade->Get_Maximo() - densidade->Perfil(i)
                                   )/(densidade->Get_Maximo() - densidade->Get_Minimo()));
858     };
859
860 }
861
862 void CPorosidade :: PorosidadeMedia ()
863 {
864     double prof_inicial;
865     double prof_final;

```

```

866     int j=0;
867     int cont=0;
868     cout << "\nEntre com a profundidade inicial da area desejada: ";
869     cin >> prof_inicial;
870     cout << "\nEntre com o fim da profundidade da area desejada: ";
871     cin >> prof_final;
872
873     for (int i = 0; i < profundidade->NumPontos() ;i++)
874     {
875         if ((profundidade->Perfil(i)) > prof_inicial)
876             { j=i; break; }
877     }
878
879     porosidadeMedia = 0.0;
880     do {
881         porosidadeMedia = porosidadeMedia + porosidade[j];
882         prof_inicial = profundidade->Perfil(j);
883         j++;
884         cont++;
885     } while (prof_inicial < prof_final);
886
887     porosidadeMedia = porosidadeMedia/cont;
888     cout << "A Porosidade media da area desejada e: "<< porosidadeMedia
889         << endl;
890 }

```

Apresenta-se na listagem 6.19 o arquivo com código da classe CSaturacaoArchieNormal.

Listing 6.19: Arquivo de cabeçalho da classe CSaturacaoArchieNormal.h

```

893 #ifndef CSaturacaoArchieNormal_H
894 #define CSaturacaoArchieNormal_H
895 #include <vector>
896 #include "CResistividade.h"
897 #include "CPorosidade.h"
898
899
900 /* Esta classe recebe os atributos e os métodos das classes
901    CSaturacaoArchieNormal e CArgilosidade para calcular a saturação de
902    óleo e de água */
903 using namespace std;
904
905
906 class CSaturacaoArchieNormal
907 {
908     protected:
909     CPorosidade* porosidade;
910     CResistividade* resistividade;
911     double a, m, n, rw ;
912     vector <double> so;

```

```

910     vector <double> sw;
911
912 public:
913     CSaturacaoArchieNormal();           //Construtor Default
914     CSaturacaoArchieNormal(CPorosidade* porosidade_ent ,CResistividade*
        resistividade_ent);
915     ~CSaturacaoArchieNormal();         //Destrutor Default
916     void CalcularSaturacao();           //Este método calcula a saturação de
        óleo e água utilizando a lei de Archie Normal
917     double Get_SO(int indice) {return so[indice];};
918     vector <double> & Get_SO_Vector() {return so;};
919
920 };
921 #endif

```

Apresenta-se na listagem 6.20 o arquivo de implementação da classe CSaturacaoArchieNormal.

Listing 6.20: Arquivo de implementação da classe CSaturacaoArchieNormal.cpp

```

924 #include <iostream>
925 #include <vector>
926 #include <string>
927 #include <cmath>
928 #include "CSaturacaoArchieNormal.h"
929
930
931 using namespace std;
932
933 CSaturacaoArchieNormal :: CSaturacaoArchieNormal() { }
934
935 CSaturacaoArchieNormal:: CSaturacaoArchieNormal(CPorosidade*
        porosidade_ent ,CResistividade* resistividade_ent)
936 {
937     porosidade = porosidade_ent;
938     resistividade = resistividade_ent;
939 }
940
941 CSaturacaoArchieNormal :: ~CSaturacaoArchieNormal() { }
942
943 void CSaturacaoArchieNormal :: CalcularSaturacao()
944 {
945     sw.clear();
946     so.clear();
947
948     cout << "Insira o coeficiente de tortuosidade a=";
949     cin >> a;
950     cout << endl;
951     cout << "Insira o coeficiente de cimentacao m=" ;

```



```

952  cin >> m;
953  cout << endl;
954  cout << "Insira o coeficiente de saturacao n=" ;
955  cin >> n;
956  cout << endl;
957  cout << "Entre com o valor da resistividade da agua Rw=";
958  cin >> rw;
959  cout << endl;
960  double s=0.0;
961  for(int i=0; i < resistividade->NumPontos(); i++)
962  {
963      sw.push_back(pow(((a*rw)/(resistividade->Perfil(i)*pow(porosidade
          ->Get_FI(i),m))), (1.0/n))) ;
964      if (sw[i]>1.0)
965          sw[i]=1.0;
966      s=1.0-sw[i];
967      so.push_back(s);
968  }
969 }

```

Apresenta-se na listagem 6.21 o arquivo com código da classe CSaturacaoArchieModificada.

Listing 6.21: Arquivo de cabeçalho da classe CSaturacaoArchieModificada.h

```

970 #ifndef CSaturacaoArchieModificada_H
971 #define CSaturacaoArchieModificada_H
972 #include <vector>
973 #include "CSaturacaoArchieNormal.h"
974 #include "CArgilosidade.h"
975
976
977 /* Esta classe recebe os atributos e os métodos das classes
          CSaturacaoArchieNormal e CArgilosidade para calcular a saturação de
          óleo e de água */
978 using namespace std;
979
980 class CSaturacaoArchieModificada : public CSaturacaoArchieNormal
981 {
982     protected:
983         CArgilosidade* argilosidade;
984         double rsh;
985
986     public:
987         CSaturacaoArchieModificada();           //Construtor Default
988         CSaturacaoArchieModificada(CPorosidade* porosidade_ent ,
          CResistividade* resistividade_ent, CArgilosidade* argilosidade_ent
          );
989         ~CSaturacaoArchieModificada();         //Destrutor Default

```

```

990     void CalcularSaturacao();
991
992
993 };
994 #endif

```

Apresenta-se na listagem 6.22 o arquivo de implementação da classe CSaturacaoArchieModificada.

Listing 6.22: Arquivo de implementação da classe CSaturacaoArchieModificada.cpp

```

996 #include <iostream>
997 #include <vector>
998 #include <string>
999 #include <cmath>
1000 #include "CSaturacaoArchieNormal.h"
1001 #include "CSaturacaoArchieModificada.h"
1002 #include "CArgilosidade.h"
1003 #include "CPorosidade.h"
1004
1005
1006 using namespace std;
1007
1008 CSaturacaoArchieModificada :: CSaturacaoArchieModificada() { }
1009
1010 CSaturacaoArchieModificada :: CSaturacaoArchieModificada(CPorosidade*
        porosidade_ent ,CResistividade* resistividade_ent,CArgilosidade*
        argilosidade_ent)
1011 {
1012     porosidade = porosidade_ent;
1013     resistividade = resistividade_ent;
1014     argilosidade = argilosidade_ent;
1015 }
1016
1017 CSaturacaoArchieModificada :: ~CSaturacaoArchieModificada() { }
1018
1019
1020 void CSaturacaoArchieModificada:: CalcularSaturacao()
1021 {
1022     sw.clear();
1023     so.clear();
1024
1025     cout << "Insira o coeficiente de tortuosidade a=";
1026     cin >> a;
1027     cout << endl;
1028     cout << "Insira o coeficiente de cimentacao m=" ;
1029     cin >> m;
1030     cout << endl;
1031     cout << "Insira o coeficiente de saturacao n=" ;

```

```

1032  cin >> n;
1033  cout << endl;
1034  cout << "Entre com o valor da resistividade da agua Rw=";
1035  cin >> rw;
1036  cout << endl;
1037  cout << "Entre com a resistividade da argila Rsh=";
1038  cin >> rsh ;
1039  cout << endl;
1040
1041
1042
1043  double erro = 0.00;
1044  double x = 0.0;
1045  double y = 0.0;
1046  double s = 0.1;
1047  double s_old = 0.1;
1048
1049  for (int i = 0 ; i < resistividade->NumPontos() ; i ++ ) {
1050      s = 0.1;
1051      do
1052      {
1053          s_old = s;
1054          y = 1.0/resistividade->Perfil(i);
1055          x = ((pow (porosidade->Get_FI(i),m) * pow (s_old,n))/(a*rw));
1056          s = (x - y)*argilosidade->Get_VSH(i)/rsh;
1057          erro = s - s_old;
1058
1059      }while ((abs(erro)>0.001) and (s > 1.00));
1060      sw.push_back(s);
1061      so.push_back(1 -sw[i]);
1062
1063  }
1064 }

```

Apresenta-se na listagem 6.23 o programa que usa a classe main.

Listing 6.23: Arquivo de implementação da função main().

```

1065 #include <iostream>
1066 #include <fstream>
1067 #include <vector>
1068 #include <string>
1069 #include "CSimulador.h"
1070
1071 using namespace std;
1072
1073 int main ()
1074 {
1075     CSimulador simulador;
1076     simulador.Run();

```

1077 }

# Capítulo 7

## Teste

Todo projeto de engenharia passa por uma etapa de testes. Neste capítulo apresentamos alguns testes do software desenvolvido. Estes testes devem dar resposta aos diagramas de caso de uso inicialmente apresentados (diagramas de caso de uso geral e específicos).

### 7.1 Teste 1: Interface

O presente trabalho apresenta interface em modo texto. Veja na Figura 7.1 a tela inicial do software, na qual o usuário entra com o nome do arquivo e informa a opção que deseja analisar.

```
=====
                                Universidade Estadual do Norte Fluminense
                                Simulador Para Estimativa de Propriedades Petrofisicas a partir de Perfis Geofisicos de Pocos
                                ERICA MELLO LISBOA
                                POLLYANA FERREIRA DA SILVA
=====
Entre com o nome do arquivo :
NA01A
Densidade importada
Profundidade importada
Gama Ray importado
Resistividade importada
Entre com a opcao desejada
1 - Argilosidade
2 - Porosidade
3 - Saturacao
4 - Ver litologia
5 - Plotar perfis
0 - Quit
█
```

Figura 7.1: Início do software

### 7.2 Teste 2: Plotar Perfis

O primeiro grupo de dados é apresentado a seguir e foram inseridos através da leitura de um arquivo com extensão .dat. Como saída tem-se os perfis plotados, Figura 7.2.

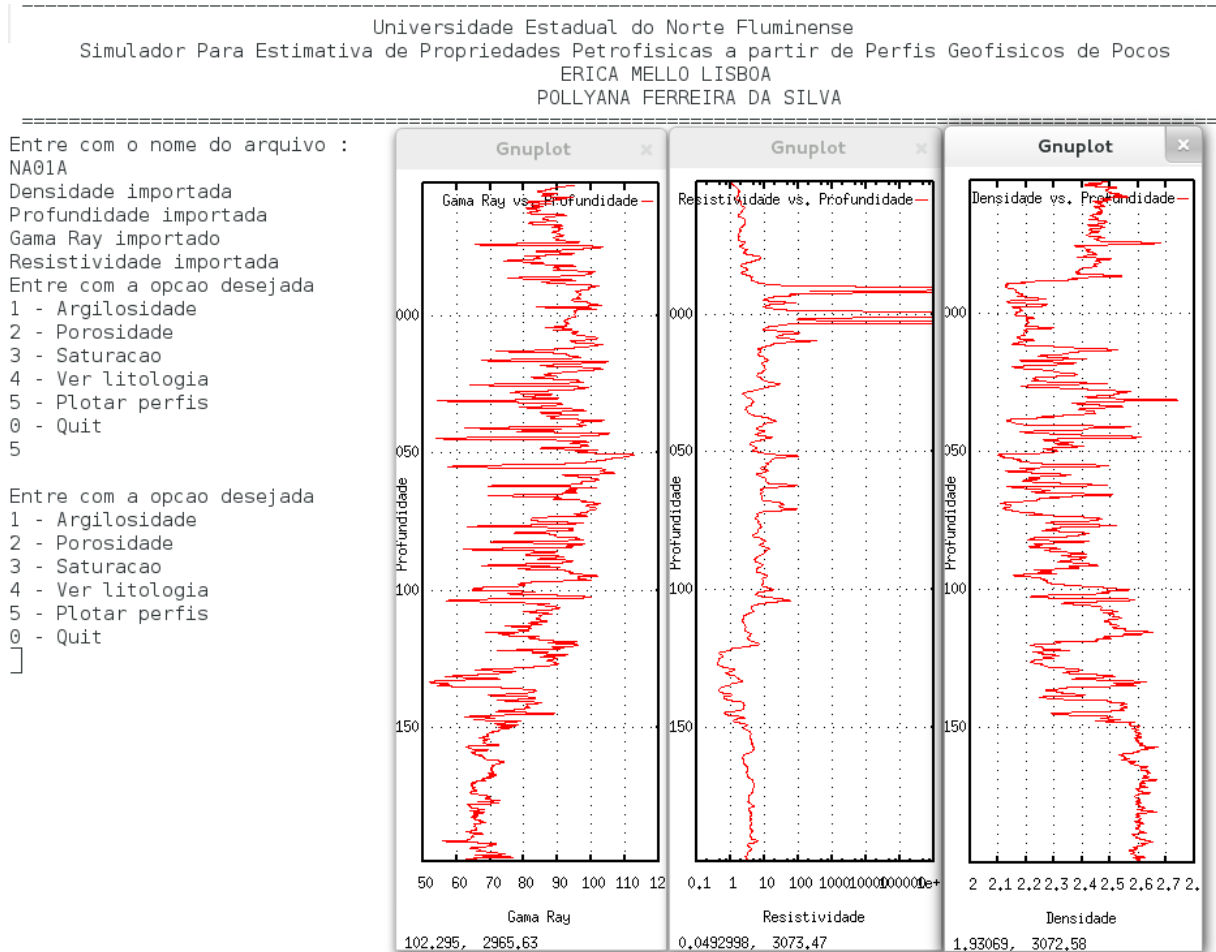


Figura 7.2: Tela do programa que mostra os perfis plotados (densidade, resistividade e radiação gama)

### 7.3 Teste 2: Cálculo da Argilosidade

1. Cálculo da Argilosidade para rocha consolidada, Figura 7.3;
2. Cálculo da Argilosidade para rocha não consolidada, Figura 7.4;
3. Cálculo da Argilosidade que não leva em consideração o grau de consolidação da rocha, Figura 7.5.

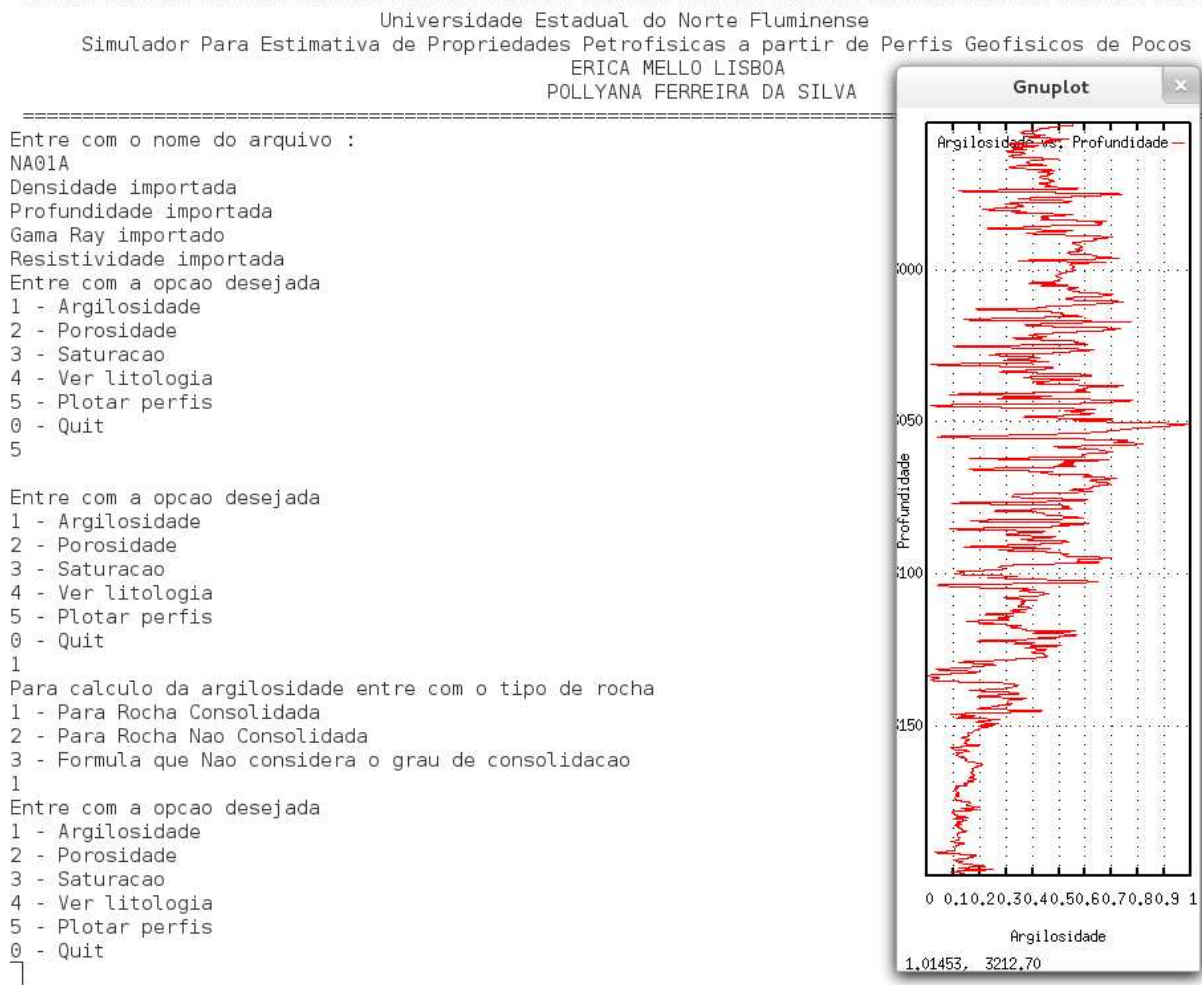


Figura 7.3: Plote da Argilosidade – Rocha Consolidada

```
[lenep@localhost cpp]$ ./a.out
```

```
=====
Universidade Estadual do Norte Fluminense
Simulador Para Estimativa de Propriedades Petrofisicas a partir de Perfis Geofisicos de Pocos
ERICA MELLO LISBOA
POLLYANA FERREIRA DA SILVA
=====
```

```
Entre com o nome do arquivo :
NA01A
Densidade importada
Profundidade importada
Gama Ray importado
Resistividade importada
Entre com a opcao desejada
1 - Argilosidade
2 - Porosidade
3 - Saturacao
4 - Ver litologia
5 - Plotar perfis
0 - Quit
1
Para calculo da argilosidade entre com o tipo de rocha
1 - Para Rocha Consolidada
2 - Para Rocha Nao Consolidada
3 - Formula que Nao considera o grau de consolidacao
2
Entre com a opcao desejada
1 - Argilosidade
2 - Porosidade
3 - Saturacao
4 - Ver litologia
5 - Plotar perfis
0 - Quit
0
```

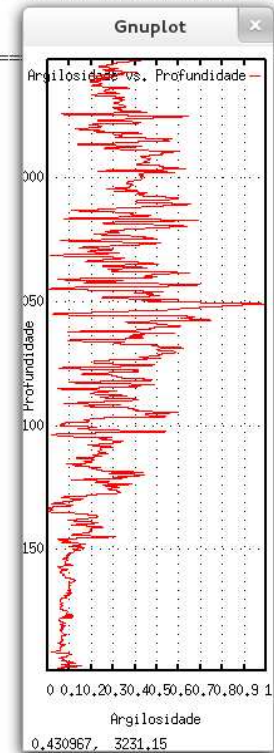


Figura 7.4: Plote da Argilosidade – Rocha não Consolidada



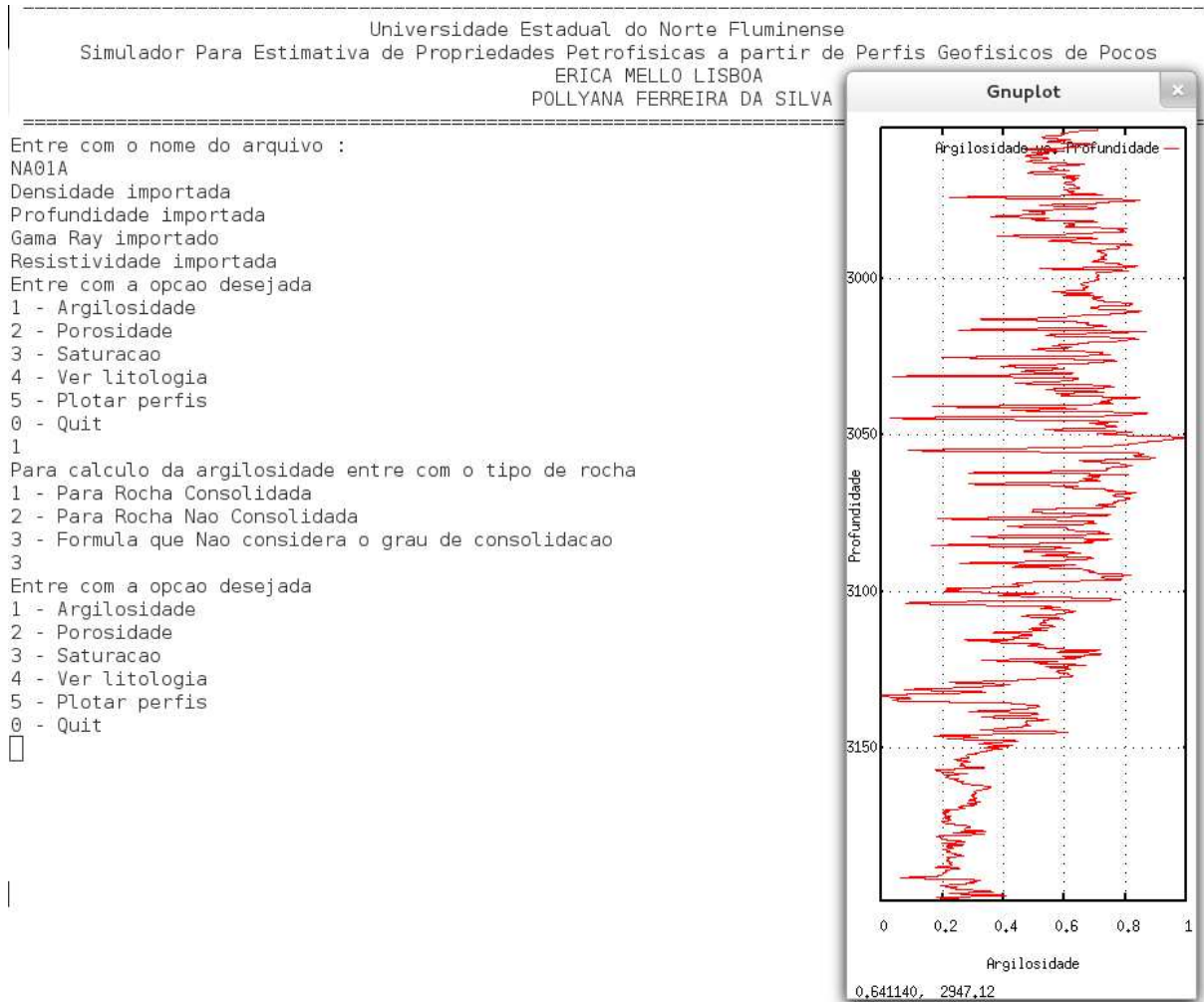


Figura 7.5: Plote da Argilosidade – não considera o grau de consolidação

## 7.4 Teste 3: Cálculo da Porosidade

O software calcula a porosidade, ao longo da profundidade, e mostra o resultado em forma de gráfico, Figura 7.6.

```
[lenep@localhost cpp]$ ./a.out
```

```
=====
Universidade Estadual do Norte Fluminense
Simulador Para Estimativa de Propriedades Petrofísicas a partir de Perfis Geofísicos de Pocos
ERICA MELLO LISBOA
POLLYANA FERREIRA DA SILVA
=====

Entre com o nome do arquivo :
NA01A
Densidade importada
Profundidade importada
Gama Ray importado
Resistividade importada
Entre com a opção desejada
1 - Argilosidade
2 - Porosidade
3 - Saturacao
4 - Ver litologia
5 - Plotar perfis
0 - Quit
2

Entre com a profundidade inicial da area desejada : 2990

Entre com o fim da profundidade da area desejada : 3011
A Porosidade media da area desejada e: 0.853419
```

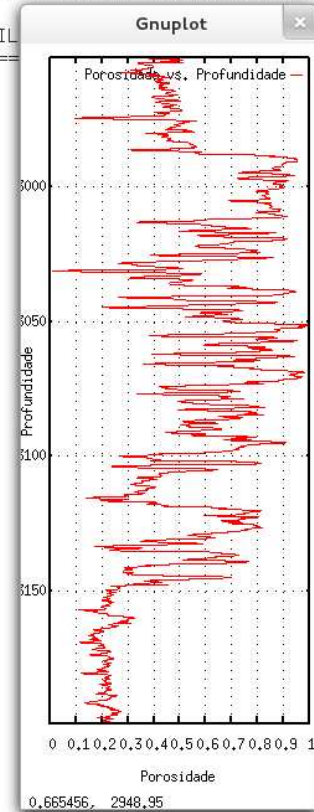


Figura 7.6: Porosidade

## 7.5 Teste 4: Cálculo da Saturação

O software calcula a saturação de hidrocarboneto, ao longo da profundidade, e mostra o resultado em forma de gráfico, Figura 7.7.

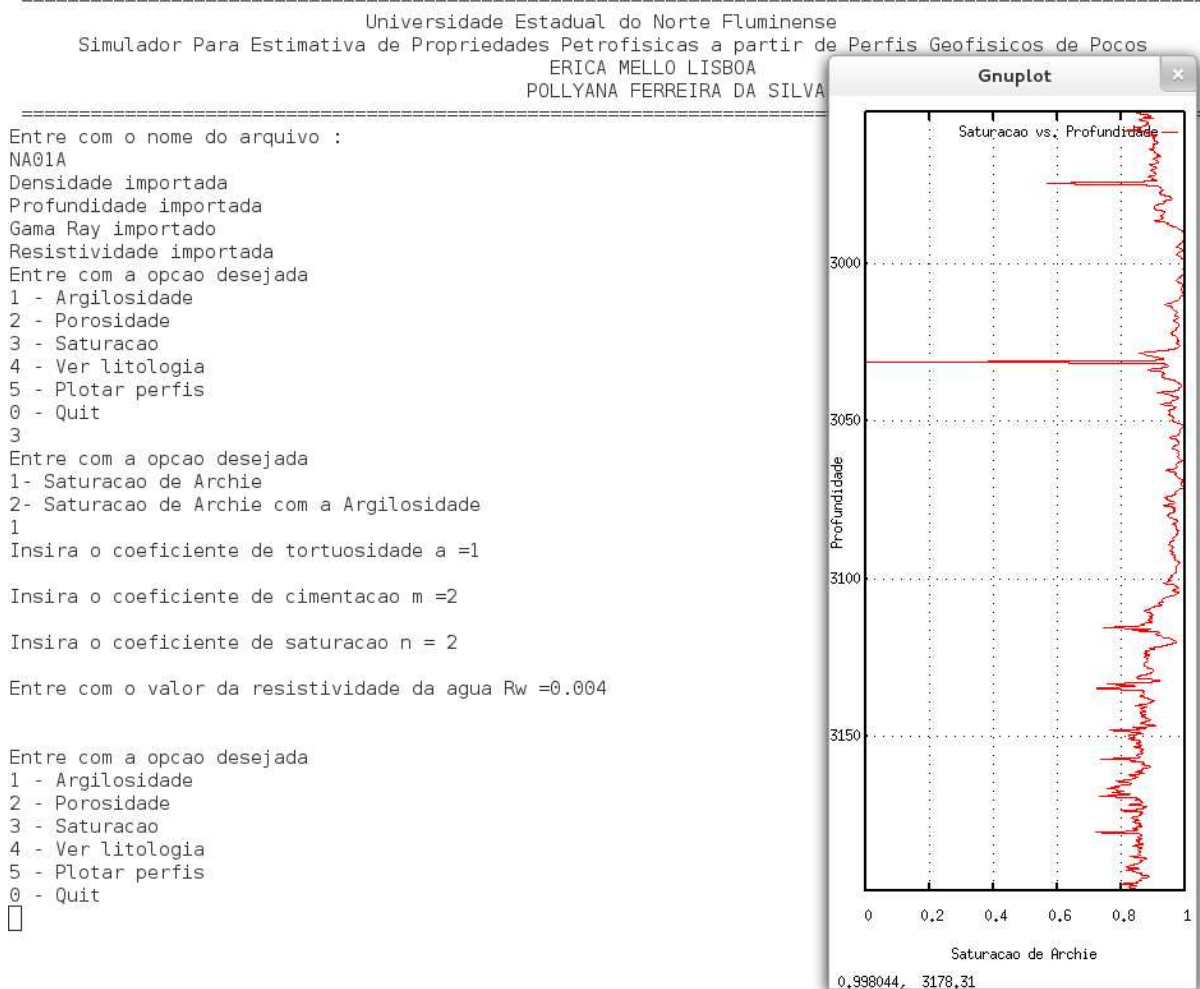


Figura 7.7: Saturação

## 7.6 Teste 5: Ver Litologia

O software utiliza o perfil de Gama Ray para definir e plotar a região de arenito e folhelho, Figura 7.8 na qual o vermelho indica a região referente ao arenito e o verde indica o folhelho.

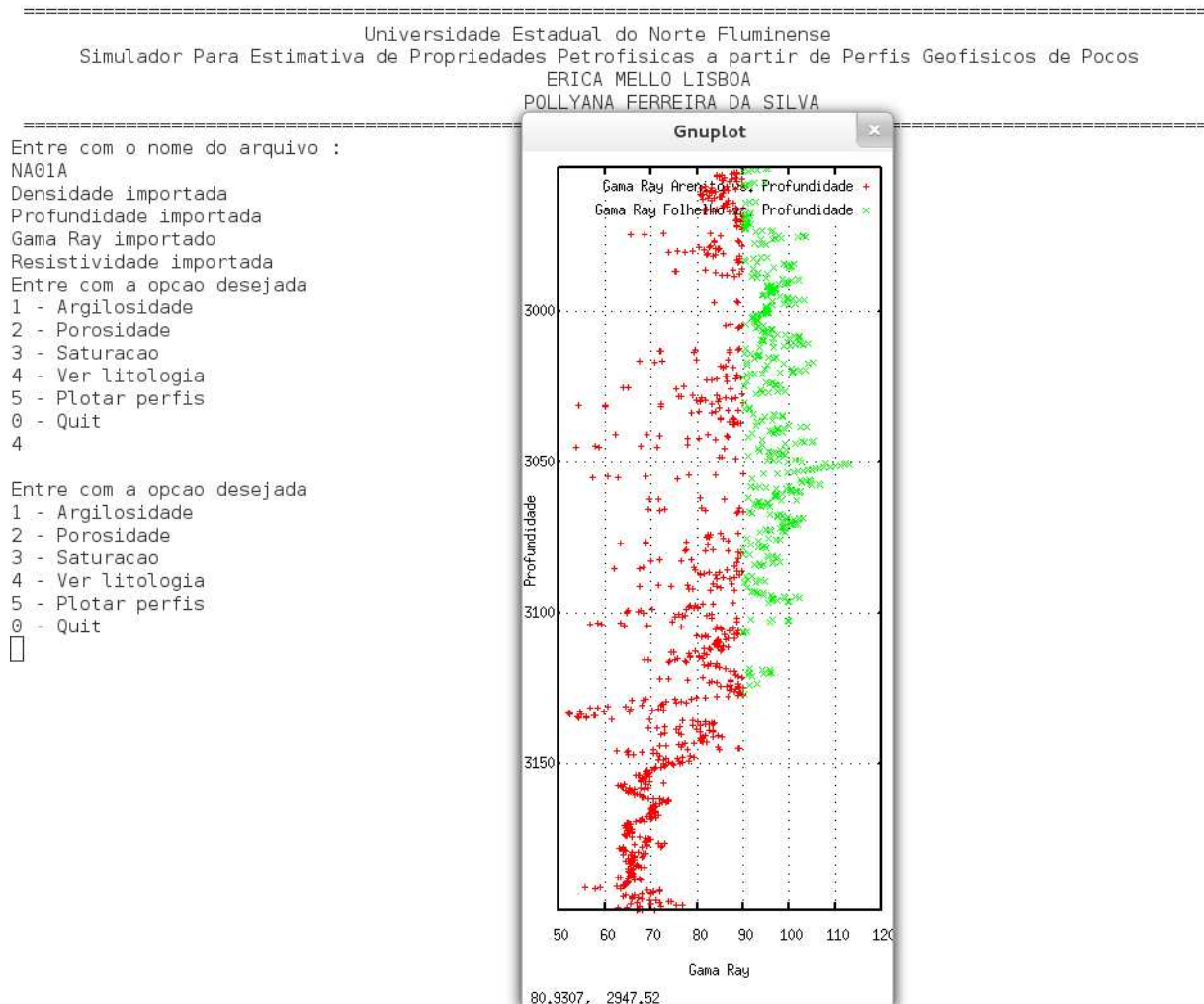


Figura 7.8: Litologia

# Capítulo 8

## Documentação

A presente documentação refere-se ao uso do simulador SEPP. Esta documentação tem o formato de uma apostila que explica passo a passo como usar o software.

### 8.1 Documentação do usuário

Descreve-se aqui o manual do usuário, um guia que explica, passo a passo a forma de instalação e uso do software desenvolvido.

#### 8.1.1 Como rodar o software

Abra o terminal, vá para o diretório onde está o código, compile o programa e, depois, execute. Logo após executar, siga os seguintes passos:

1. Entre com o nome do arquivo que contém os dados da perfilagem (com extensão .dat);
2. Entre com a opção desejada:
  - (a) Argilosidade – opção 1;
  - (b) Porosidade – opção 2;
  - (c) Saturação – opção 3;
  - (d) Ver litologia – opção 4;
  - (e) Plotar perfis – opção 5;
  - (f) Sair do programa – opção 0;
3. Caso entre com a opção 1, escolha a opção referente ao tipo de rocha:
  - (a) Para rocha consolidada – opção 1;
  - (b) Para rocha não consolidada – opção 2;

- (c) Fórmula que não considera o grau de consolidação – opção 3;
- 4. Após isso o resultado será mostrado em forma de gráfico;
- 5. Caso entre com a opção 2, o resultado da porosidade será plotado em forma de gráfico;
- 6. Caso entre com a opção 3, escolha a opção referente ao tipo de fórmula:
  - (a) Saturação de Archie – opção 1;
    - i. informe o coeficiente de tortuosidade ( $a$ );
    - ii. informe o coeficiente de cimentação ( $m$ );
    - iii. informe o coeficiente de saturação ( $n$ );
    - iv. informe a resistividade da água ( $rw$ );
  - (b) Saturação de Archie com argilosidade – opção 2;
    - i. informar o tipo da rocha para calcular a argilosidade;
    - ii. além de informar os coeficientes  $a$ ,  $m$ ,  $n$  e  $rw$ , é necessário informar, também, a resistividade média da argila ( $rsh$ );
- 7. Caso entre com a opção 4 o gráfico será mostrado indicando, com cores diferentes, a região de arenito e folhelho;
- 8. Caso entre com a opção 5 aparecerão três graficos referentes ao perfis de Radiação Gama, Resistividade e Densidade;
- 9. Caso entre com a opção 0 o software será encerrado.

## 8.2 Documentação para desenvolvedor

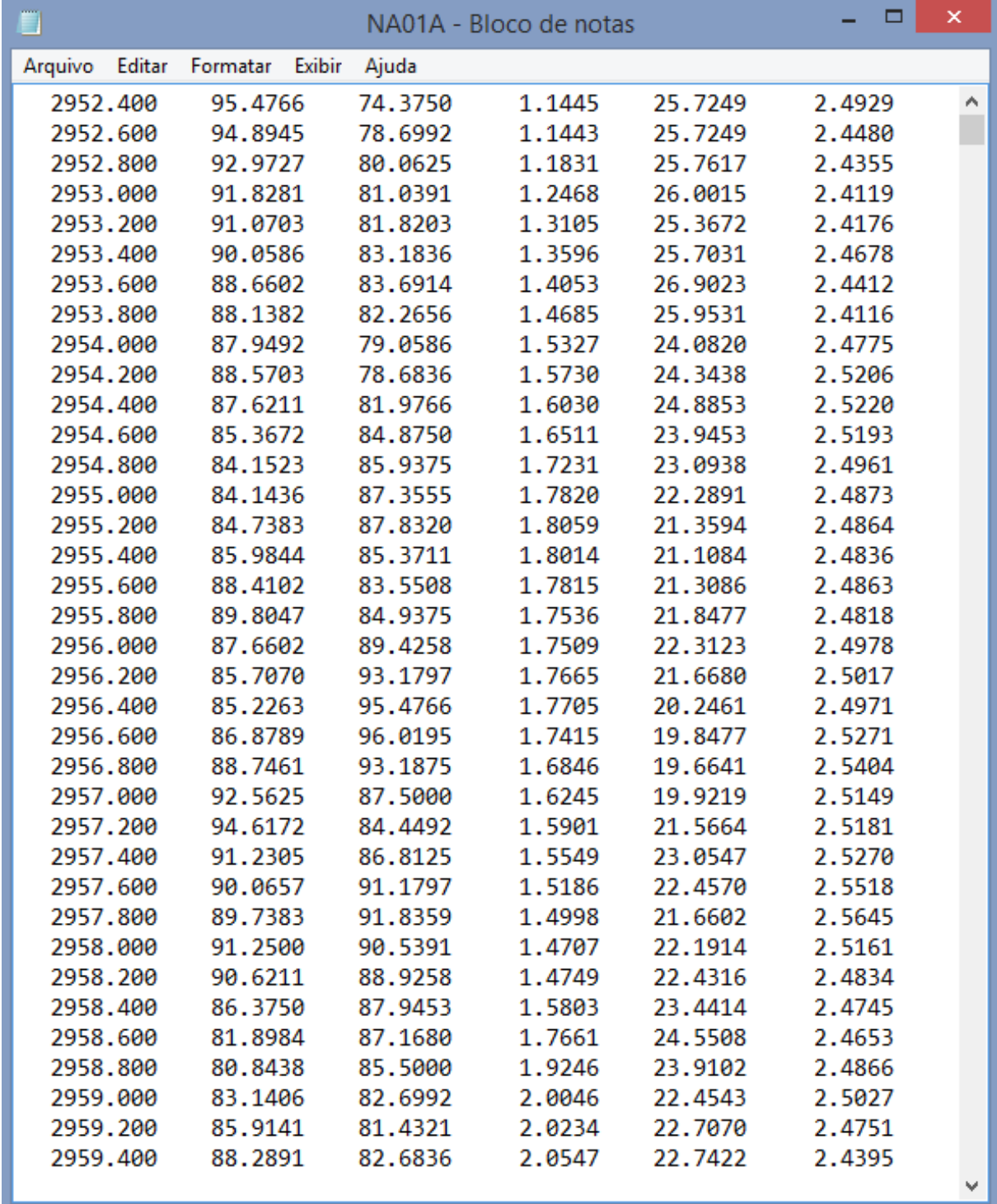
Apresenta-se nesta seção a documentação para o desenvolvedor, isto é, informações para usuários que queiram modificar, aperfeiçoar ou ampliar este software.

### 8.2.1 Dependências

Para compilar o software é necessário atender as seguintes dependências:

- No sistema operacional GNU/Linux:
  - Instalar o compilador `g++` da GNU disponível em <http://gcc.gnu.org>.
  - Para instalar no GNU/Linux use o comando: `yum install gcc`.
- No sistema operacional Windows:

- Instalar um compilador apropriado.
  - Recomenda-se o Dev C++ disponível em <http://dev-c.softonic.com.br/>.
- O software gnuplot, deve estar instalado.
  - Gnuplot está disponível no endereço <http://www.gnuplot.info/>.
  - É possível que haja necessidade de setar o caminho para execução do gnuplot.
- O programa depende da existência de um arquivo de dados (formato .dat), Figura 8.1, para preencher os vetores dos perfis.



Arquivo	Editar	Formatar	Exibir	Ajuda		
2952.400	95.4766	74.3750	1.1445	25.7249	2.4929	
2952.600	94.8945	78.6992	1.1443	25.7249	2.4480	
2952.800	92.9727	80.0625	1.1831	25.7617	2.4355	
2953.000	91.8281	81.0391	1.2468	26.0015	2.4119	
2953.200	91.0703	81.8203	1.3105	25.3672	2.4176	
2953.400	90.0586	83.1836	1.3596	25.7031	2.4678	
2953.600	88.6602	83.6914	1.4053	26.9023	2.4412	
2953.800	88.1382	82.2656	1.4685	25.9531	2.4116	
2954.000	87.9492	79.0586	1.5327	24.0820	2.4775	
2954.200	88.5703	78.6836	1.5730	24.3438	2.5206	
2954.400	87.6211	81.9766	1.6030	24.8853	2.5220	
2954.600	85.3672	84.8750	1.6511	23.9453	2.5193	
2954.800	84.1523	85.9375	1.7231	23.0938	2.4961	
2955.000	84.1436	87.3555	1.7820	22.2891	2.4873	
2955.200	84.7383	87.8320	1.8059	21.3594	2.4864	
2955.400	85.9844	85.3711	1.8014	21.1084	2.4836	
2955.600	88.4102	83.5508	1.7815	21.3086	2.4863	
2955.800	89.8047	84.9375	1.7536	21.8477	2.4818	
2956.000	87.6602	89.4258	1.7509	22.3123	2.4978	
2956.200	85.7070	93.1797	1.7665	21.6680	2.5017	
2956.400	85.2263	95.4766	1.7705	20.2461	2.4971	
2956.600	86.8789	96.0195	1.7415	19.8477	2.5271	
2956.800	88.7461	93.1875	1.6846	19.6641	2.5404	
2957.000	92.5625	87.5000	1.6245	19.9219	2.5149	
2957.200	94.6172	84.4492	1.5901	21.5664	2.5181	
2957.400	91.2305	86.8125	1.5549	23.0547	2.5270	
2957.600	90.0657	91.1797	1.5186	22.4570	2.5518	
2957.800	89.7383	91.8359	1.4998	21.6602	2.5645	
2958.000	91.2500	90.5391	1.4707	22.1914	2.5161	
2958.200	90.6211	88.9258	1.4749	22.4316	2.4834	
2958.400	86.3750	87.9453	1.5803	23.4414	2.4745	
2958.600	81.8984	87.1680	1.7661	24.5508	2.4653	
2958.800	80.8438	85.5000	1.9246	23.9102	2.4866	
2959.000	83.1406	82.6992	2.0046	22.4543	2.5027	
2959.200	85.9141	81.4321	2.0234	22.7070	2.4751	
2959.400	88.2891	82.6836	2.0547	22.7422	2.4395	

Figura 8.1: Exemplo do arquivo de dados

## 8.2.2 Documentação usando doxygen

A documentação do código do simulador desenvolvido foi realizada usando padrão JAVADOC, através do software doxygen que lê o arquivos com os códigos (\*.h e \*.cpp) e gera uma documentação útil e de fácil navegação no formato html. Segue exemplo da documentação gerada, Figura 8.2 , e de alguns diagramas gerados pelo software , Figuras 8.3 e 8.4.



Atividades Firefox Qui 15:02

My Project: Data Structures - Mozilla Firefox

Arquivo Editar Exibir Histórico Favoritos Ferramentas Ajuda

My Project: Data Structures

file:///home/lenep/Documents/cpp/html/annotated.html

## My Project

Main Page	Data Structures	Files
Data Structures	Data Structure Index	Class Hierarchy
		Data Fields

### Data Structures

Here are the data structures with brief descriptions:

<b>N</b> std	
<b>C</b> CArgilosidade	Esta classe guarda o valor da argilosidade calculada de acordo com a formula desejada, referente aos três métodos de cálculo
<b>C</b> CDensidade	
<b>C</b> CGamaRay	
<b>C</b> CLitologia	Esta classe plota o grafico de GamaRay especificando onde está cada litologia
<b>C</b> CPerfil	Esta classe serve para armazenar os valores das profundidades e dos perfis de GamaRay, Resistividade e Densidade
<b>C</b> CPorosidade	
<b>C</b> CProfundidade	
<b>C</b> CResistividade	
<b>C</b> CSaturacaoArchieModificada	
<b>C</b> CSaturacaoArchieNormal	
<b>C</b> CSimulador	
<b>C</b> Gnuplot	Classe de interface para acesso ao programa gnuplot
<b>C</b> GnuplotException	Erros em tempo de execucao

Figura 8.2: Documentação do projeto gerada pelo doxygen

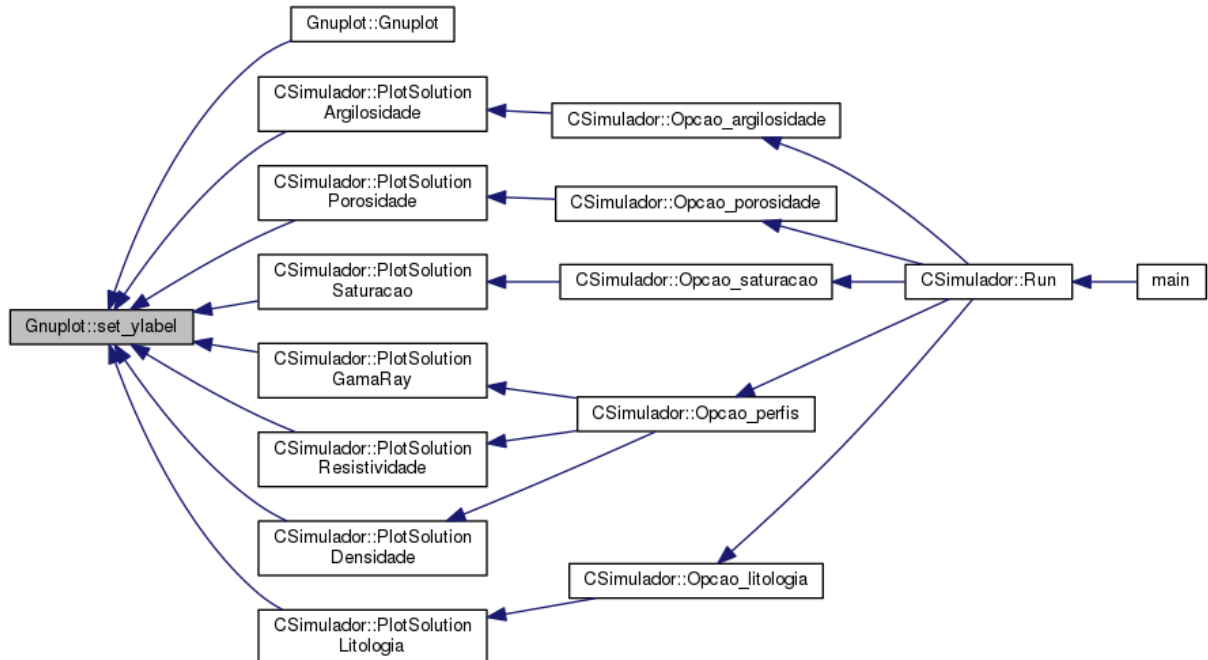


Figura 8.3: Diagrama gerado pelo doxygen: Relacionamento entre as classes CSimulador e CGnuplot

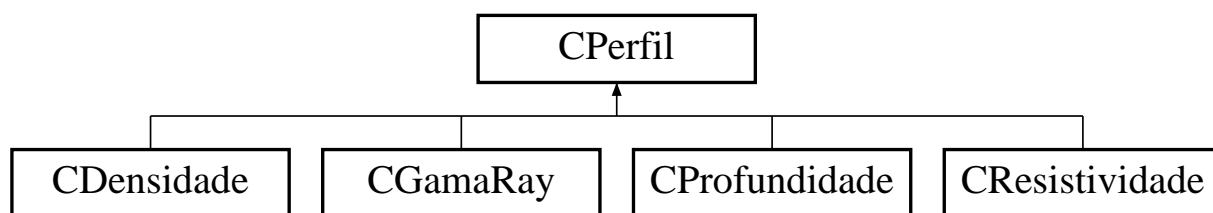


Figura 8.4: Diagrama gerado pelo doxygen: Herança entre as classes

## Capítulo 9

### Referências Bibliográficas

[1]NERY, G.G. PERFILAGEM GEOFÍSICA. Salvador. 2004. Disponível em: <http://www.gerald...>. Acesso em 22 de Julho de 2013.

[2]RIDER, M. The Geological Interpretation of Well Logs. 2 ed. Scotland. Rider-French Consulting Ltd, Sutherland. 2002. p280.

[3]STEVANATO, A.C.R.S. Análise Petrofísica de Reservatórios. Campinas. 2011. Graduação em Geologia. Universidade Estadual de Campinas.

[4]THOMAS, J.E. Fundamentos de Engenharia de Petróleo. Rio de Janeiro. Editora Interciência. 2001. p.271.

[5]UMBRELLO, Editor de diagramas.

[6]EDITOR DE DIAGRAMAS DIA.

[7]KATE, Editor de texto.

[8]DOXYGEN, Gerador de documentação.

[9]GNU PLOT, Gerador de gráficos.

# Índice Remissivo

Análise de domínio, 6  
Análise orientada a objeto, 14  
AOO, 14  
Assuntos, 11  
assuntos, 12  
  
Casos de uso, 4  
Cenário, 4  
colaboração, 23  
comunicação, 23  
Concepção, 3  
Controle, 27  
  
Diagrama de colaboração, 23  
Diagrama de componentes, 28  
Diagrama de execução, 29  
Diagrama de máquina de estado, 24  
Diagrama de pacotes, 12  
Diagrama de pacotes (assuntos), 12  
Diagrama de sequência, 21  
  
Efeitos do projeto nas heranças, 28  
Efeitos do projeto nos métodos, 28  
Elaboração, 6  
especificação, 3  
Especificações, 3  
estado, 24  
Eventos, 21  
  
Heranças, 28  
heranças, 28  
  
Identificação de pacotes, 11  
Implementação, 31  
métodos, 28  
módulos, 12  
Mensagens, 21  
modelo, 27  
  
Plataformas, 27  
POO, 27  
Projeto do sistema, 26  
Projeto orientado a objeto, 27  
Protocolos, 26  
  
Recursos, 26