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

BANCO DE DADOS DE FLUIDOS DE PERFURAÇÃO

TRABALHO DA DISCIPLINA PROGRAMAÇÃO PRÁTICA

ANNA MARA CORRÊA DE OLIVEIRA
RAIZA GOMES DE SOUZA

MACAÉ - RJ
Março - 2016

Sumário

Capítulo 1

Introdução

No presente trabalho desenvolve-se um projeto de engenharia em linguagem orientada a objeto que tem como principal objetivo o gerenciamento de informações de fluidos de perfuração desenvolvidos no Laboratório de Fluidos do LENEP-Laboratório de Engenharia e Exploração de Petróleo a partir da construção de um banco de dados. Dessa forma a principal finalidade do programa é desenvolver um banco de dados onde será armazenado infromações relacionadas às propriedades físicas e químicas dos fluidos, onde o usuárrio terá acesso a essas informações assim como poderá incluir novas informações a respeito de novos fluidos desencolvidos.

1.1 Escopo do Problema

Banco de dados são gerenciadores de grandes grupos de informações. Esse gerenciamento consiste em definir a estrutura para o armazenamento de informações e o fornecimento de mecanismos para manipulá-las. Esse gerenciamneto é executado pelos Sistenas de Gerenciamento de Banco de Dados (SGBD), softwares que possuem recursos para efetuar esse tipo de manipulação.

Os fluidos de perfuração são misturas complexas,sua reprodução requer o armazenamento de dados como tipo de base,teor de base,propriedades físicas e químicas,propriedades reológicas e aditivos no computador.

1.2 Objetivos

Os objetivos deste trabalho são:

- Objetivo geral:
  - Desenvolver um banco de dados contendo informações a respeito de fluidos de perfuração desenvolvidos no Laboratório de Fluidos do LENEP-Laboratório de Engenharia e Exploração de Petróleo.

- Objetivos específicos:
  - Possibilitar inclusão de informações de novos fluidos de perfuração;
  - Possibilitar acesso às informações do banco de dados pelo usuário;
  - Possibilitar realização de pesquisa de informações no banco de dados pelo usuário.

Capítulo 2

Especificação

Apresenta-se neste capítulo a especificação do sistema a ser modelado e desenvolvido.

2.1 Especificação do programa

Deseja-se desenvolver um projeto de engenharia com interface em modo texto que permita ao usuário ter acesso a um banco de dados de fluidos de perfuração desenvolvidos em laboratório com diferentes propriedades físicas e químicas. O usuário poderá incluir e excluir novos fluidos de perfuração ao banco de dados, listar e exibir fluidos de perfuração, realizar pesquisa de fluidos com propriedades especificas determinadas pelo usuário e gerar relatórios.

Para isso o usuário primeiramente terá acesso a um menu com opções : banco de dados, abrir diretório de trabalho, abrir diretório de Dados e configurações. Ao selecionar a opção Banco de Dados, o usuário terá acesso ao menu com as opções de listar, pesquisar, inserir,excluir,exportar e informações de um determinado fluido de perfuração.

Por ser um software científico o mesmo deve ser passível de modificações, logo dever ter o seu código aberto. O banco de dados será desenvolvido em linguagem de programação orientada a objeto C++ e será utilizado em laboratórios do LENEP / CCT/ UENF - Macaé - RJ para fins estudantis, laboratoriais e de pesquisa.

2.2 Casos de uso do Programa

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

Apresenta-se a seguir alguns diagramas de caso de uso. O objetivo é gerar uma percepção básica das interações do usuário com o banco de dados.

2.3 Diagrama de caso de uso geral

Abaixo encontra-se o diagrama de caso de uso geral que descreve a interação do usuário com o banco de dados.

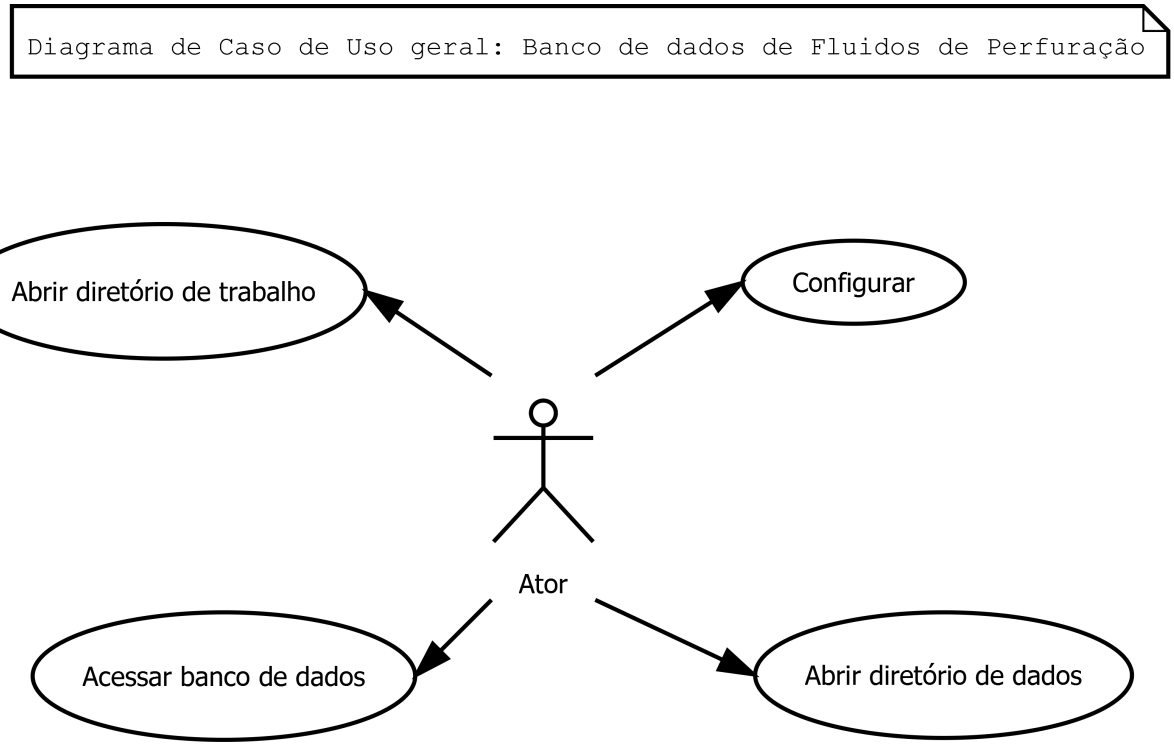

Figura 2.1: Diagrama de caso de uso geral

2.4 Diagramas de caso de uso específicos

Abaixo encontra-se o diagrama de caso de uso especifico “Acessar Banco de Dados”

Abaixo encontra-se o diagrama de caso de uso especifico “Inserir Fluido de Perfuração”

Abaixo encontra-se o diagrama de caso de uso especifico “Pesquisar Fuidos com propriedades específicas”

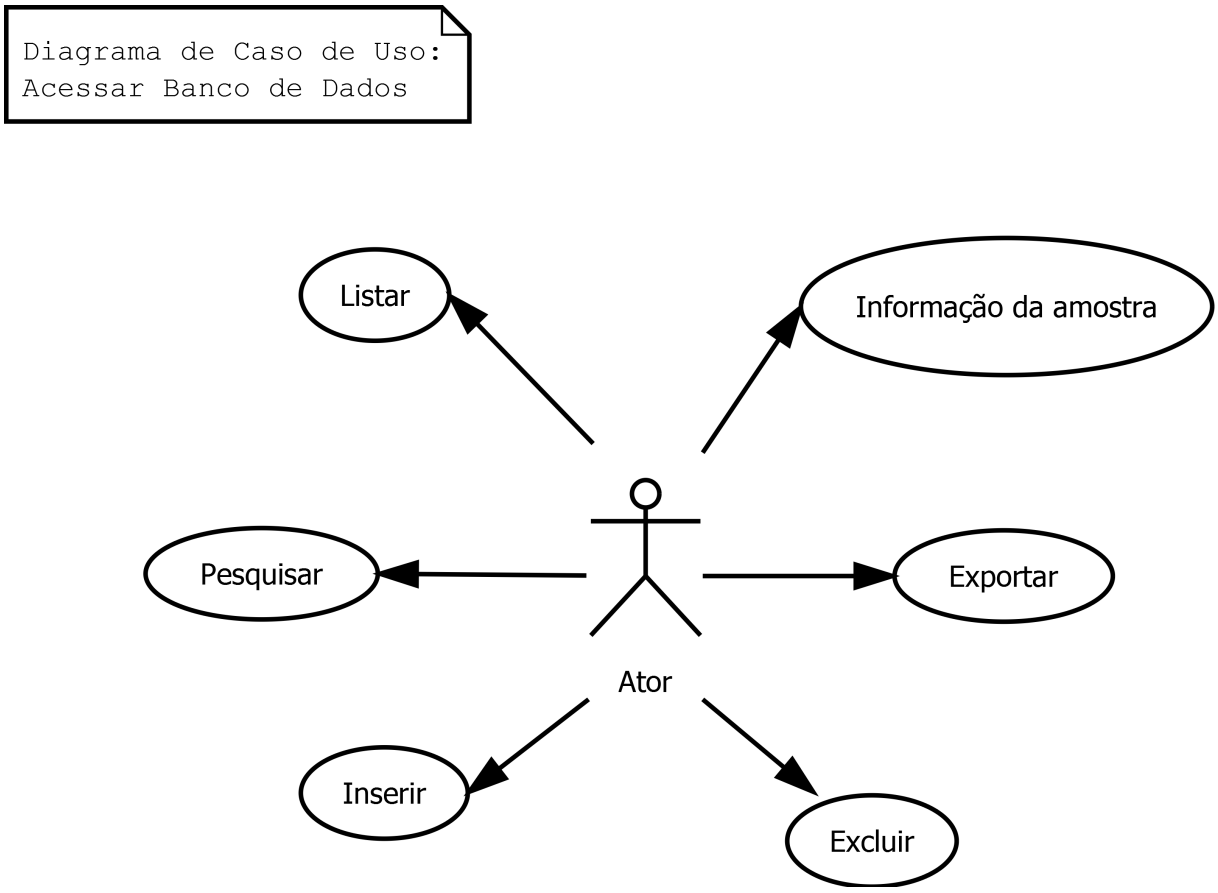

Figura 2.2: Diagrama de Caso de uso específico: Acessar banco de dados

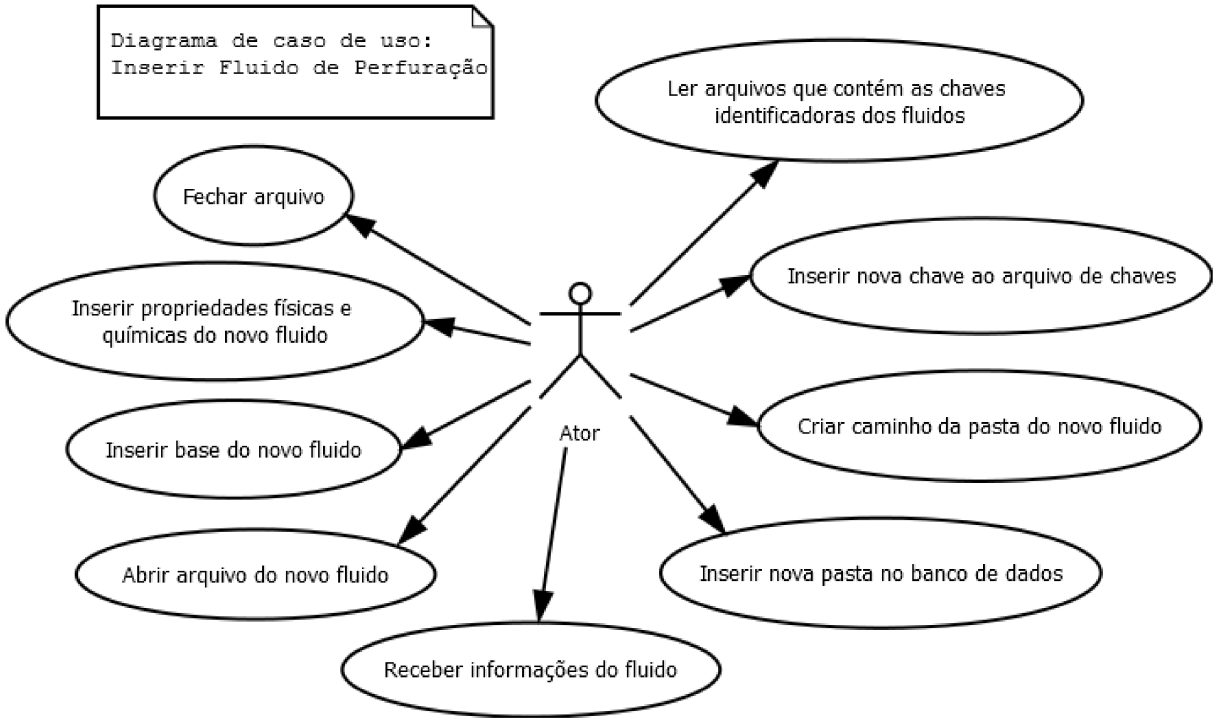

Figura 2.3: Diagrama de caso de uso específico: Inserir Fluido de Perfuração

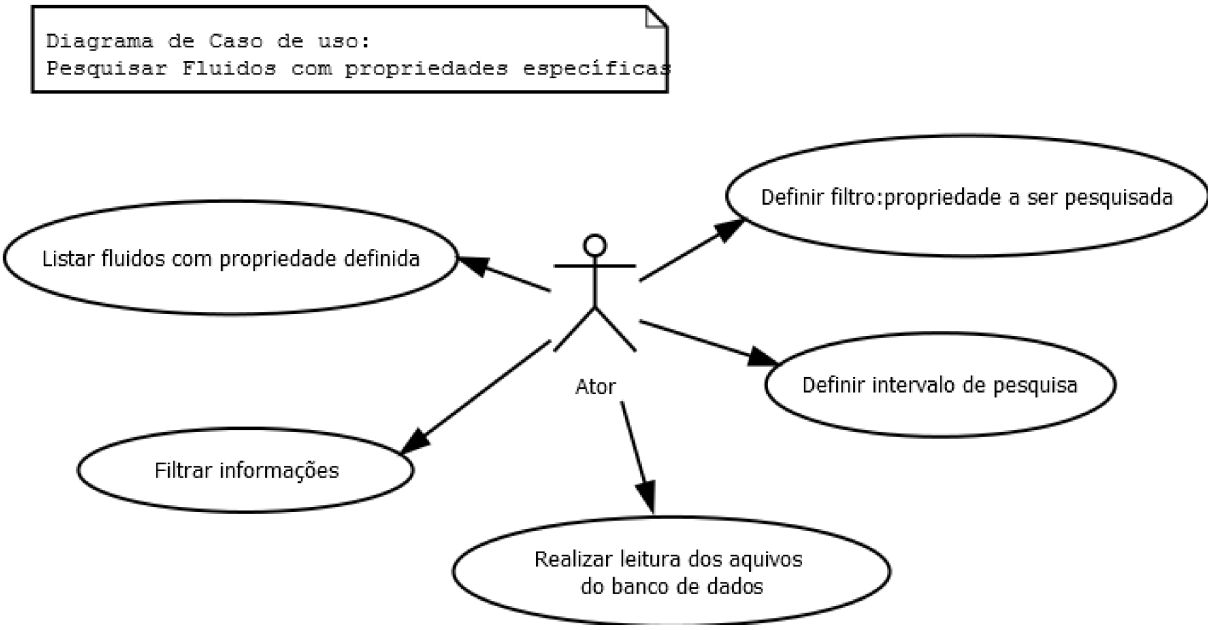

Figura 2.4: Diagrama de caso de uso específico: Pesquisar Fuidos com propriedades específicas

Capítulo 3

Elaboração

No processo de elaboração, é realizado um estudo de abrangência do software em desenvolvimento, ajustando os requisitos iniciais do sistema que foram definidos na etapa de especificação. Tem-se como objetivo possibilitar o desenvolvimento de um sistema útil, que atenda as necessidades do usuário e permita futuras extensões do programa.

3.1 Análise de Domínio

A análise de domínio é uma parte da elaboração; seu objetivo é entender o domínio, a abrangência do sistema a ser desenvolvido. Envolve itens como estimar o reuso do software utilizando-se da criação de bibliotecas genéricas. Neste ponto, o analista pensa no sistema de uma forma mais genérica, identificando conceitos fundamentais que podem ser reaproveitados em outros sistemas.

Considerando que o software tem como objetivo principal a criação de um banco de dados de fluidos de perfuração desenvolvidos no Laboratório de Engenharia e Exploração de Petróleo-LENEP, base óleo ou base água, que contém as características de acordo com suas composições quimicas e tipos de aditivos, a análise de domínio envolverá:

- O uso dos livros de engenharia de reservatório, de programação orientada a objeto, de Fluidos de Perfuração para desenvolvimento do software.
- Consultas a livros, sites, artigos, monografias, dados de trabalhos de iniciação científica dos alunos do LENEP.
- Como o programa tem interface bastante simplificada e realiza rotinas que dependem dos dados de entrada e dados armazenados, a necessidade de uma grande Memória Ram deve ser analisada.

3.1.1 Banco de dados

Um SGBD tem que ter algumas particularidades e deve facilitar o processo de definir (especificar tipos de dados a serem armazenados), construir (armazenar dados que possam

ser manipulados por um SGBD) e manipular (inserir, atualizar e remover base dados de diversas aplicações). As principais características de um SGBD são:

- Controle de redundância: pode-se construir regras para que o gerenciamento seja mais eficaz evitando assim a redundância dos dados e economizando espaço em disco. Por exemplo, um aluno só pode ser cadastrado uma única vez em cada curso; cada disciplina só pode ser cadastrada uma vez em um único curso; ou ainda, cada aluno só pode se inscrever uma vez em cada matéria.

- Restrição a acesso não autorizado: Em um banco de dados com vários usuários, cada um tem acesso no que lhe é permitido. Com um SGBD é possível restringir os acessos de cada usuário ou grupo de usuários, permitido assim acessos autorizados para cada usuário.

- Garantia de armazenamento persistente: Com um SGBD é possível armazenar dados de uma forma organizada.

- Garantia de armazenamento de estruturas para o processamento eficiente de consultas: Uma outra característica de um SGBD é que além de armazenar dados ele deve prover mecanismo que facilitem a busca, a inserção ou atualização da base de dados.

- Compartilhamento de dados: SGBDs multiusuários devem fornecer controle de concorrência para assegurar que atualizações simultâneas resultem em modificações corretas.

- Fornecimento de múltiplas interfaces: Devido aos vários tipos de usuários, com variados níveis de conhecimento técnico, um SGBD deve fornecer uma variedade de interfaces para atendê-los. Os tipos de interfaces incluem linguagens de consulta para usuários ocasionais, interfaces de linguagem de programação para programadores de aplicações, formulários e interfaces dirigidas por menus para usuários comuns.

- Representação de relacionamento complexo entre dados: Uma base de dados pode possuir uma variedade de dados que estão inter-relacionados de muitas maneiras. Um SGBD deve ter a capacidade de representar uma variedade de relacionamentos complexos entre dados, bem como recuperar e modificar dados relacionados de maneira fácil e eficiente.

- Backup e restauração: Garantir backup e restauração de dados é tarefa essencial para qualquer SGBD. Mesmo que as falhas sejam ocasionadas por falhas de software ou hardware ele deve garantir a integridade dos dados.

- Restrições de integridade: Num SGBD é possível impor restrições, por exemplo, em uma tabela ALUNO que contém atributos: Nome, CPF, Endereço, Tel, o atributo Nome possa ter no máximo 50 caracteres, e que CPF pode ter 11 caracteres e que

Tel pode receber 11 inteiros, ou ainda, a tabela Turma deve ser preenchida com dados da tabela Professor e da tabela Aluno etc.

Banco de dados orientado a objeto

Visando acompanhar a tendência da época e também possibilitar resolver as limitações que os bancos de dados possuíam, foi proposto um novo sistema de banco de dados orientados a objeto (BDOO).

De uma forma bem simples pode-se dizer que o BDOO é nada mais que a junção entre conceitos de OO com conceiros de SGBD, ou seja, ele é todo baseado nos paradigmas da OO unido aos objetivos básicos dos SGBD.

3.1.2 Fluidos de Perfuração

Os fluidos de perfuração são vistos de diferentes maneiras por diferentes autores. O instituto Americano de Petróleo (API) considera fluido de perfuração qualquer fluido circulante capaz de tornar a operação de perfuração viável. Autores como Thomas et al. (2001) consideram os fluidos de perfuração como misturas complexas de sólidos, líquidos, produtos químicos e, por vezes, até de gases.

Um fluido de perfuração além de ter de realizar suas funções primordiais, que são a suspensão, o controle de pressão, a estabilização das formações, apresentar poder de flutuação e de resfriamento da broca (Duarte, 2004), também deve apresentar características adequadas para que possam ser utilizados nas diversas formações. Sendo assim, um fluido de perfuração deve ser estável quimicamente, facilitar a separação dos cascalhos na superfície, ser inerte (não reagir) com as rochas produtoras, ser capaz de aceitar tratamento físico e/ou químico, ser passível de bombeamento, deve apresentar baixo grau de corrosão e abrasão (esfoliamento) em relação à coluna de perfuração e a outros equipamentos da coluna de perfuração, e ainda não ser agressivo ao meio ambiente (Thomas et al., 2001). Além das funções cruciais de um fluido de perfuração, eles apresentam funções e características secundárias, tais como: resfriar e limpar pequenas impurezas, apresentar baixo custo de operação, facilitar as interpretações geológicas do material retirado do poço, dentre outras.

Os principais componentes dos fluidos de perfuração são a base (ar, água e óleo) e os aditivos químicos. De acordo com a base predominante utilizada em sua preparação os fluidos são classificados em base ar, água e/ou base óleo. A maior parte das operações de perfuração no mundo usam lamas base água, contra apenas cerca de 5 a 10% que utilizam fluidos base óleo e uma porção ainda menor de poços que são perfurados com fluidos base ar (Caenn, 1995).

Segundo Caenn et al (1995), os aditivos mais comuns utilizados nos fluidos de perfuração são os polímeros, surfactantes, sais e bentonitas. Pesquisas são continuamente executadas para aumentar a performance dos fluidos de perfuração e aditivos individuais

são frequentemente desenvolvidos para alterar uma ou mais propriedades da lama, para que assim possa ser formulado o fluido que atenda às necessidades exigidas para cada aplicação.Os principais aditivos utilizados são adensantes, sais, redutores de filtrado, biopolímeros, viscosificantes, dispersantes, defloculantes, emulsionantes, biocidas, salmoura, lubrificantes, inibidores de corrosão e controladores de pH (Bleier, 1992; Economides, 1998; Veiga, 1998; Barbosa, 2005; Candler & Friedheim, 2006).

- Adensantes - substâncias usadas para aumentar a densidade com o intuito de controlar a pressão hidrostática do poço para assim prevenir a ocorrência de blow outs ou o dano à formação. Qualquer substância mais densa que o fluido e que não provoque nenhum efeito adverso nas demais propriedades podem ser usadas. Além do custo, deve-se levar em consideração o volume ocupado pelo aditivo. Os materiais mais usados com essa finalidade são Dolomita, Calcita, Hematita, Galena e especialmente a Barita (BaSO4) (Darley& Gray, 1988).

- Viscosificantes - agentes utilizados para conferir viscosidade alta em baixo cisalhamento e viscosidade baixa em alto cisalhamento. Neste caso, o aditivo mais utilizado é a bentonita, a qual incha em contato com a água reduzindo a fricção entre a coluna de perfuração e as paredes do poço, também podem ser utilizados polímeros sintéticos como o policatiônico;

- Biopolímeros- usados no controle reológico e para melhorar o processo de carregamento de cascalhos durante a perfuração, geralmente atuam tornando o fluido mais viscoso. Os polímeros mais utilizados na indústria são: CMC (carboximetilcelulose), HEC (hidroxietilcelulose) e o CMS (carboximetilamido);

- Sais - atuam como inibidores das formações ativas de maneira a reduzir o escoamento hidráulico para a formação, além de estimular o escoamento de água da formação argilosa para o fluido de perfuração. Os sais mais utilizados em fluidos de perfuração base água são: cloreto de sódio (NaCl), cloreto de potássio (KCl) e cloreto de cálcio ($CaCl2$), entretanto também podem ser utilizados polímeros naturais e sintéticos;

- Salmouras - utilizada como a fase aquosa, tem a função de balancear as interações dos fluidos de perfuração com argilas ou sais solúveis das formações. Normalmente utiliza-se NaCl ou KCl como salmouras para fluidos à base de água e CaCl2 para fluidos sintéticos ou à base de óleo;

- Redutores de filtrado – adicionados com o objetivo de controlar a perda de fluido,atuam minimizando a penetração do fluido de perfuração na formação e promovendo a melhoria do reboco formado nas paredes do poço. Geralmente utiliza-se amido, bentonita, lignita ou polímeros para alcançar tal finalidade;

- Dispersantes - Os aditivos do tipo lignosulfonatos e lignito possuem a função de dispersarem os sólidos presentes nos fluidos de perfuração e, por isso, são conhecidos como dispersantes;
- Defloculantes - com o intuito de prevenir a floculação dos sólidos ativos nos fluidos de perfuração utilizam-se principalmente poliacrilatos de cálcio, sódio e potássio;
- Emulsionantes - tais como os ácidos graxos e alquilados sulfonados, responsáveis por formar, manter e estabilizar emulsões óleo em água e água em óleo;
- Biocidas - aditivos como glutaraldeído, sais quaternários de amônio e tiocianato usados para controlar os processos fermentativos do fluido de perfuração devido à ação de microorganismos;
- Lubrificantes - aplicados para reduzir o atrito entre a coluna de perfuração e as paredes do poço usam-se, por exemplo, ésteres de ácidos graxos e polipropilenoglicol;
- Inibidores de corrosão - entram na formulação do fluido com o intuito de prevenir corrosão e descamação dos tubos e demais equipamentos de perfuração. Para este fim tem-se aminas e álcoois de cadeia longa;
- Controladores de pH – usam-se hidróxidos de sódio ou potássio, ácido acético, acetato e carbonato de sódio como aditivos com função principal de controlar o pH dos fluidos numa faixa preestabelecida, mas também como redutores de corrosão e estabilizadores de emulsões.

3.2 Identificação de pacotes

Em UML, um pacote é um mecanismo de agrupamento genérico que contém classes que fazem parte de um assunto e relacionam-se por um conceito comum. Em outras palavras, agrupam classes que se relacionam com maior frequência.

3.3 Diagrama de pacotes

Os pacotes principais serão as bibliotecas e o executável, além dos pacotes:

- Pacote Fluido: Contém as propriedades físicas e químicas associadas aos fluidos e as diferentes características associadas ao tipo de base , óleo ou água, e aos aditivos.
- Pacote Interface: Contém a estrutura necessária para definição da interface do programa.
- Pacote Banco de Dados: Contém as informações dos diferentes tipos de fluidos e todas suas propriedades relaciondas.

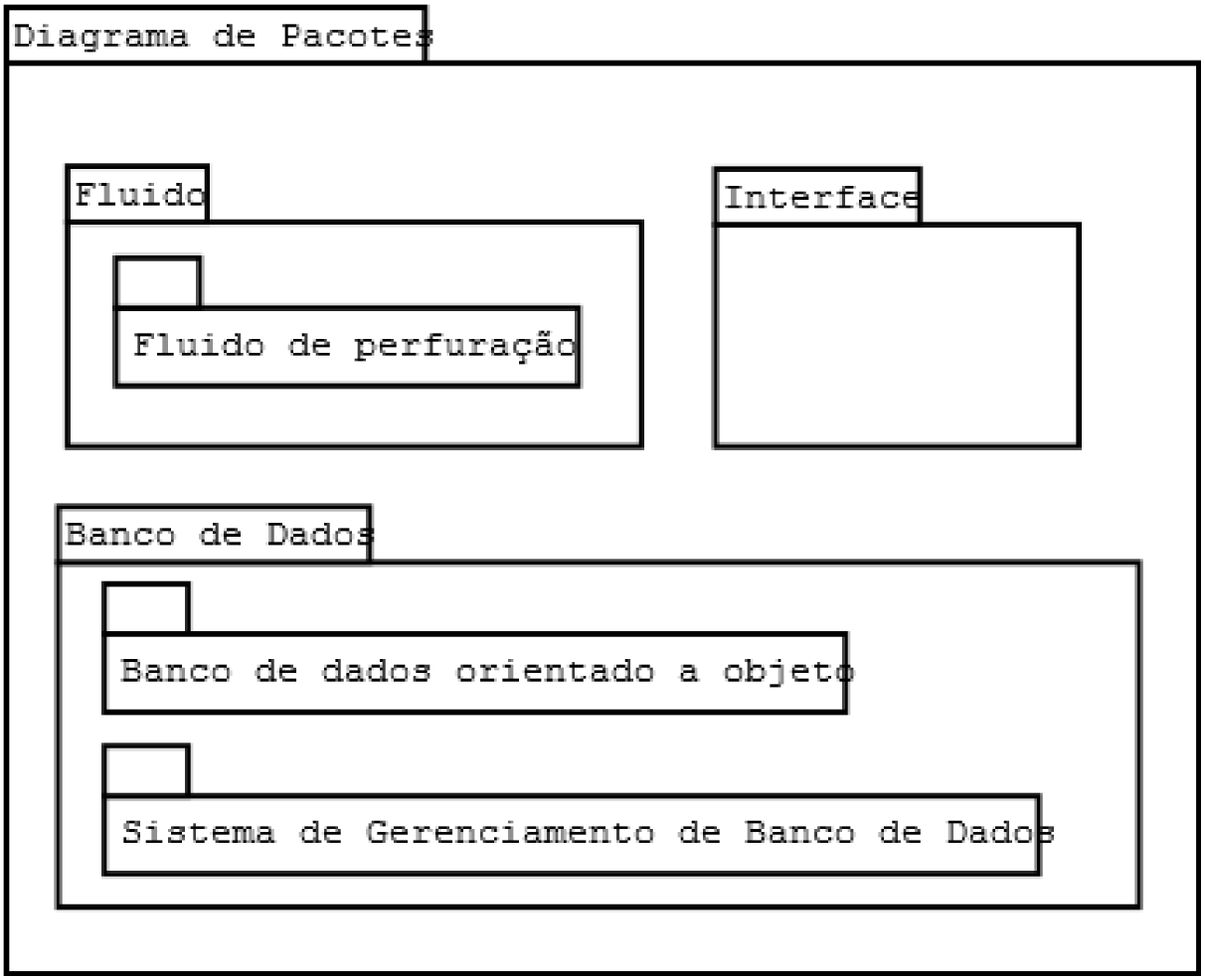

Figura 3.1: Diagrama de pacotes

Capítulo 4

AOO – Análise Orientada a Objeto

A terceira etapa do desenvolvimento de um sistema 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 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 CFluidodePerfuracao: representa a classe que contém informações do fluido de perfuração como seus aditivos, base, propriedades físicas e químicas..
  - atributo chave: representa a chave de identificação do fluido.
  - atributo base: representa o tipo de base que compõe o fluido: água, óleo ou gás.

- atributo teorbase: representa o teor (%) de base no fluido.
- atributo teoragua: representa o teor (%) de água no fluido.
- atributo adensante: representa o adensante usado podendo ser dolomita, por exemplo.
- atributo conc_adensante: representa a concentração de adensante no fluido de perfuração.
- atributo inibidor_fa: representa o inibidor de formações ativas (sal) usado podendo ser cloreto de sódio, por exemplo.
- atributo conc_inibidor_fa: representa a concentração do sal no fluido de perfuração.
- atributo redutor_f: representa o redutor de filtrado usado podendo ser amido, por exemplo.
- atributo conc_redutor_f: representa a concentração do redutor de filtrado no fluido de perfuração.
- atributo biopolimero: representa o biopolímero usado podendo ser carboximetilcelulose, por exemplo.
- atributo conc_biopolimero: representa a concentração do biopolímero no fluido de perfuração.
- atributo viscosificante: representa o controlador de viscosidade usado podendo ser bentonita, por exemplo.
- atributo conc_viscosificante: representa a concentração do sal no fluido de perfuração.
- atributo dispersante: representa o dispersante usado podendo ser lignosulfonatos, por exemplo.
- atributo conc_dispersante: representa a concentração do dispersante no fluido de perfuração.
- atributo defloculante: representa o defloculante usado podendo ser poliacrilato de cálcio, por exemplo.
- atributo conc_defloculante: representa a concentração do defloculante no fluido de perfuração.
- atributo emulsificante: representa o emulsificante usado podendo ser ácido graxo, por exemplo.
- atributo conc_emulsificante: representa a concentração do emulsificante no fluido de perfuração.
- atributo biocida: representa o biocida usado podendo ser glutaraldeído, por exemplo.

- atributo conc_biocida: representa a concentração do biocida no fluido de perfuração.
- atributo lubrificante: representa o lubrificante usado podendo ser ácido graxo, por exemplo.
- atributo conc_lubrificante: representa a concentração do lubrificante no fluido de perfuração.
- atributo inibidor_c: representa o inibidor de corrosão usado podendo ser ácidos de cadeia longa, por exemplo.
- atributo conc_inibidor_c: representa a concentração do inibidor de corrosão no fluido de perfuração.
- atributo controlador_pH: representa o controlador de pH usado podendo ser hidróxido de sódio, por exemplo.
- atributo conc_controlador_pH: representa a concentração do controlador de pH no fluido de perfuração.
- atributo pH_min: representa o valor inferior da faixa de pH para o fluido desenvolvido.
- atributo pH_max: representa o valor superior da faixa de pH para o fluido desenvolvido.
- atributo pe_min: representa o valor inferior da faixa de peso específico para o fluido desenvolvido.
- atributo pe_max: representa o valor superior da faixa de peso específico para o fluido desenvolvido.
- atributo gel_i_ae: representa a força gel, forças atrativas elétricas dentro de um fluido de perfuração, quando submetido às condições estáticas após 10 segundos e antes do envelhecimento;
- atributo temp_e: representa a temperatura de envelhecimento, temperatura da estufa aquecedora na qual o fluido foi inserido numa célula de aço;
- atributo gel_i_de: representa a força gel, forças atrativas elétricas dentro de um fluido de perfuração, quando submetido às condições estáticas após 10 segundos e depois do envelhecimento;
- atributo gel_f_ae: representa a força gel, forças atrativas elétricas dentro de um fluido de perfuração, quando submetido às condições estáticas após 10 minutos e antes do envelhecimento;
- atributo gel_f_de: representa a força gel, forças atrativas elétricas dentro de um fluido de perfuração, quando submetido às condições estáticas após 10 minutos e depois do envelhecimento;

  - atributo est_el_ae: representa a estabilidade elétrica medida pela voltagem requerida para iniciar um fluxo de corrente no fluido antes do envelhecimento.
  - atributo est_el_de: representa a estabilidade elétrica medida pela voltagem requerida para iniciar um fluxo de corrente no fluido depois do envelhecimento.
  - atributo c_lubricidade: representa o coeficiente de lubricidade do fluido de perfuração;
  - atributo filtrado: representa o volume de filtrado obtido em análise laboratorial.
  - atributo VA_ae: representa a viscosidade aparente do fluido antes do envelhecimento.
  - atributo VA_de: representa a viscosidade aparente do fluido depois do envelhecimento.
  - atributo VP_ae: representa a viscosidade plástica do fluido antes do envelhecimento.
  - atributo VP_de: representa a viscosidade plástica do fluido depois do envelhecimento.
  - atributo LE_ae: representa o limite de escoamento do fluido antes do envelhecimento.
  - atributo LE_de: representa o limite de escoamento do fluido depois do envelhecimento.
  - atributo teor_solidos: representa o teor (%) de sólidos presentes no fluido.
  - atributo salinidade: representa a salinidade do fluido.

- Classe CBancodeDados: representa toda estrutura para o desenvolvimento e gerenciamento do banco de dados.

  - atributo int_chave: inteiro que armazena uma linha do arquivo com as chaves.
  - atributo chave_aux: vetor que armazena as chaves lidas no arquivo.
  - atributo item: armazena os titulos e subtitulos nos arquivos com as informações dos fluidos.
  - atributo tipo:
  - atributo resposta:armazena resposta do usuário.
  - atributo filtro: representa o filtro a ser utilizado na pesquisa ao banco de dados.
  - atributo dia:representa o dia de cadastramento da amostra.
  - atributo mes:representa o mês de cadastramento da amostra.
  - atributo ano: representa o ano de cadastramento da amostra.

- atributo num_int: representa um número inteiro.
- atributo caminho_fluido: representa o caminho dos arquivos que contém as informações dos fluidos no computador.
- atributo caminho_bd: representa o caminho do banco de dados no computador.
- atributo fluido: vetor do tipo CFluidodePerfuração de fluidos de perfuração.
- atributo fluido_aux: vetor auxiliar;
- atributo i: contador.
- atributo num: representa um numero.
- char ch: representa um caracter.
- atributo num_double:representa um número do tipo double.
- atributo const_vazio_num_double: representa um número do tipo double.
- atributo caminho_amostra: representa o caminho que se encontra a pasta da amostra de fluido.
- atributo diretorio_amostra: representa o caminho que se encontra o diretorio.
- atributo nome_amostra: representa o nome da amostra.
- atributo item_amostra: representa .
- atributo comando_remover_str: comando para remover amostra em string.
- atributo dado_str:
- atributo nome: representa o nome dado pelo usuário para a amostra.
- atributo caminho: representa o caminho que se encontra a pasta da amostra de fluido.
- atributo linha_str: representa a leitura de uma linha do arquivo.
- atributo vazio_str
- const_vazio_str
- atributo fin: variavel de leitura do arquivo.
- atributo fout: variavel de escrita do arquivo.
- atributo chave: Vetor de chaves lidas no arquivo.
- atributo amostra_fluido: vetor de amostras
- atributo amostra_fluido_vec_aux: vetor de amostras auxiliar.
- atributo amostra_fluido_vec_aux_2: vetor de amostras auxiliar.
- atributo vetor_resposta: vetor que armazena resposta.
- atributo ss: stringstream para conversao de inteiro para string stringstream.

  - atributo comando_remover: cria comando para remover amostra para lixeira.
  - atributo comando_caminho_amostra: cria comando para criar um caminho para o diretório.
  - atributo comando: cria comando para criar o diretório.
  - atributo caminho_bd: cria comando para criar um caminho para a pasta bd.
  - atributo linha_ss:
  - atributo num_double_ss:
  - método Listar_fluido(vector<CFluidodePerfuracao>): Lista os fluidos de perfuração.
  - método Listar_fluido(vector<CFluidodePerfuracao>,int): Lista fluidos de acordo com um filtro.
  - método Exibir_fluido(int chave_): Exibe um determinado fluido.
  - método Inserir_fluido(): Insere um fluido ao banco de dados.
  - método Excluir_amostra(int): Exclui um fluido do banco de dados.
  - método Ler_fluido_basico(): Realiza a leitura dos arquivos referentes aos fluidos.
  - método ler_amostra_basico(int): : Realiza a leitura dos arquivos referentes aos fluidos filtrados.
  - método ler_informacao(int): Ler informações dos fluidos.
  - método teste_vazio(double):
  - método teste_vazio(string):

- Classe CInterface: representa a interface em modo texto do programa.

  - atributo resposta: representa a resposta ao usuário referente ao menu de opções.
  - atributo item:
  - atributo c:contador.
  - atributo comando_abrir:
  - atributo titulo_ss:
  - atributo titulo: representa os títulos que aparecerão para o usuário ao selecionar uma opção do menu.
  - atributo informacao_str:informação da amostra.
  - atributo programa:
  - atributo senha: senha do administrador para configurar o programa.

  - atributo senha_confirma: senha a ser confirmada inserida pelo usuário.
  - atributo ch:caracter.
  - atributo amostras: vetor do tipo vector<CFluidodePerfuracao> de amostras.
  - atributo vetor_resposta: vetor do tipo vector<int> que armazena as respostas.
  - método inicia(): método para iniciar o programa.
  - método abre_DT(): método para abrir diretório de trabalho.
  - método abre_bd():método para abrir pasta do banco de dados.
  - método abre_relatorio(int): método para abrir relatórios.
  - método ajusta_janela(): método para ajustar janela.
  - método limpa_janela(): método para limpar janela.
  - método gerenciar_usuario(CUsuario&): método para gerenciar usuário (administrador e usuário comum).

- Classe CMenu: representa a classe com todos menus.

  - atributo resposta: representa a resposta do usuário.
  - atributo resposta_str: representa a resposta do usuário convertida para string.
  - atributo ch: representa um caracter.
  - atributo vetor_resposta: vetor do tipo vector<int> que armazena a resposta do usuário..
  - método menu_inicio(): método referente ao menu de início.
  - método menu_banco_de_dados(): método referente ao menu que lista as funções referentes ao banco de dados.
  - método menu_exportar_dados(): menu referente à geração de relatórios.
  - método menu_banco_de_dados_pesquisar(): menu referente às opções de pesquisa.
  - método menu_filtro(): menu referente às opções de filtro.
  - método menu_configuracoes(): menu referente às opções de configurações.
  - método menu_sim_nao(): menu referente à resposta do usuário (sim ou não).
  - método menu_exibir_voltar(): menu referente às opções exibir ou voltar.
  - método menu_usuario(): menu oculto referente ao gerenciamento de usuário.

- Classe CRelatorio: representa a classe que exporta as amostras em forma de relatório.

  - atributo chave: representa a chave da amostra a ser exportada.

  - atributo int_chave: representa a chave da amostra a ser exportada.
  - atributo resposta: representa a resposta do usuário.
  - atributo c: contador.
  - atributo v_amostra: vetor do tipo vector<CFluidodePerfuracao> de amostra.
  - método exportar_amostra(int): método para exportar as amostras em forma de relatório.

- Classe CUsuario: representa a classe que gerencia os tipos de usuário.
  - atributo tipo: representa o tipo de usuário a ser escolhido.
  - atributo senha: representa a senha do administrador.
  - atributo senha_crpt: representa a senha criptografada.
  - atributo comando_str: representa um comando para
  - atributo comando_ss::
  - método set_tipo(int): método para setar o tipo de usuário.
  - método get_tipo(): método para retornar o tipo de usuário.
  - método set_senha(string): método para setar a senha do administrador.
  - método get_senha(): método para retornar a senha do administrador.
  - método set_senha_crpt_arquivo(string): método para setar a senha do administrador criptografada.
  - método get_senha_crpt_arquivo(): método para retornar a senha do administrador criptografada.

- CConfiguracao: representa a classe que configura o explorador e o editor de texto.
  - atributo caminho_ss:
  - atributo caminho_str:
  - atributo nome_programa: nome do programa definido pelo usuário.
  - atributo nome_explorador_linux: representa o explorador do linux.
  - atributo nome_explorador_windows: representa o explorador do windows.
  - atributo nome_editor_texto_linux: representa o nome do editor texto do linux.
  - atributo nome_editor_texto_windows: representa o nome do editor texto do windows.
  - atributo tipo_programa: representa o tipo de programa.

  - atributo str;
  - atributo opcao: representa a opção selecionada pelo usuário no menu.
  - método opcao_abrir_DT(): método para abrir diretório.
  - método opcao_abrir_DT(int): método para abrir diretório.
  - método opcao_programa(string, string): método para a troca do explorador e do editor de texto.
  - método opcao_programa(string): método para a troca do explorador

- CHeader: representa a classe que implementa os cabeçalhos.
  - atributo status_usuario: representa o status que aparecerá na tela de acordo com o tipo de usuário.
  - atributo i: contador.
  - atributo largura: vetor de largura do retângulo do titulo.
  - método header_carregando(): referente à animação de carregamento do programa.
  - método header_titulo(int): escreve titulo do programa.
  - método header_inicio(): escreve subtítulo do programa.
  - método header_banco_de_dados(): escreve subtítulo de listar amostras.
  - método header_banco_de_dados_lista_de_amostras(): escreve subtítulo de amostras listadas.
  - método header_banco_de_dados_exibir_amostras(): escreve subtitulo de exibir amostras.
  - método header_banco_de_dados_pesquisa_de_amostras(): escreve subtitulo de amostras listadas.
  - método header_banco_de_dados_inserir_amostras(): escreve subtitulo de inserir amostras.
  - método header_banco_de_dados_excluir_amostras(): escreve subtitulo de excluir amostras.
  - método header_banco_de_dados_exportar_amostras(): escreve subtitulo de excluir amostras.
  - método header_banco_de_dados_informacao_amostras(): escreve subtitulo de informações de amostras.
  - método header_configuracoes(): escreve subtitulo de configurações.
  - método header_menu_abre_DT(): escreve subtitulo do menu abrir diretório.
  - método header_menu_abre_bd(): escreve subtitulo do menu banco de dados.
  - método header_tipo_usuario(): escreve subtitulo do menu.

4.2 Diagrama de sequência

O diagrama de sequência enfatiza a troca de eventos e mensagens e sua ordem temporal. Contém informações sobre o fluxo de controle do programa. Estabelece o relacionamento dos atores (usuários e sistemas externos) com alguns objetos do sistema.

Veja o diagrama de sequência na Figura 4.2.

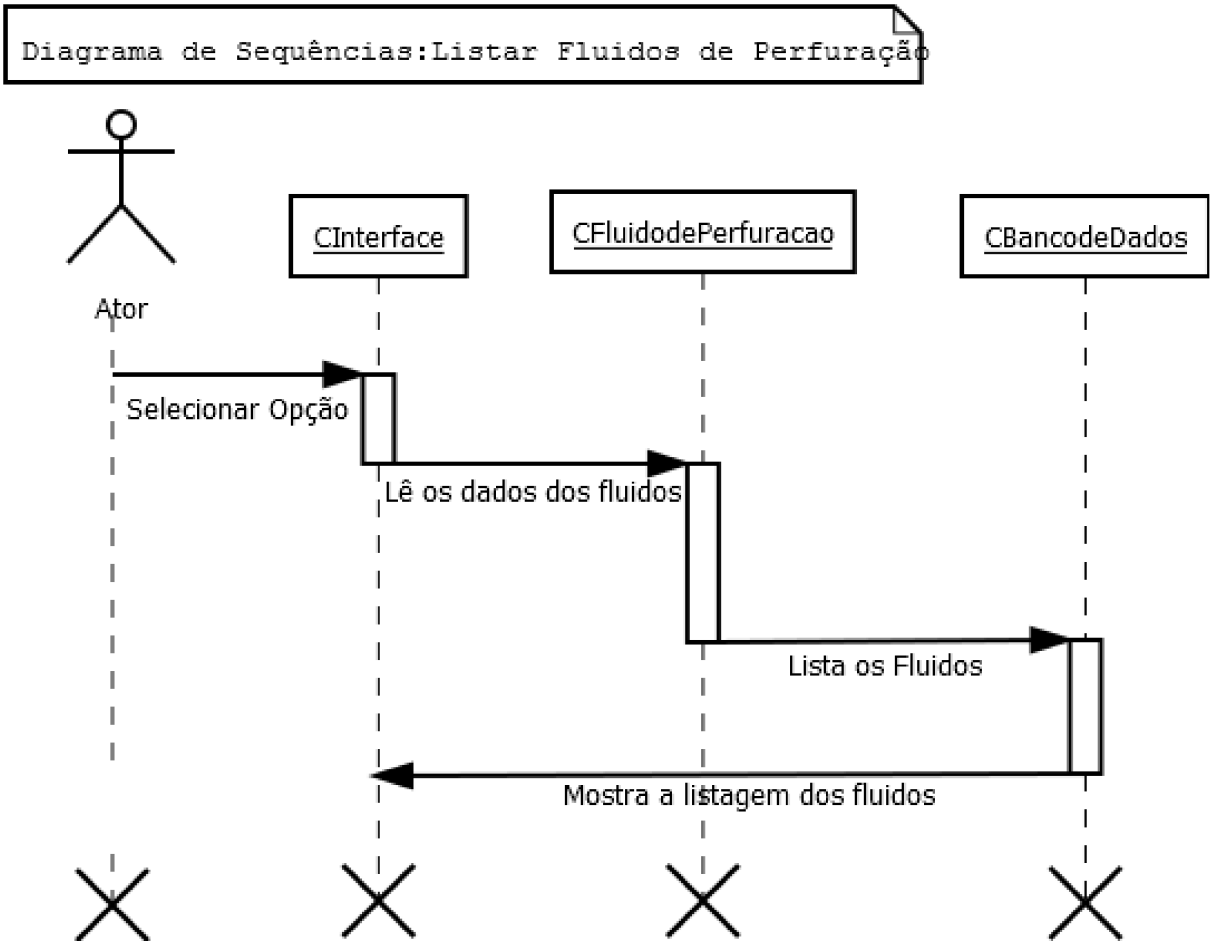

Figura 4.2: Diagrama de seqüência: Listar Fluidos de Perfuração

4.3 Diagrama de colaboração

O diagrama de colaboração pode ser desenvolvido como uma extensão do diagrama de caso de uso, detalhando o mesmo por meio da inclusão de objetos, mensagens e parâmetros trocados entre objetos. O principal objetivo deste diagrama é a interação e a troca de mensagens e dados entre os objetos.

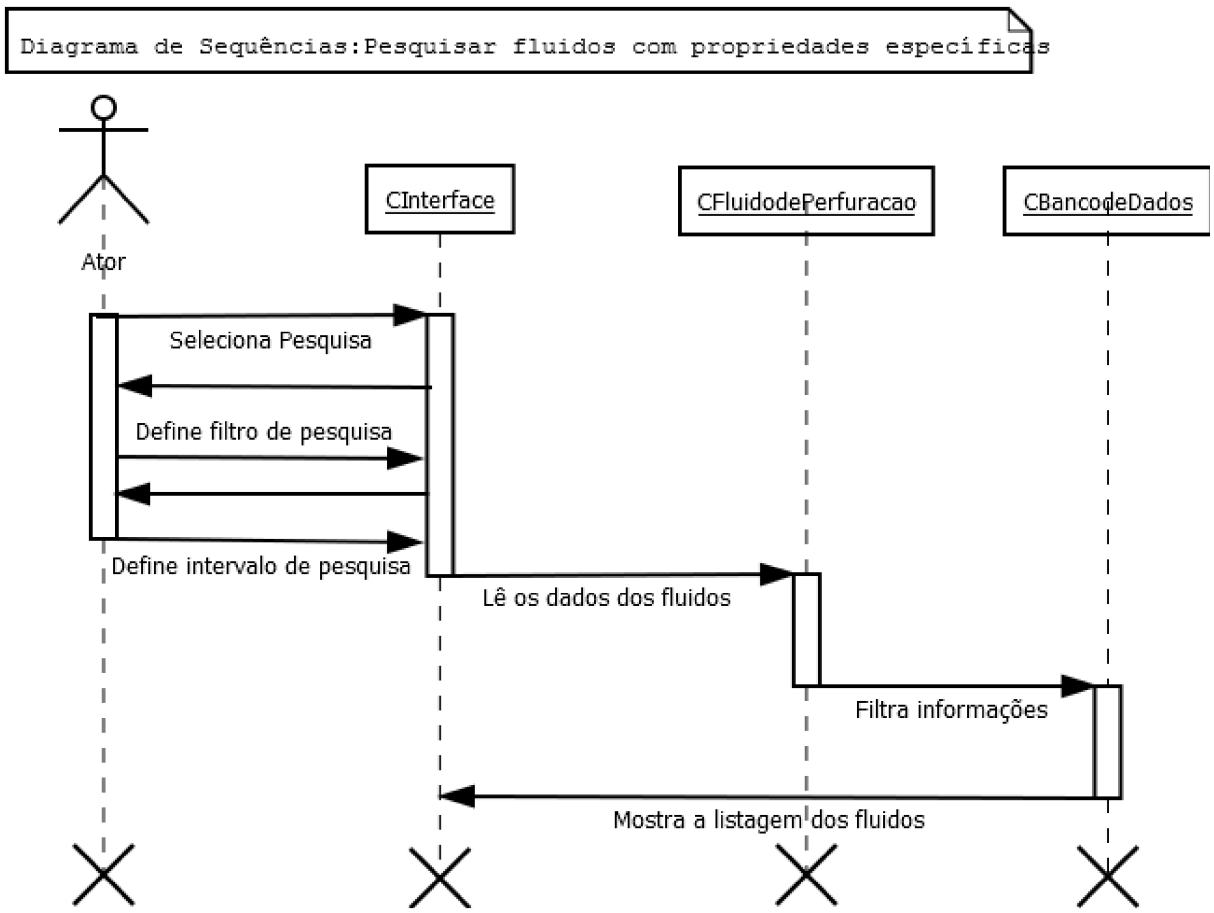

Figura 4.3: Diagrama de Sequência:Pesquisar Fluidos com propriedades específicas

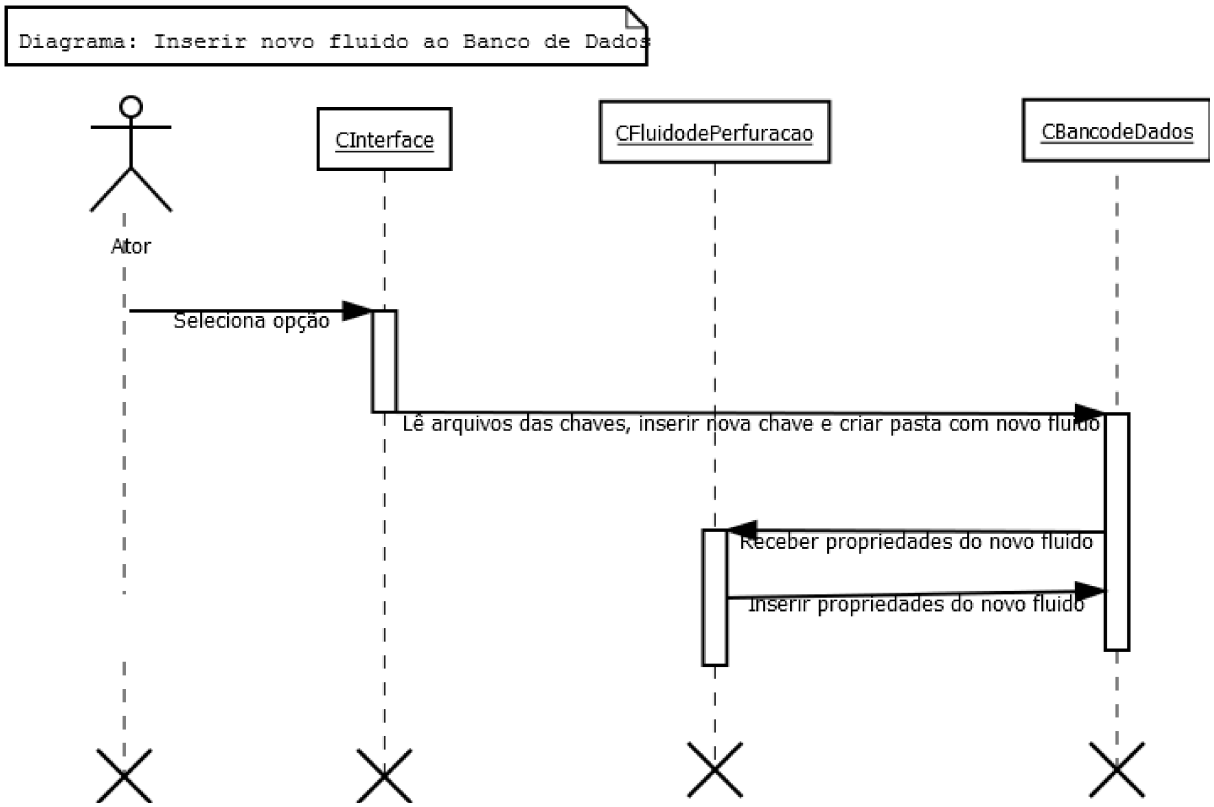

Figura 4.4: Diagrama se Sequência:Inserir Novo Fluido ao banco de Dados

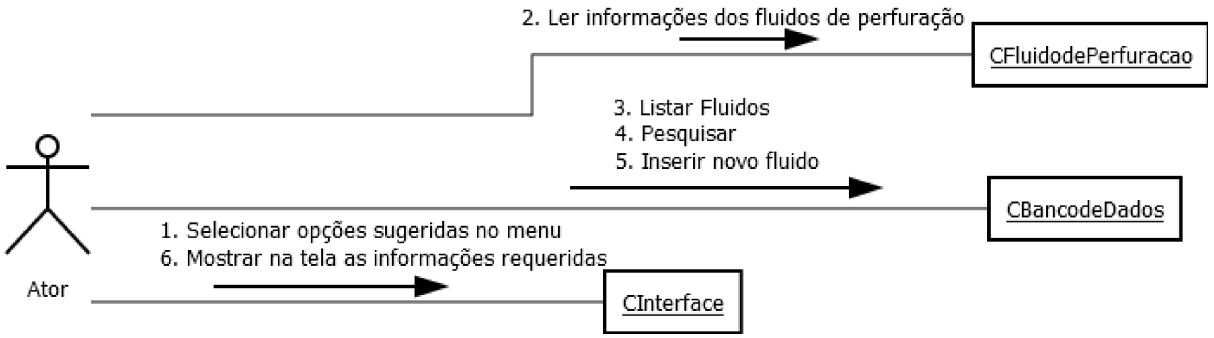

Figura 4.5: Diagrama de colaboração

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. Existem basicamente dois usos para máquinas de estado: máquinas de estado comportamentais e máquinas de estado para protocolos.

Máquinas de estado comportamentais podem ser utilizadas para especificar o comportamento de vários tipos de elementos. Por exemplo, podem ser utilizadas para modelar o comportamento de entidades individuais (objetos), por meio da modificação dos valores de seus atributos.

Máquinas de estado para protocolos expressam as transições legais que um objeto pode desenvolver. Com seu uso, pode-se definir o ciclo de vida de objetos, ou uma determina ordem na invocação de suas operações. Para este tipo de máquina de estado, interfaces e portas podem estar associados.

Veja na Figura 4.6 o diagrama de máquina de estado para o banco de dados.

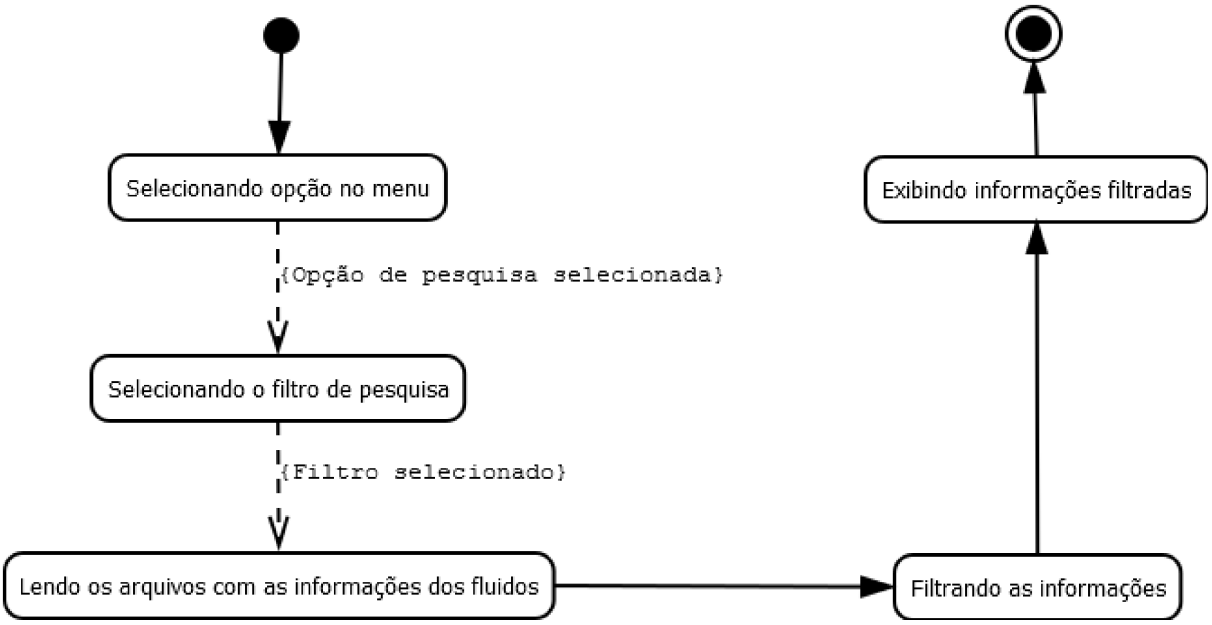

Figura 4.6: Diagrama de máquina de estado

4.5 Diagrama de atividades

O diagrama de atividades é um diagrama UML utilizado para modelar o aspecto comportamental de processos. Neste diagrama, uma atividade é modelada como uma sequência estruturada de ações controladas por nós de decisão e sincronismo.

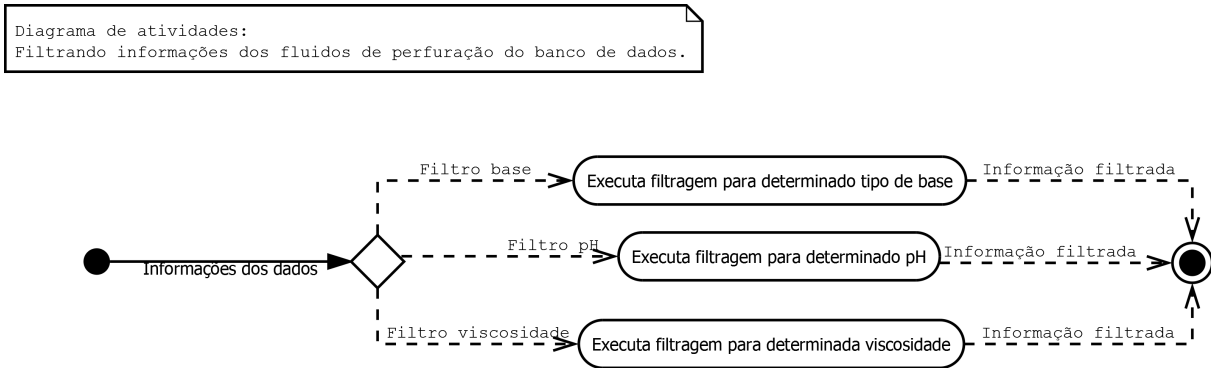

Figura 4.7: Diagrama de atividades

Capítulo 5

Projeto

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 biblitoecas 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 para o nome das classes, padrões de retorno e de parâmetros em métodos, características da interface do programa e características de desempenho.

5.1 Projeto do sistema

Segundo Rumbaugh et al. 1994, o projeto do sistema é a estratégia de alto nível para resolver o problema e elaborar uma solução. Deve-se preocupar com itens como:

• Definição do protocolo de comunicação entre os diversos elementos externos (como dispositivos);

• Definição de loops de controle, das escalas de tempo;

• Identificação de subsistemas;

• Identificação de concorrências;

• Identificação de depósitos de dados (implicam em modicações no diagrama de atividades);

• Identificação e alocação dos recursos globais, das condições extremas e de prioridades (implicam em modicações no diagrama de componentes);

• Identificação e seleção da implementação de controle (implicam em modicações no diagrama de implantação);

• Identificação das estruturas arquitetônicas comuns.

1. Protocolos

- Definição dos protocolos de comunicação entre os diversos elementos externos.
  - Esta versão do software não inclui comunicação com elementos externos (internet).
- Definição dos protocolos de comunicação entre os diversos elementos internos.
  - O programa utilizará uma máquina computacional com HD, processador, teclado para a entrada de dados e o monitor para a saída de dados. Os arquivos gerados pelo programa estarão em formato de texto em um banco de dados.
- Definição do formato dos arquivos gerados pelo programa.
  - Os arquivos de texto serão gerados em formato ASCII (não formatado).

2. Recursos

- Identificação e alocação dos recursos globais, como os recursos do sistema serão alocados, utilizados, compartilhados e liberados. Implicam modificações no diagrama de componentes.
- Identificação da necessidade do uso de banco de dados. Implicam em modificações nos diagramas de atividades e de componentes.
  - O programa implementará um banco de dados.

3. Controle

- Identificação da necessidade de otimização.
- Identificação e definição de *loops* de controle e das escalas de tempo.
- Identificação de concorrências.
  - Não será considerado nessa versão.

4. Plataformas

- Identificação das estruturas arquitetônicas comuns.
  - O programa será desenvolvido em linguagem de programação orientada a objeto C++.
- Identificação e definição das plataformas a serem suportadas: hardware, sistema operacional e linguagem de programação.
  - O programa será desenvolvido em computador Intel 32/64 bits, com sistema operacional Windows usando linguagem C++ orientada a objeto.
- Seleção das bibliotecas externas

5. Subsistemas

- Identificação dos subsistemas:
  - Fluido de Perfuração;
  - Perfuração;
  - Banco de Dados;

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 programa, 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.

Efeitos do projeto no modelo estrutural

- Novas classes e associações oriundas das bibliotecas selecionadas e da linguagem escolhida devem ser acrescentadas ao modelo.
  - Após a análise e o projeto do sistema surgiu a necessidade da criação de novas classes e associações. Problemas como esse poderão surgir durante a implementação do banco de dados, sendo assim passível de modificação ou criação de novas classes, atributos e métodos.
- Estabelecer as dependências e restrições associadas à plataforma escolhida.

Efeitos do projeto no modelo dinâmico

- Revisar os diagramas de sequência e de colaboração considerando a plataforma escolhida.
- Verificar a necessidade de se revisar, ampliar e adicionar novos diagramas de máquina de estado e de atividades.

Efeitos do projeto nos atributos

- Atributos novos podem ser adicionados a uma classe, como, por exemplo, atributos específicos de uma determinada linguagem de programação (acesso a disco, ponteiros, constantes e informações correlacionadas).

Efeitos do projeto nas heranças

- Reorganização da herança no diagrama de classes.

Efeitos do projeto nas associações

- Reorganização das associações.

Efeitos do projeto nas otimizações

- Otimização do sistema.
- Identifique pontos a serem otimizados em que podem ser utilizados processos concorrentes.

5.3 Diagrama de componentes

O diagrama de componentes (Veja Figura 5.1) mostra a forma como os componentes do programa 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 e código-fonte.

5.4 Diagrama de Execução

O diagrama de implantação(execução) é um diagrama de alto nível que inclui relações entre o sistema e o hardware e que se preocupa com os aspectos da arquitetura computacional escolhida. Seu enfoque é o hardware.

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

Para o banco de dados será necessário apenas um computador que possua HD, teclado para a entrada de dados e monitor para a saída de dados.Também não será preciso uma rede de computadores, visto que o banco de dados depende somente da interação usuário - simulador.

Diagrama de Execução

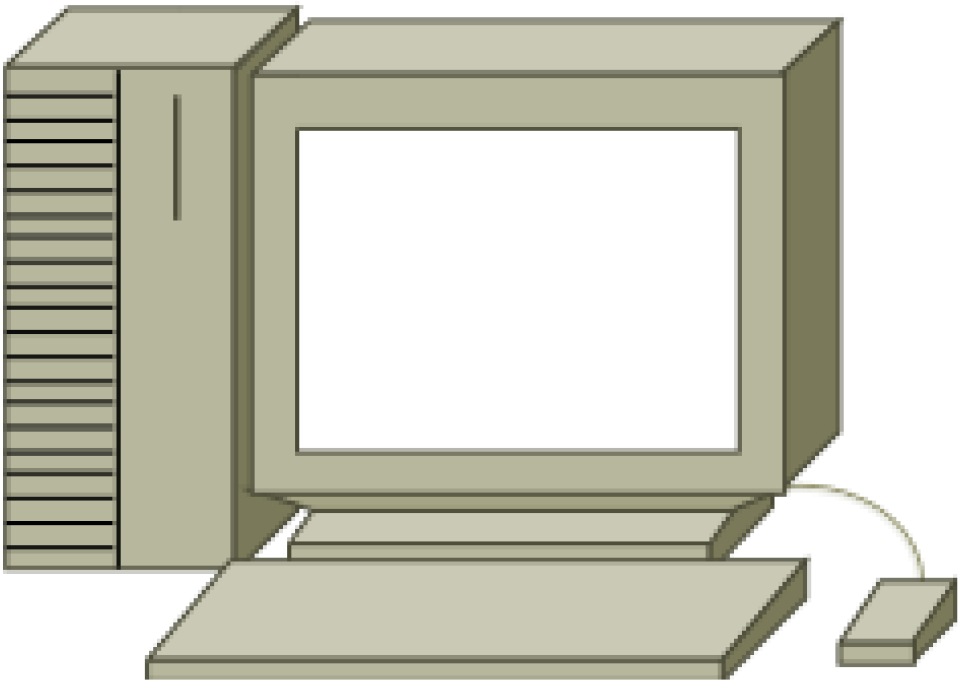

Figura 5.1: Diagrama de Execução.

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 CFluidodePerfuracao.

Listing 6.1: Arquivo de cabeçalho da classe CFluidodePerfuracao.

```
1 #ifndef CFluidodePerfuracao_h
2 #define CFluidodePerfuracao_h
3 #include <string>
4 #include <vector>
5 #include <iostream>
6 using namespace std;
7
8 class CFluidodePerfuracao
9 {
10 public:
11   //Atributos
12
13   //Componentes
14   int chave; // representa a chave de identificação do fluido.
15   string nome_do_fluido;
16   int dia;
17   int mes;
18   int ano;
19   string base; //representa o tipo de base que compõe o fluido, água,
       óleo ou gás.
20   double teorbase; //representa o teor(%) de base no fluido.
21   double teoragua; //representa o teor(%) de água no fluido.
22
```

```
23  //Propriedades fisicas e quimicas
24  double pH_min; //representa o valor inferior da faixa de pH para o
        fluido desenvolvido.
25  double pH_max; //representa o valor superior da faixa de pH para o
        fluido desenvolvido.
26  double pe_min; //representa o valor inferior da faixa de peso
        específico para o fluido desenvolvido.
27  double pe_max; //representa o valor superior da faixa de peso
        específico para o fluido desenvolvido.
28  double gel_i_ae; //representa a força gel, forças atrativas elétricas
        dentro de um fluido de perfuração, quando submetido às condições
        estáticas após 10 segundos e antes do envelhecimento;
29  double temp_e; //representa a temperatura de envelhecimento,
        temperatura da estufa aquecedora na qual o fluido foi inserido numa
         célula de aço;
30  double gel_i_de; //representa a força gel, forças atrativas elétricas
        dentro de um fluido de perfuração, quando submetido às condições
        estáticas após 10 segundos e depois do envelhecimento;
31  double gel_f_ae; //representa a força gel, forças atrativas elétricas
        dentro de um fluido de perfuração, quando submetido às condições
        estáticas após 10 minutos e antes do envelhecimento;
32  double gel_f_de; //representa a força gel, forças atrativas elétricas
        dentro de um fluido de perfuração, quando submetido às condições
        estáticas após 10 minutos e depois do envelhecimento;
33  double est_el_ae; //representa a estabilidade elétrica medida pela
        voltagem requerida para iniciar um fluxo de corrente no fluido
        antes do envelhecimento.
34  double est_el_de; //representa a estabilidade elétrica medida pela
        voltagem requerida para iniciar um fluxo de corrente no fluido
        depois do envelhecimento.
35  double c_lubricidade; //representa o coeficiente de lubricidade do
        fluido de perfuração;
36  double filtrado; //representa o volume de filtrado obtido em análise
        laboratorial.
37  double VA_ae; //representa a viscosidade aparente do fluido antes do
        envelhecimento.
38  double VA_de; //representa a viscosidade aparente do fluido depois do
        envelhecimento.
39  double VP_ae; //representa a viscosidade plástica do fluido antes do
        envelhecimento.
40  double VP_de; //representa a viscosidade plástica do fluido depois do
        envelhecimento.
41  double LE_ae; //representa o limite de escoamento do fluido antes do
        envelhecimento.
42  double LE_de; //representa o limite de escoamento do fluido depois do
        envelhecimento.
43  double teor_solidos; //representa o teor(%) de sólidos presentes no
        fluido.
```

```
44  double salinidade; //representa a salinidade do fluido.
45
46  //ADiTIVOS
47
48  string adensante; //representa o adensante usado podendo ser dolomita,
        por exemplo.
49  double conc_adensante; //representa a concentração de adensante no
        fluido de perfuração.
50  string inibidor_fa; //representa o inibidor de formações ativas (sal)
        usado podendo ser cloreto de sódio, por exemplo.
51  double conc_inibidor_fa; //representa a concentração do sal no fluido
        de perfuração.
52  string redutor_f; //representa o redutor de filtrado usado podendo ser
        amido, por exemplo.
53  double conc_redutor_f; //representa a concentração do redutor de
        filtrado no fluido de perfuração.
54  string biopolimero; //representa o biopolímero usado podendo ser
        carboximetilcelulose, por exemplo.
55  double conc_biopolimero; //representa a concentração do biopolímero no
        fluido de perfuração.
56  string viscosificante; //representa o controlador de viscosidade usado
        podendo ser bentonita, por exemplo.
57  double conc_viscosificante; //representa a concentração do sal no
        fluido de perfuração.
58  string dispersante; //representa o dispersante usado podendo ser
        lignosulfonatos, por exemplo.
59  double conc_dispersante; //representa a concentração do dispersante no
        fluido de perfuração.
60  string defloculante; //representa o defloculante usado podendo ser
        poliacrilato de cálcio, por exemplo.
61  double conc_defloculante; //representa a concentração do defloculante
        no fluido de perfuração.
62  string emulsificante; //representa o emulsificante usado podendo ser
        ácido graxo, por exemplo.
63  double conc_emulsificante; //representa a concentração do
        emulsificante no fluido de perfuração.
64  string biocida; //representa o biocida usado podendo ser glutaraldeído
        , por exemplo.
65  double conc_biocida; //representa a concentração do biocida no fluido
        de perfuração.
66  string lubrificante; //representa o lubrificante usado podendo ser
        ácido graxo, por exemplo.
67  double conc_lubrificante; //representa a concentração do lubrificante
        no fluido de perfuração.
68  string inibidor_c; //representa o inibidor de corrosão usado podendo
        ser ácidos de cadeia longa, por exemplo.
69  double conc_inibidor_c; //representa a concentração do inibidor de
        corrosão no fluido de perfuração.
```

```
70   string controlador_pH; //representa o controlador de pH usado podendo
         ser hidróxido de sódio, por exemplo.
71   double conc_controlador_pH; //representa a concentração do controlador
          de pH no fluido de perfuração.
72
73 };
74
75 #endif
```

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

Listing 6.2: Arquivo de implementação da classe CFluidodePerfuracao.

```
77 #include <iostream>
78 #include <stdio.h>
79 //verifica sistema operacional
80 #ifdef __linux
81
82     #include <cstdlib>
83
84 #elif _WIN32
85
86     #include <windows.h>
87     #include <stdlib.h>//para system
88     #include <conio.h>
89
90 #else
91
92 #endif
93
94 #include <string>
95 #include "CFluidodePerfuracao.h"
```

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

Listing 6.3: Arquivo de cabeçalho da classe CBancodeDados.

```
97 #ifndef CBancodeDados_h
98 #define CBancodeDados_h
99 #include <iostream>
100 #include <fstream>
101 #include <vector>
102 #include <string>
103 #include <sstream>//para conversao de int para string
104 #include "CHeader.h"
105 #include "CConfiguracao.h"
106 #include "CFluidodePerfuracao.h"
107
108 using namespace std;
109
```

```
110 class CBancodeDados
111 {
112
113 public:
114   int int_chave;
115   int int_chave_aux;
116   int item;
117   int tipo;
118   int resposta;//respsota
119   int int_posicao;
120   int filtro;//filtro da pesquisa
121   int dia;
122   int mes;
123   int ano;
124   int num_int;
125
126
127   unsigned int i;//i para contagem do for;
128   double num;
129   char ch;
130   double num_double;
131   double const_vazio_num_double;
132
133   string caminho_amostra;
134   string diretorio_amostra;      //string nome_amostra;
135   string item_amostra;
136   string comando_remover_str;//comando para remover amostra em string
137   string dado_str;
138   string nome;
139   string caminho;
140   string linha_str;
141   string vazio_str;
142   string const_vazio_str;
143
144   ifstream fin;//variavel de leitura do arquivo
145   ofstream fout;
146
147   vector<int> chave;//Vetor de chaves lidas no arquivo
148   vector<CFluidodePerfuracao> amostra_fluido; //vetor de amostras
149   vector<CFluidodePerfuracao> amostra_fluido_vec_aux;
150   vector<CFluidodePerfuracao> amostra_fluido_vec_aux_2;
151   vector<int> vetor_resposta;
152
153   stringstream ss;//stringstream para conversao de inteiro para string
154   stringstream comando_remover;//cria comando para remover amostra para
        lixeira
155   stringstream comando_caminho_amostra;
156   stringstream comando;
```

```
157   stringstream nome_amostra;
158   stringstream caminho_bd;
159   stringstream linha_ss;
160   stringstream num_double_ss;
161
162   CFluidodePerfuracao amostra_aux;
163   CFluidodePerfuracao amostra_fluido_aux;//obj auxiliar
164   CConfiguracao configuracao;
165
166
167 public:
168   void listar_amostra(vector<CFluidodePerfuracao>);//lista todas as
          amostras
169   void listar_amostra(vector<CFluidodePerfuracao>, int);
170   void exibir_amostra(int);
171   void inserir_amostra();//insere amostras
172   void excluir_amostra(int);//exclui amostras
173   vector<CFluidodePerfuracao> ler_amostra_basico();
174   vector<CFluidodePerfuracao> ler_amostra_basico(int);
175   string teste_vazio(double);
176   double teste_vazio(string);
177
178 };
179
180 #endif
```

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

Listing 6.4: Arquivo de implementação da classe CBancodeDados.

```
182 #ifdef __linux
183 #elif _WIN32
184 #include <windows.h>
185 #else
186 #endif
187
188 #include <cstdlib>
189 #include <stdlib.h>//para system
190 #include <stdio.h>
191 #include <string>
192 #include <sstream>//para conversao de int para string
193 #include <vector>
194 #include <fstream>
195 #include <iostream>
196 #include <cstring>
197 #include "CBancodeDados.h"
198 #include "CFluidodePerfuracao.h"
199
200 using namespace std;
201
```

```
202 vector<CFluidodePerfuracao> CBancodeDados::ler_amostra_basico()//funcao
      listar amostras
203 {
204 //
      ////////////////////////////////////////////////////////////////////////////////

205 //Inicio da leitura das chaves (Primeiro le o arquivo de informacao do
      banco de dados)
206 //
      ////////////////////////////////////////////////////////////////////////////////

207   vector<int> chave;//vetor que armazena as chaves
208   vector<CFluidodePerfuracao> amostra_fluido;
209   ifstream fin;//objeto de leitura
210   int int_chave;//inteiro que recebe cada linha lida no arquivo
211   unsigned int i;//inteiro para contagem no for
212
213   #ifdef __linux
214   caminho_bd<<"bd"<<char(47)<<"ini"<<char(47)<<"bdinfo";
215   #elif _WIN32
216   caminho_bd<<"bd"<<char(92)<<"ini"<<char(92)<<"bdinfo";
217   #else
218
219   #endif
220
221   fin.open(caminho_bd.str().c_str());//abre o arquivo de informacoes do
        bd
222   caminho_bd.str("");
223   while(! fin.eof())     //carrega o arquivo de informacoes do bd
224   {
225     fin >> int_chave;
226     chave.push_back(int_chave);
227   }
228   fin.close();
229
230 ////////////////////////////////////////////
231 //Termino da leitura das chaves
232 ////////////////////////////////////////////
233
234 ////////////////////////////////////////////
235 //Inicio da leitura das amostras
236 ////////////////////////////////////////////
237
238   CFluidodePerfuracao amostra_fluido_aux;//objeto amostra fluido
        auxiliar
239
240   string caminho_amostra;//caminho da amostra a ser lida incluindo o
        nome dela
```

```
241  string item_amostra;//string para ler os titulos e subtitulos dos
         respectivos itens das amostras
242  stringstream ss;//stringstream para conversao de inteiro para string
243  num_int=0;//inicia variavel auxiliar
244  i=0;
245
246  for(i=0;i<chave.size();i++)//For para leitura de todas as amostras
247  {
248
249    #ifdef __linux
250    ss<<"bd"<<char(47)<<chave[i]<<char(47)<<chave[i];//"bd/chave/chave"
         e exatamente assim que o fin le o arquivo(linux)
251    #elif _WIN32
252    ss<<"bd"<<char(92)<<chave[i]<<char(92)<<chave[i];//"bd\chave\chave"
         e exatamente assim que o fin le o arquivo(windows)
253    #else
254
255    #endif
256    caminho_amostra=ss.str();//converte o caminho da amostra incluindo o
         nome para string
257    ss.str("");//apaga conteudo de ss
258    fin.open(caminho_amostra.c_str());//abre todas as amostras
         existentes
259
260    //
         ////////////////////////////////////////////////////////////////////////////

261    //lÃª o arquivo de texto e atribui os dados do respectivo objeto
         amostra_fluido[i]////
262    //
         ////////////////////////////////////////////////////////////////////////////

263    //Nesta etapa usa-se o objeto amostra_fluido_aux para depois de
         setado ser incluido//
264    //no vetor de amostras
                                                                          //
265    //
         ////////////////////////////////////////////////////////////////////////////


266
267    fin>>item_amostra;//le a string "-chave"
268    fin>>amostra_fluido_aux.chave;//le a chave da amostra i
269    fin>>item_amostra;//le a string data
270    fin>>item_amostra;//le a string  dia:
271    fin>>amostra_fluido_aux.dia;//le a string
272    fin>>item_amostra;//le a string  mes:
273    fin>>amostra_fluido_aux.mes;//le a string
274    fin>>item_amostra;//le a string  ano:
```

```
275     fin>>amostra_fluido_aux.ano;//le a string
276     fin>>item_amostra;//le a string -nome
277     fin>>amostra_fluido_aux.nome_do_fluido;//le o nome do fluido
278     fin>>item_amostra;//le a string -base
279     fin>>amostra_fluido_aux.base;//le o nome da base
280     fin>>item_amostra;//le a string -teorbase
281     fin>>amostra_fluido_aux.teorbase;//le o teor da base
282     fin>>item_amostra;//le a string -teor_de_agua
283     fin>>amostra_fluido_aux.teoragua;//le o teor de agua
284     fin>>item_amostra;//le a string linha
285     fin>>item_amostra;//le a string PROPRIEDADES FÍSICAS E QUÍMICAS
286     fin>>item_amostra;//le a string pH_min
287     fin>>amostra_fluido_aux.pH_min;//le o pH_min
288     fin>>item_amostra;//le a string pH_max
289     fin>>amostra_fluido_aux.pH_max;//le o pH_max
290     fin>>item_amostra;//le a string --pe_min
291     fin>>amostra_fluido_aux.pe_min;//le pe_min
292     fin>>item_amostra;//le a string --pe_max
293     fin>>amostra_fluido_aux.pe_max;//le pe_max
294     fin>>item_amostra;//le a string --temp_e
295     fin>>amostra_fluido_aux.temp_e;//le temp_e
296     fin>>item_amostra;//le a string --gel_i_ae
297     fin>>amostra_fluido_aux.gel_i_ae;//le gel_i_ae
298     fin>>item_amostra;//le a string --gel_f_ae
299     fin>>amostra_fluido_aux.gel_f_ae;//le gel_f_ae
300     fin>>item_amostra;//le a string --gel_i_de
301     fin>>amostra_fluido_aux.gel_i_de;//le gel_i_de
302     fin>>item_amostra;//le a string --gel_f_de
303     fin>>amostra_fluido_aux.gel_f_de;//le gel_f_de
304     fin>>item_amostra;//le a string --est_el_ae
305     fin>>amostra_fluido_aux.est_el_ae;//le est_el_ae
306     fin>>item_amostra;//le a string --est_el_de
307     fin>>amostra_fluido_aux.est_el_de;//le est_el_de
308     fin>>item_amostra;//le a string --est_el_de
309     fin>>amostra_fluido_aux.est_el_de;//le est_el_de
310     fin>>item_amostra;//le a string -c_lubricidade
311     fin>>amostra_fluido_aux.c_lubricidade;//le c_lubricidade
312     fin>>item_amostra;//le a string -filtrado
313     fin>>amostra_fluido_aux.filtrado;//le filtrado
314     fin>>item_amostra;//le a string -salinidade
315     fin>>amostra_fluido_aux.salinidade;//le salinidade
316     fin>>item_amostra;//le a string linha
317     fin>>item_amostra;//le a string PROPRIEDADES REOLÚGICAS
318     fin>>item_amostra;//le a string VA_ae
319     fin>>amostra_fluido_aux.VA_ae;//le VA_ae
320     fin>>item_amostra;//le a string VA_de
321     fin>>amostra_fluido_aux.VP_de;//le VA_ae
322     fin>>item_amostra;//le a string VP_ae
```

```
323     fin>>amostra_fluido_aux.VP_ae;//le VP_ae
324     fin>>item_amostra;//le a string VP_de
325     fin>>amostra_fluido_aux.VP_de;//le VP_de
326     fin>>item_amostra;//le a string LE_ae
327     fin>>amostra_fluido_aux.LE_ae;//le LE_ae
328     fin>>item_amostra;//le a string LE_de
329     fin>>amostra_fluido_aux.LE_de;//le LE_de
330     fin>>item_amostra;//le a string linha
331     fin>>item_amostra;//le a string ADITIVOS
332     fin>>item_amostra;//le a string adensante
333     fin>>amostra_fluido_aux.adensante;//le adensante
334     fin>>item_amostra;//le a string conc_adensante
335     fin>>amostra_fluido_aux.conc_adensante;//le conc_adensante
336     fin>>item_amostra;//le a string inibidor_fa
337     fin>>amostra_fluido_aux.inibidor_fa;//le inibidor_fa
338     fin>>item_amostra;//le a string conc_inibidor_fa
339     fin>>amostra_fluido_aux.conc_inibidor_fa;//le conc_inibidor_fa
340     fin>>item_amostra;//le a string redutor_f
341     fin>>amostra_fluido_aux.redutor_f;//le redutor_f
342     fin>>item_amostra;//le a string conc_redutor_f
343     fin>>amostra_fluido_aux.conc_redutor_f;//le conc_redutor_f
344     fin>>item_amostra;//le a string biopolimero
345     fin>>amostra_fluido_aux.biopolimero;//le biopolimero
346     fin>>item_amostra;//le a string conc_biopolimero
347     fin>>amostra_fluido_aux.conc_biopolimero;//le conc_biopolimero
348     fin>>item_amostra;//le a string viscosificante
349     fin>>amostra_fluido_aux.viscosificante;//le viscosificante
350     fin>>item_amostra;//le a string conc_viscosificante
351     fin>>amostra_fluido_aux.conc_viscosificante;//le conc_viscosificante
352     fin>>item_amostra;//le a string dispersante
353     fin>>amostra_fluido_aux.dispersante;//le dispersante
354     fin>>item_amostra;//le a string conc_dispersante
355     fin>>amostra_fluido_aux.conc_dispersante;//le conc_dispersante
356     fin>>item_amostra;//le a string defloculante
357     fin>>amostra_fluido_aux.defloculante;//le defloculante
358     fin>>item_amostra;//le a string conc_defloculante
359     fin>>amostra_fluido_aux.conc_defloculante;//le conc_defloculante
360     fin>>item_amostra;//le a string emulsificante
361     fin>>amostra_fluido_aux.emulsificante;//le emulsificante
362     fin>>item_amostra;//le a string conc_emulsificante
363     fin>>amostra_fluido_aux.conc_emulsificante;//le conc_emulsificante
364     fin>>item_amostra;//le a string biocida
365     fin>>amostra_fluido_aux.biocida;//le biocida
366     fin>>item_amostra;//le a string conc_biocida
367     fin>>amostra_fluido_aux.conc_biocida;//le conc_biocida
368     fin>>item_amostra;//le a string lubrificante
369     fin>>amostra_fluido_aux.lubrificante;//le lubrificante
370     fin>>item_amostra;//le a string conc_lubrificante
```

```
371     fin>>amostra_fluido_aux.conc_lubrificante;//le conc_lubrificante
372     fin>>item_amostra;//le a string inibidor_c
373     fin>>amostra_fluido_aux.inibidor_c;//le inibidor_c
374     fin>>item_amostra;//le a string conc_inibidor_c
375     fin>>amostra_fluido_aux.conc_inibidor_c;//le conc_inibidor_c
376     fin>>item_amostra;//le a string controlador_pH
377     fin>>amostra_fluido_aux.controlador_pH;//le controlador_pH
378     fin>>item_amostra;//le a string controlador_pH
379     fin>>amostra_fluido_aux.conc_controlador_pH;//le conc_controlador_pH
380
381     amostra_fluido.push_back(amostra_fluido_aux);
382     fin.close();
383
384   }
385   return amostra_fluido;
386
387 }
388
389 vector<CFluidodePerfuracao> CBancodeDados::ler_amostra_basico(int
      filtro_)//funcao listar amostras
390 {
391   vector<CFluidodePerfuracao> amostra_fluido_vec_aux;
392   amostra_fluido_vec_aux.clear();
393   amostra_fluido_vec_aux=ler_amostra_basico();//le tododas infos basicas
394   //de todas amostras
395   vector<CFluidodePerfuracao> amostra_fluido;
396   amostra_fluido.clear();
397   int filtro;
398   filtro=filtro_;
399   int int_chave;
400   unsigned int c;
401   double dado_double;
402   string nome;
403   //comeca preencher o vetor de amostras amostra_fluifo
404   amostra_fluido.push_back(amostra_fluido_vec_aux[0]);//retorna a
        amostra zero como base de calculos
405   switch (filtro)//testa a resposta
406
407   {
408     case 1:
409       cout<<"Informe a CHAVE: ";
410       cin>>int_chave;
411       cin.get();
412       for(c=0;c<amostra_fluido_vec_aux.size();c++)//corre amostras
413       {
414         if(amostra_fluido_vec_aux[c].chave==int_chave)
415         {
416           amostra_fluido.push_back(amostra_fluido_vec_aux[c]);
```

```
417         }
418       }
419
420       break;
421
422     case 2:
423       cout<<"Informe o nome do Fluido: ";
424       cin>>nome;
425       cin.get();
426       for(c=0;c<amostra_fluido_vec_aux.size();c++)//corre amostras
427       {
428         if(amostra_fluido_vec_aux[c].nome_do_fluido==nome)
429         {
430           amostra_fluido.push_back(amostra_fluido_vec_aux[c]);
431         }
432       }
433
434       break;
435
436     case 3:
437       cout<<"Informe a Base: ";
438       cin>>nome;
439       cin.get();
440       for(c=0;c<amostra_fluido_vec_aux.size();c++)//corre amostras
441       {
442         if(amostra_fluido_vec_aux[c].base==nome)
443         {
444           amostra_fluido.push_back(amostra_fluido_vec_aux[c]);
445         }
446       }
447
448       break;
449
450     case 4:
451       cout<<"Informe o pH minimo: ";
452
453
454
455       cin>>nome;
456       cin.get();
457
458       if(nome=="-")
459       {
460         dado_double=999999;
461       }
462       else
463       {
464         if(atof(nome.c_str()))
```

```
465         {
466           dado_double=atof(nome.c_str());
467         }
468         else
469         {
470           dado_double=999999;
471         }
472       }
473 
474 
475       for(c=0;c<amostra_fluido_vec_aux.size();c++)//corre amostras
476       {
477         if(amostra_fluido_vec_aux[c].pH_min==dado_double)
478         {
479           amostra_fluido.push_back(amostra_fluido_vec_aux[c]);
480         }
481       }
482 
483       break;
484 
485     case 5:
486       cout<<"Informe␣o␣pH␣maximo:␣";
487       cin>>nome;
488       cin.get();
489 
490       if(nome=="-")
491       {
492         dado_double=999999;
493       }
494       else
495       {
496         if(atof(nome.c_str()))
497         {
498           dado_double=atof(nome.c_str());
499         }
500         else
501         {
502           dado_double=999999;
503         }
504       }
505 
506       for(c=0;c<amostra_fluido_vec_aux.size();c++)//corre amostras
507       {
508         if(amostra_fluido_vec_aux[c].pH_max==dado_double)
509         {
510           amostra_fluido.push_back(amostra_fluido_vec_aux[c]);
511         }
512       }
```

```
513
514       break;
515
516
517
518     case 6:
519       cout<<"Informe␣o␣Teor␣de␣agua:␣";
520       cin>>nome;
521       cin.get();
522
523       if(nome=="-")
524       {
525         dado_double=999999;
526       }
527       else
528       {
529         if(atof(nome.c_str()))
530         {
531           dado_double=atof(nome.c_str());
532         }
533         else
534         {
535           dado_double=999999;
536         }
537       };
538       for(c=0;c<amostra_fluido_vec_aux.size();c++)//corre amostras
539       {
540         if(amostra_fluido_vec_aux[c].teoragua==dado_double)
541         {
542           amostra_fluido.push_back(amostra_fluido_vec_aux[c]);
543         }
544       }
545
546       break;
547
548     case 7:
549       cout<<"Informe␣o␣peso␣especifico␣minimo:␣";
550        cin>>nome;
551       cin.get();
552
553
554       if(nome=="-")
555       {
556         dado_double=999999;
557       }
558       else
559       {
560         if(atof(nome.c_str()))
```

```
561         {
562           dado_double=atof(nome.c_str());
563         }
564         else
565         {
566           dado_double=999999;
567         }
568       }
569 
570 
571       for(c=0;c<amostra_fluido_vec_aux.size();c++)//corre amostras
572       {
573         if(amostra_fluido_vec_aux[c].pe_min==dado_double)
574         {
575           amostra_fluido.push_back(amostra_fluido_vec_aux[c]);
576         }
577       }
578 
579       break;
580 
581 
582 
583     case 8:
584       cout<<"Informe␣o␣peso␣especifico␣maximo:␣";
585       cin>>nome;
586       cin.get();
587 
588       if(nome=="-")
589       {
590         dado_double=999999;
591       }
592       else
593       {
594         if(atof(nome.c_str()))
595         {
596           dado_double=atof(nome.c_str());
597         }
598         else
599         {
600           dado_double=999999;
601         }
602       }
603 
604 
605       for(c=0;c<amostra_fluido_vec_aux.size();c++)//corre amostras
606       {
607         if(amostra_fluido_vec_aux[c].pe_max==dado_double)
608         {
```

```
609           amostra_fluido.push_back(amostra_fluido_vec_aux[c]);
610         }
611       }
612
613       break;
614
615
616
617
618     case 9:
619       cout<<"Informe a Temperatura de envelhecimento: ";
620       cin>>nome;
621       cin.get();
622
623       if(nome=="-")
624       {
625         dado_double=999999;
626       }
627       else
628       {
629         if(atof(nome.c_str()))
630         {
631           dado_double=atof(nome.c_str());
632         }
633         else
634         {
635           dado_double=999999;
636         }
637       }
638       for(c=0;c<amostra_fluido_vec_aux.size();c++)//corre amostras
639       {
640         if(amostra_fluido_vec_aux[c].temp_e==dado_double)
641         {
642           amostra_fluido.push_back(amostra_fluido_vec_aux[c]);
643         }
644       }
645
646       break;
647
648     case 10:
649       cout<<"Informe a Salinidade: ";
650       cin>>nome;
651       cin.get();
652
653       if(nome=="-")
654       {
655         dado_double=999999;
656       }
```

```
657       else
658       {
659         if(atof(nome.c_str()))
660         {
661           dado_double=atof(nome.c_str());
662         }
663         else
664         {
665           dado_double=999999;
666         }
667       }
668       for(c=0;c<amostra_fluido_vec_aux.size();c++)//corre amostras
669       {
670         if(amostra_fluido_vec_aux[c].salinidade==dado_double)
671         {
672           amostra_fluido.push_back(amostra_fluido_vec_aux[c]);
673         }
674       }
675
676       break;
677
678     case 11:
679       cout<<"Informe␣o␣volume␣de␣filtrado:␣";
680       cin>>nome;
681       cin.get();
682
683       if(nome=="-")
684       {
685         dado_double=999999;
686       }
687       else
688       {
689         if(atof(nome.c_str()))
690         {
691           dado_double=atof(nome.c_str());
692         }
693         else
694         {
695           dado_double=999999;
696         }
697       }
698       for(c=0;c<amostra_fluido_vec_aux.size();c++)//corre amostras
699       {
700         if(amostra_fluido_vec_aux[c].filtrado==dado_double)
701         {
702           amostra_fluido.push_back(amostra_fluido_vec_aux[c]);
703         }
704       }
```

```
705
706       break;
707
708     case 12:
709       cout<<"Informe o Dia: ";
710       cin>>dia;
711       cin.get();
712       for(c=0;c<amostra_fluido_vec_aux.size();c++)//corre amostras
713       {
714         if(amostra_fluido_vec_aux[c].dia==dia)
715         {
716           amostra_fluido.push_back(amostra_fluido_vec_aux[c]);
717         }
718       }
719
720       break;
721
722     case 13:
723       cout<<"Informe o Mes: ";
724       cin>>mes;
725       cin.get();
726       for(c=0;c<amostra_fluido_vec_aux.size();c++)//corre amostras
727       {
728         if(amostra_fluido_vec_aux[c].mes==mes)
729         {
730           amostra_fluido.push_back(amostra_fluido_vec_aux[c]);
731         }
732       }
733
734       break;
735
736
737     case 14:
738       cout<<"Informe o ano: ";
739       cin>>ano;
740       cin.get();
741       for(c=0;c<amostra_fluido_vec_aux.size();c++)//corre amostras
742       {
743         if(amostra_fluido_vec_aux[c].ano==ano)
744         {
745           amostra_fluido.push_back(amostra_fluido_vec_aux[c]);
746         }
747       }
748
749       break;
750
751     case 15: // listar todos
752       amostra_fluido=amostra_fluido_vec_aux;
```

```
      break;

    default:
      cout<<"Filtro␣invalido."<<endl;
      //verifica sistema operacional
      #ifdef __linux__
      //sleep(1000);
      #elif _WIN32
      Sleep(1000);
      #else

      #endif
      resposta=1;
      break;
  }


  return amostra_fluido;

}

void CBancodeDados::exibir_amostra(int chave_)
{

  int_chave=chave_;
  CFluidodePerfuracao amostra_aux;
  vector<CFluidodePerfuracao> v_amostra;
  unsigned int c;
  //lendo todas as amostras basico
  v_amostra=ler_amostra_basico();
  //procurando amostra correspontende a chave informada
  for(c=0;c<v_amostra.size();c++)
  {
    if (int_chave==v_amostra[c].chave)
    {string teste_vazio(double);
      double teste_vazio(string);
      amostra_aux=v_amostra[c];
      break;
    }
  }
  //inicia das exibicoes basica das amostras
  cout<<"_________________________________________________"<<endl;
  cout<<"␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣"<<endl;
  cout<<"␣␣␣␣␣␣Exibindo␣Amostra␣de␣Chave␣"<<int_chave<<"␣␣␣␣␣"<<endl;
  cout<<"_________________________________________________"<<endl;
  cout<<endl<<endl;
  cout<<"Data␣de␣Cadastro␣----␣:␣"<<amostra_aux.dia<<"/"<<amostra_aux.
     mes<<"/"<<amostra_aux.ano<<endl;
```

```
800 cout<<"Nome␣do␣Fluido␣-------:␣"<<amostra_aux.nome_do_fluido<<endl;
801 cout<<"Base␣----------------␣:␣"<<amostra_aux.base<<endl;
802 cout<<"Teor␣de␣Base␣--------␣:␣"<<teste_vazio(amostra_aux.teorbase)<<
    endl;
803 cout<<"Teor␣de␣Agua␣--------␣:␣"<<teste_vazio(amostra_aux.teoragua)<<
    endl<<endl;
804 cout<<"␣␣␣␣-----Propriedades␣Fisicas␣e␣Quimicas-----␣␣␣␣␣␣"<<endl<<
    endl;
805 cout<<"pH␣Minimo␣-----------␣:␣"<<teste_vazio(amostra_aux.pH_min)<<
    endl;
806 cout<<"pH␣Maximo␣-----------␣:␣"<<teste_vazio(amostra_aux.pH_max)<<
    endl;
807 cout<<"Peso␣Especifico␣Minimo:␣"<<teste_vazio(amostra_aux.pe_min)<<
    endl;
808 cout<<"Peso␣Especifico␣Maximo:␣"<<teste_vazio(amostra_aux.pe_max)<<
    endl;
809 cout<<"Temperatura␣de␣envelhecimento:␣"<<teste_vazio(amostra_aux.
    temp_e)<<endl;
810 cout<<"Forca␣gel␣inicial␣antes␣do␣envelhecimento:␣"<<teste_vazio(
    amostra_aux.gel_i_ae)<<endl;
811 cout<<"Forca␣gel␣inicial␣depois␣do␣envelhecimento:␣"<<teste_vazio(
    amostra_aux.gel_i_de)<<endl;
812 cout<<"Forca␣gel␣final␣antes␣do␣envelhecimento:␣"<<teste_vazio(
    amostra_aux.gel_f_ae)<<endl;
813 cout<<"Forca␣gel␣final␣depois␣do␣envelhecimento:␣"<<teste_vazio(
    amostra_aux.gel_f_de)<<endl;
814 cout<<"Estabilidade␣eletrica␣antes␣do␣envelhecimento:␣"<<teste_vazio(
    amostra_aux.est_el_ae)<<endl;
815 cout<<"Estabilidade␣eletrica␣depois␣do␣envelhecimento:␣"<<teste_vazio(
    amostra_aux.est_el_de)<<endl;
816 cout<<"Coeficiente␣de␣lubricidade:␣"<<teste_vazio(amostra_aux.
    c_lubricidade)<<endl;
817 cout<<"Volume␣de␣filtrado:␣"<<teste_vazio(amostra_aux.filtrado)<<endl;
818 cout<<"Teor␣de␣solidos␣----------␣:␣"<<teste_vazio(amostra_aux.
    teor_solidos)<<endl;
819 cout<<"Salinidade␣----------␣:␣"<<teste_vazio(amostra_aux.temp_e)<<
    endl<<endl;
820 cout<<"␣␣␣␣----Propriedades␣Reologicas----␣␣␣␣␣␣"<<endl<<endl;
821 cout<<"Viscosidade␣Aparente␣antes␣do␣envelhecimento:␣"<<teste_vazio(
    amostra_aux.VA_ae)<<endl;
822 cout<<"Viscosidade␣Aparente␣depois␣do␣envelhecimento:␣"<<teste_vazio(
    amostra_aux.VA_de)<<endl;
823 cout<<"Viscosidade␣Plastica␣antes␣do␣envelhecimento:␣"<<teste_vazio(
    amostra_aux.VP_ae)<<endl;
824 cout<<"Viscosidade␣Plastica␣depois␣do␣envelhecimento:␣"<<teste_vazio(
    amostra_aux.VP_de)<<endl<<endl;
825 cout<<"␣␣␣␣----Aditivos----␣␣␣␣␣␣"<<endl<<endl;
826 cout<<"Adensante␣----------␣:␣"<<amostra_aux.adensante<<endl;
```

```
827   cout<<"Concentracao␣de␣Adensante␣----------␣:␣"<<teste_vazio(
          amostra_aux.conc_adensante)<<endl;
828   cout<<"Inibidor␣de␣Formacoes␣Ativas␣-------␣:␣"<<amostra_aux.
          inibidor_fa<<endl;
829   cout<<"Concentracao␣do␣Inibidor␣de␣Formacoes␣Ativas␣----------␣:␣"<<
          teste_vazio(amostra_aux.conc_inibidor_fa)<<endl;
830   cout<<"Redutor␣de␣Filtrado␣␣␣:␣"<<amostra_aux.redutor_f<<endl;
831   cout<<"Concentracao␣do␣Redutor␣de␣Filtrado␣----------␣:␣"<<teste_vazio
          (amostra_aux.conc_redutor_f)<<endl;
832   cout<<"Biopolimero␣␣␣:␣"<<amostra_aux.biopolimero<<endl;
833   cout<<"Concentracao␣do␣Biopolimero␣----------␣:␣"<<teste_vazio(
          amostra_aux.conc_biopolimero)<<endl;
834   cout<<"Viscosificante␣␣␣:␣"<<amostra_aux.viscosificante<<endl;
835   cout<<"Concentracao␣do␣Viscosificante----------␣:␣"<<teste_vazio(
          amostra_aux.conc_viscosificante)<<endl;
836   cout<<"Dispersante␣␣␣:␣"<<amostra_aux.dispersante<<endl;
837   cout<<"Concentracao␣do␣Dispersante----------␣:␣"<<teste_vazio(
          amostra_aux.conc_dispersante)<<endl;
838   cout<<"Defloculante␣␣␣:␣"<<amostra_aux.defloculante<<endl;
839   cout<<"Concentracao␣do␣Defloculante----------␣:␣"<<teste_vazio(
          amostra_aux.conc_defloculante)<<endl;
840   cout<<"Emulsificante␣␣␣:␣"<<amostra_aux.emulsificante<<endl;
841   cout<<"Concentracao␣do␣Emulsificante----------␣:␣"<<teste_vazio(
          amostra_aux.conc_emulsificante)<<endl;
842   cout<<"Biocida␣␣␣:␣"<<amostra_aux.biocida<<endl;
843   cout<<"Concentracao␣do␣Biocida----------␣:␣"<<teste_vazio(amostra_aux.
          conc_biocida)<<endl;
844   cout<<"Lubrificante␣␣␣:␣"<<amostra_aux.lubrificante<<endl;
845   cout<<"Concentracao␣do␣Lubrificante----------␣:␣"<<teste_vazio(
          amostra_aux.conc_lubrificante)<<endl;
846   cout<<"Inibidor␣de␣corrosao␣␣␣:␣"<<amostra_aux.inibidor_c<<endl;
847   cout<<"Concentracao␣do␣Inibidor␣de␣Corrosao----------␣:␣"<<teste_vazio
          (amostra_aux.conc_inibidor_c)<<endl;
848   cout<<"Controlador␣de␣pH␣␣␣:␣"<<amostra_aux.controlador_pH<<endl;
849   cout<<"Concentracao␣do␣Controlador␣de␣pH----------␣:␣"<<teste_vazio(
          amostra_aux.conc_controlador_pH)<<endl;
850   cout<<endl<<endl;
851 }
852
853 void CBancodeDados::listar_amostra(vector<CFluidodePerfuracao>
        amostra_fluido_)//funcao listar amostras
854 {
855
856   i=0;//inteiro para contagem no for
857
858   amostra_fluido=amostra_fluido_;
859   ////////////////////////////////////////////
860   //Inicia a listagem das amostras lidas no disco
```

```
861  ////////////////////////////////////////////
862  //escreve linha superior ao titulo dos atributos das amostras
863  for(i=0;i<90;i++)
864  {
865
866    #ifdef __linux
867    cout<<"\u2500";
868    #elif _WIN32
869    cout<<(char)196;
870    #else
871
872    #endif
873
874  }
875  cout<<endl;//finaliza linha
876  //escreve titulo dos atributos das amostras
877  cout.setf(ios::left);
878  cout.width(7);
879  cout<<"CHAVE";
880
881  cout.setf(ios::left);
882  cout.width(20);
883  cout<<"NOME␣DO␣FLUIDO";
884
885  cout.setf(ios::left);
886  cout.width(10);
887  cout<<"BASE";
888
889  cout.setf(ios::left);
890  cout.width(20);
891  cout<<"TEOR␣DE␣BASE";
892
893  cout.setf(ios::left);
894  cout.width(20);
895  cout<<"TEOR␣DE␣AGUA";
896
897  cout.setf(ios::left);
898  cout.width(20);
899  cout<<"DATA";
900
901  cout<<endl;//finaliza titulo dos itens
902
903  //escreve a linha abaixo do titulo das tabelas
904  for(i=0;i<90;i++)
905  {
906    #ifdef __linux
907    cout<<"\u2500";
908    #elif _WIN32
```

```
909     cout<<(char)196;
910     #else
911
912     #endif
913   }
914
915   //escreve os atributos das amostras na tela
916
917   cout<<endl;//pula linha para primeira listagem
918   for(i=0;i<amostra_fluido.size();i++)//comeca listar da 1, amostra zero
          é a base
919   {
920
921     if(amostra_fluido[i].chave!=0)
922     {
923
924       cout.setf(ios::left);
925       cout.width(7);
926       cout<<amostra_fluido[i].chave;//escreve as chaves
927
928       cout.setf(ios::left);
929       cout.width(20);
930       cout<<amostra_fluido[i].nome_do_fluido;//escreve o nome do fluido
931
932       cout.setf(ios::left);
933       cout.width(10);
934       cout<<amostra_fluido[i].base;//escreve a base
935
936       cout.setf(ios::left);
937       cout.width(20);
938       cout<<teste_vazio(amostra_fluido[i].teorbase);//escreve o teor de
             base
939
940       cout.setf(ios::left);
941       cout.width(20);
942       cout<<teste_vazio(amostra_fluido[i].teoragua);//escreve o teor de
             agua
943
944       cout.setf(ios::left);
945       cout.width(20);
946       if(amostra_fluido[i].dia<10)ss<<0;
947                           ss<<amostra_fluido[i].dia<<"/";
948       if(amostra_fluido[i].mes<10)ss<<0;
949                           ss<<amostra_fluido[i].mes<<"/";
950       if(amostra_fluido[i].ano<10)ss<<0;
951                           ss<<amostra_fluido[i].ano;//constroi data
952                           cout<<ss.str();
953       ss.str("");
```

```
954
955         cout<<endl;//pula linha antes das pausas
956
957       }
958     }
959
960     i=0;
961     cout<<endl;
962
963     //escreve linha inferior
964     for(i=0;i<90;i++)
965     {
966       #ifdef __linux
967       cout<<"\u2500";
968       #elif _WIN32
969       cout<<(char)196;
970       #else
971
972       #endif
973     }
974     cout<<endl;
975     chave.clear();
976 }
977
978 void CBancodeDados::listar_amostra(vector<CFluidodePerfuracao>
      amostra_fluido_, int filtro_)//funcao listar amostras
979 {
980
981     i=0;
982     filtro=filtro_;
983     amostra_fluido=amostra_fluido_;
984     ////////////////////////////////////////////
985     //Inicia a listagem das amostras lidas no disco
986     ////////////////////////////////////////////
987     //escreve linha superior ao titulo dos atributos das amostras
988     for(i=0;i<80;i++)
989     {
990
991       #ifdef __linux
992       cout<<"\u2500";
993       #elif _WIN32
994       cout<<(char)196;
995       #else
996
997       #endif
998
999     }
1000    cout<<endl;//finaliza linha
```

```
1001  //escreve titulo dos atributos das amostras
1002  cout.setf(ios::left);
1003  cout.width(7);
1004  cout<<"CHAVE";
1005
1006  cout.setf(ios::left);
1007  cout.width(20);
1008  cout<<"NOME␣DO␣FLUIDO";
1009
1010  cout.setf(ios::left);
1011  cout.width(10);
1012  cout<<"DATA";
1013
1014  switch(filtro)
1015  {
1016
1017    case 3:
1018      cout.setf(ios::left);
1019      cout.width(7);
1020      cout<<"Base";
1021      break;
1022
1023    case 4:
1024      cout.setf(ios::left);
1025      cout.width(10);
1026      cout<<"pH␣minimo";
1027      break;
1028
1029    case 5:
1030      cout.setf(ios::left);
1031      cout.width(10);
1032      cout<<"pH␣maximo";
1033      break;
1034
1035    case 6:
1036      cout.setf(ios::left);
1037      cout.width(10);
1038      cout<<"Teor␣de␣agua";
1039      break;
1040
1041    case 7:
1042      cout.setf(ios::left);
1043      cout.width(10);
1044      cout<<"Peso␣especifico␣minimo";
1045      break;
1046
1047    case 8:
1048      cout.setf(ios::left);
```

```
1049        cout.width(10);
1050        cout<<"Peso␣especifico␣maximo";
1051        break;
1052
1053      case 9:
1054        cout.setf(ios::left);
1055        cout.width(20);
1056        cout<<"Temperatura␣de␣envelhecimento";
1057        break;
1058
1059      case 10:
1060        cout.setf(ios::left);
1061        cout.width(10);
1062        cout<<"Salinidade";
1063        break;
1064
1065      case 11:
1066        cout.setf(ios::left);
1067        cout.width(10);
1068        cout<<"Filtrado";
1069        break;
1070
1071      case 14:
1072        cout.setf(ios::left);
1073        cout.width(7);
1074        cout<<"Ano";
1075        break;
1076
1077    }
1078    cout<<endl;//finaliza titulo dos itens
1079
1080    //escreve a linha abaixo do titulo das tabelas
1081    for(i=0;i<80;i++)
1082    {
1083      #ifdef __linux
1084      cout<<"\u2500";
1085      #elif _WIN32
1086      cout<<(char)196;
1087      #else
1088
1089      #endif
1090    }
1091
1092    //escreve os atributos das amostras na tela
1093    for(i=0;i<amostra_fluido.size();i++)
1094    {
1095      if(amostra_fluido[i].chave!=0) //nao lista amostra de chave zero
1096      {
```

```
1097
1098
1099        cout<<endl;
1100
1101        cout.setf(ios::left);
1102        cout.width(7);
1103        cout<<amostra_fluido[i].chave;//escreve as chaves
1104
1105        cout.setf(ios::left);
1106        cout.width(20);
1107        cout<<amostra_fluido[i].nome_do_fluido;//escreve o nome do fluido
1108
1109        cout.setf(ios::left);
1110        cout.width(20);
1111        cout<<amostra_fluido[i].base;//escreve a base
1112
1113        cout.setf(ios::left);
1114        cout.width(7);
1115        cout<<teste_vazio(amostra_fluido[i].teorbase);//escreve o teor de
              base
1116
1117        cout.setf(ios::left);
1118        cout.width(7);
1119        cout<<teste_vazio(amostra_fluido[i].teoragua);//escreve o teor de
              agua
1120
1121        cout.setf(ios::left);
1122        cout.width(10);
1123        if(amostra_fluido[i].dia<10)ss<<0;
1124                            ss<<amostra_fluido[i].dia<<"/";
1125        if(amostra_fluido[i].mes<10)ss<<0;
1126                            ss<<amostra_fluido[i].mes<<"/";
1127        if(amostra_fluido[i].ano<10)ss<<0;
1128                            ss<<amostra_fluido[i].ano;//constroi data
1129                            cout<<ss.str();
1130        ss.str("");
1131
1132        switch(filtro)
1133        {
1134
1135          case 3:
1136            cout.setf(ios::left);
1137            cout.width(7);
1138            cout<<teste_vazio(amostra_fluido[i].base);
1139            break;
1140
1141          case 4:
1142            cout.setf(ios::left);
```

```
1143            cout.width(7);
1144            cout<<teste_vazio(amostra_fluido[i].pH_min);
1145            break;
1146
1147          case 5:
1148            cout.setf(ios::left);
1149            cout.width(7);
1150            cout<<teste_vazio(amostra_fluido[i].pH_max);
1151            break;
1152
1153          case 6:
1154            cout<<teste_vazio(amostra_fluido[i].teoragua);
1155            break;
1156
1157          case 7:
1158            cout.setf(ios::left);
1159            cout.width(7);
1160            cout<<teste_vazio(amostra_fluido[i].pe_min);
1161            break;
1162
1163          case 8:
1164            cout.setf(ios::left);
1165            cout.width(7);
1166            cout<<teste_vazio(amostra_fluido[i].pe_max);
1167            break;
1168
1169          case 9:
1170            cout.setf(ios::left);
1171            cout.width(7);
1172            cout<<teste_vazio(amostra_fluido[i].temp_e);
1173            break;
1174
1175          case 10:
1176            cout.setf(ios::left);
1177            cout.width(7);
1178            cout<<teste_vazio(amostra_fluido[i].salinidade);
1179            break;
1180
1181          case 11:
1182            cout.setf(ios::left);
1183            cout.width(7);
1184            cout<<teste_vazio(amostra_fluido[i].filtrado);
1185            break;
1186
1187          case 14:
1188            cout.setf(ios::left);
1189            cout.width(7);
1190            cout<<teste_vazio(amostra_fluido[i].ano);
```

```
1191            break;
1192
1193          }
1194
1195      }
1196
1197
1198      }
1199      i=0;
1200      cout<<endl;
1201
1202      //escreve linha inferior
1203      for(i=0;i<80;i++)
1204      {
1205        #ifdef __linux
1206        cout<<"\u2500";
1207        #elif _WIN32
1208        cout<<(char)196;
1209        #else
1210
1211        #endif
1212      }
1213      cout<<endl;
1214      chave.clear();
1215  }
1216
1217  void CBancodeDados::excluir_amostra(int int_chave_)
1218  {
1219    ////////////////////////////////////
1220    //Inicio da exclusao chave
1221    ////////////////////////////////////
1222    vector<int> chave;//vetor que armazena as chaves
1223
1224    int_chave=int_chave_;//inteiro que recebe chave informada
1225
1226    //abre o arquivo de informacoes do bd
1227    #ifdef __linux
1228    caminho_bd<<"bd"<<char(47)<<"ini"<<char(47)<<"bdinfo";
1229    #elif _WIN32
1230    caminho_bd<<"bd"<<char(92)<<"ini"<<char(92)<<"bdinfo";
1231    #else
1232
1233    #endif
1234    fin.open(caminho_bd.str().c_str());
1235    caminho_bd.str("");
1236    //carrega o arquivo de informacoes do bd
1237    while(! fin.eof())
1238    {
```

```
1239        fin >> int_chave_aux;
1240        chave.push_back(int_chave_aux);
1241      }
1242      fin.close();
1243      //inicia pesquisa da chave a ser deletada
1244      for(i=0;i<=chave.size();i++)
1245      {
1246        if (chave[i]==int_chave)
1247        {
1248          chave.erase(chave.begin() + i);//deleta a chave encontrada
1249        }
1250      }
1251
1252      //reescreve vetor no arquivo bdinfo
1253      ofstream fout;
1254      #ifdef __linux
1255      caminho_bd<<"bd"<<char(47)<<"ini"<<char(47)<<"bdinfo";
1256      #elif _WIN32
1257      caminho_bd<<"bd"<<char(92)<<"ini"<<char(92)<<"bdinfo";
1258      #else
1259
1260      #endif
1261      fout.open(caminho_bd.str().c_str());//abre bdinfo para escrever
          informacoes atualizadas
1262      caminho_bd.str("");
1263      //testa se nao abriu o arquivo
1264      if(fout.fail())
1265      {
1266        cout<<"Nao␣foi␣possivel␣reescrever␣a␣chave."<<endl;
1267      }
1268      for(i=0;i<(chave.size()-1);i++)//escreve ate o punultimo item do
          vetor chave pulando linha
1269      {
1270        fout<<chave[i]<<endl;
1271      }
1272      //para nÃfo pular linha ao fim do arquivo, escreve o ultimo item do
          vetor chave
1273      fout<<chave[i];
1274      fout.close();//fecha o arquivo de informacoes do banco de dados
1275      i=0;
1276      ////////////////////////////////////
1277      //Fim da exclusao chave
1278      ////////////////////////////////////
1279      ////////////////////////////////////
1280      //Inicio da remocao da amostra para \lixo
1281      ////////////////////////////////////
1282      stringstream comando_remover;//Declaracao das variaveis do comando
          de remover para lixeira
```

```
1283    //criar o comando a ser executado
1284    //comando: "move bd\0.txt bd\lixo" Comando para mover o arquivo da
            amostra
1285    comando_remover.str("");//apagand stringstream
1286    //verifica sistema operacional
1287    #ifdef __linux
1288    //mv ./bd/0.txt ./bd/lixo
1289    comando_remover<<"mv"<<"␣"<<"./bd/"<<int_chave<<"␣"<<"./bd/lixo";
1290    #elif _WIN32
1291    comando_remover<<"move"<<"␣"<<"bd"<<char(92)<<int_chave<<"␣"<<"bd"<<
            char(92)<<"lixo";
1292
1293    #else
1294
1295    #endif
1296    system(comando_remover.str().c_str());//Manda o comando pro prompt
            para mover o arquivo p lixeira
1297    comando_remover.str("");//apagand stringstream
1298
1299    }
1300
1301  string CBancodeDados::teste_vazio(double num_double_)
1302  {
1303    num_double_ss.str("");
1304    num_double=num_double_;
1305    const_vazio_num_double=999999;
1306    if(num_double==const_vazio_num_double)
1307    {
1308      return("-");
1309    }
1310    else
1311    {
1312      num_double_ss<<num_double;//converte o double para string
1313      return(num_double_ss.str());
1314    }
1315  }
1316
1317  double CBancodeDados::teste_vazio(string vazio_str_)
1318  {
1319    double dado_double;
1320    vazio_str=vazio_str_;
1321    const_vazio_str="-";
1322    if(vazio_str==const_vazio_str)
1323    {
1324      return 999999;
1325    }
1326    else
1327    {
```

```
1328        return dado_double=atof(vazio_str.c_str());//converte a string que
                entrou para double
1329      }
1330    }
1331
1332    void CBancodeDados::inserir_amostra()
1333    {
1334      //Inclusao de chave no arquivo bdinfo
1335
1336      vector<int> chave;//vetor que armazena as chaves
1337      ifstream fin;//objeto de leitura
1338      int int_chave;//inteiro que recebe cada linha lida no arquivo
1339
1340      #ifdef __linux
1341      caminho_bd<<"bd"<<char(47)<<"ini"<<char(47)<<"bdinfo";
1342      #elif _WIN32
1343      caminho_bd<<"bd"<<char(92)<<"ini"<<char(92)<<"bdinfo";
1344      #else
1345
1346      #endif
1347
1348      fin.open(caminho_bd.str().c_str());//abre o arquivo de informacoes
             do bd usando metodos que retorna uma string de um numero e
             converte a string C++ para para padrao C
1349      caminho_bd.str("");
1350      while(! fin.eof())//carrega o arquivo de informacoes do bd(chaves) e
             realiza leitura até o final do arquivo
1351      {
1352        fin >> int_chave;
1353        chave.push_back(int_chave);
1354      }
1355      fin.close();
1356
1357      //Escrita da nova chave
1358      #ifdef __linux
1359      caminho_bd<<"bd"<<char(47)<<"ini"<<char(47)<<"bdinfo";
1360      #elif _WIN32
1361      caminho_bd<<"bd"<<char(92)<<"ini"<<char(92)<<"bdinfo";
1362      #else
1363
1364      #endif
1365
1366      ofstream fout;
1367      fout.open(caminho_bd.str().c_str(),ios::app);//abre bdinfo e vai
             para o final do arquivo
1368      caminho_bd.str("");
1369      //testa se nao abriu o arquivo
1370      if(fout.fail())
```

```
1371    {
1372      cout<<"Nao␣foi␣possivel␣inserir␣amostra."<<endl;
1373    }
1374
1375    int_chave++;//soma ultima chave mais um
1376    fout<<endl;//pula uma linha para inserir a chave
1377    fout<<int_chave;//escreve a chave posterior no fim do arquivo
1378
1379    fout.close();
1380
1381    //Fim da inclusao da chave
1382
1383    //Inclusao do novo arquivo para novo fluido
1384
1385    //cria diretorio
1386    #ifdef __linux
1387    comando_caminho_amostra<<"mkdir␣"<<"bd"<<char(47)<<int_chave;//
           comando mkdir cria diretorio
1388    #elif _WIN32
1389    comando_caminho_amostra<<"mkdir␣"<<"bd"<<char(92)<<int_chave;
1390    #else
1391
1392    #endif
1393
1394    system(comando_caminho_amostra.str().c_str());//cria o diretorio a
           ser gravado os arquivos
1395    comando_caminho_amostra.str("");//apaga stringstream
1396
1397    //Formar nome do arquivo
1398
1399    #ifdef __linux
1400    nome_amostra<<"bd"<<char(47)<<int_chave<<char(47)<<int_chave;
1401    #elif _WIN32
1402    nome_amostra<<"bd"<<char(92)<<int_chave<<char(92)<<int_chave;
1403    #else
1404
1405    #endif
1406
1407    //Escrita no arquivo do novo fluido
1408
1409    fout.open(nome_amostra.str().c_str());//abre o arquivo
1410    nome_amostra.str("");
1411
1412    string nome;
1413    float num;
1414
1415    cout<<"---------------------------------"<<endl;
1416    cout<<"Preencha␣com␣os␣dados␣da␣amostra:␣"<<endl;
```

```
1417    cout<<"Para␣dado␣numerico␣nulo,␣informe␣"<<endl;
1418    cout<<"apenas␣um␣-␣␣␣␣␣␣␣␣␣·␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣"<<endl;
1419    cout<<"----------------------------------"<<endl;
1420    fout<<"-chave:"<<endl;//escreve o titulo chave
1421    fout<<int_chave<<endl;//escreve a chave
1422
1423    fout<<"-data:␣"<<endl;//escreve a string -data:
1424    fout<<"--dia:␣"<<endl;//escreve a string --dia:
1425    cout<<"Data␣de␣Cadastro␣da␣Amostra:␣"<<endl;;
1426    repetedia:
1427
1428    cout<<"Dia:␣";
1429    while(!(cin>>num_int))//correcao de erro caso entre alguma resposta
          errada
1430    {
1431      cin.clear();
1432      cin>>ch;
1433      cout<<endl<<"Resposta␣Incorreta!"<<endl;
1434      cin.get();
1435      goto repetedia;//nao aceita dia maior que 31
1436    }
1437
1438    if (num_int>31)
1439    {
1440      cout<<endl<<"Resposta␣Incorreta!"<<endl;
1441      cout<<"Dia␣maior␣que␣31!"<<endl;
1442      goto repetedia;//nao aceita dia maior que 31
1443    }
1444    fout<<num_int<<endl;
1445
1446    fout<<"--mes:␣"<<endl;//escreve a string --dia:
1447    repetemes:
1448    cout<<"Mes:␣";
1449    while(!(cin>>num_int))//correcao de erro caso entre alguma resposta
          errada
1450    {
1451      cin.clear();
1452      cin>>ch;
1453      cin.get();
1454      cout<<endl<<"Resposta␣Incorreta!"<<endl;
1455      goto repetemes;//nao aceita dia maior que 31
1456    }
1457
1458    if (num_int>12)
1459    {
1460      cout<<endl<<"Resposta␣Incorreta!"<<endl;
1461      cout<<"Mes␣maior␣que␣12!"<<endl;
1462      goto repetemes;//nao aceita dia maior que 31
```

```
1463    }
1464    fout<<num_int<<endl;
1465
1466    fout<<"--ano:␣"<<endl;//escreve a string --dia:
1467    cout<<"Ano:␣";
1468    while(!(cin>>num_int))//correcao de erro caso entre alguma resposta
            errada
1469    {
1470      cin.clear();
1471      cin>>ch;
1472      cout<<endl<<"Resposta␣Incorreta!"<<endl;
1473    }
1474    cin.get();
1475    fout<<num_int<<endl;
1476
1477    fout<<"-Nome_do_Fluido_de_Perfuracao:␣"<<endl;//escreve o titulo
            bacia
1478    cout<<"Nome␣do␣Fluido␣de␣Perfuracao␣(9␣caracteres):␣";
1479    cin>>nome;
1480    cin.get();
1481    fout<<nome<<endl;
1482
1483    fout<<"-Base:␣"<<endl;//escreve o titulo bacia
1484    cout<<"Base␣:␣";
1485    cin>>nome;
1486    cin.get();
1487    fout<<nome<<endl;
1488
1489    fout<<"-Teor_da_base:␣"<<endl;
1490
1491    cout<<"Teor␣da␣base:␣";
1492    cin>>nome;
1493    cin.get();
1494
1495    if(nome=="-")
1496    {
1497      num=999999;
1498    }
1499    else
1500    {
1501      if(atof(nome.c_str()))
1502      {
1503        num=atof(nome.c_str());
1504      }
1505      else
1506      {
1507        num=999999;
1508      }
```

```
1509    }
1510
1511    fout<<num<<endl;
1512
1513
1514    fout<<"-teor_de_agua:␣"<<endl;
1515    cout<<"Teor␣de␣Agua:␣";
1516    cin>>nome;
1517    cin.get();
1518
1519    if(nome=="-")
1520    {
1521      num=999999;
1522    }
1523    else
1524    {
1525      if(atof(nome.c_str()))
1526      {
1527        num=atof(nome.c_str());
1528      }
1529      else
1530      {
1531        num=999999;
1532      }
1533    }
1534    fout<<num<<endl;
1535
1536    fout<<"
          ------------------------------------------------------------------------
          "<<endl;
1537    fout<<"PROPRIEDADES_FISICAS_E_QUIMICAS"<<endl;
1538    fout<<"-pH_min:␣"<<endl;
1539    cout<<"pH␣minimo:␣␣";
1540    cin>>nome;
1541    cin.get();
1542
1543    if(nome=="-")
1544    {
1545      num=999999;
1546    }
1547    else
1548    {
1549      if(atof(nome.c_str()))
1550      {
1551        num=atof(nome.c_str());
1552      }
1553      else
1554      {
```

```
1555          num=999999;
1556        }
1557      }
1558      fout<<num<<endl;
1559
1560      fout<<"-pH_max:␣"<<endl;
1561      cout<<"pH␣maximo:␣";
1562      cin>>nome;
1563      cin.get();
1564
1565      if(nome=="-")
1566      {
1567        num=999999;
1568      }
1569      else
1570      {
1571        if(atof(nome.c_str()))
1572        {
1573          num=atof(nome.c_str());
1574        }
1575        else
1576        {
1577          num=999999;
1578        }
1579      }
1580      fout<<num<<endl;
1581
1582
1583      fout<<"-Peso_especifico_min:␣"<<endl;
1584      cout<<"Peso␣especifico␣minimo:␣";
1585      cin>>nome;
1586      cin.get();
1587
1588      if(nome=="-")
1589      {
1590        num=999999;
1591      }
1592      else
1593      {
1594        if(atof(nome.c_str()))
1595        {
1596          num=atof(nome.c_str());
1597        }
1598        else
1599        {
1600          num=999999;
1601        }
1602      }
```

```
1603    fout<<num<<endl;
1604
1605    fout<<"-Peso_especifico_max:␣"<<endl;
1606
1607    cout<<"Peso␣especifico␣maximo:␣";
1608    cin>>nome;
1609    cin.get();
1610
1611    if(nome=="-")
1612    {
1613      num=999999;
1614    }
1615    else
1616    {
1617      if(atof(nome.c_str()))
1618      {
1619        num=atof(nome.c_str());
1620      }
1621      else
1622      {
1623        num=999999;
1624      }
1625    }
1626    fout<<num;
1627
1628    fout<<"-Temperatura_envelhecimento:␣"<<endl;
1629
1630    cout<<"Temperatura␣do␣envelhecimento:␣";
1631    cin>>nome;
1632    cin.get();
1633
1634    if(nome=="-")
1635    {
1636      num=999999;
1637    }
1638    else
1639    {
1640      if(atof(nome.c_str()))
1641      {
1642        num=atof(nome.c_str());
1643      }
1644      else
1645      {
1646        num=999999;
1647      }
1648    }
1649    fout<<num<<endl;
1650
```

```
1651    fout<<"-Forca_gel_inicial_antes_do_envelhecimento:␣"<<endl;
1652
1653    cout<<"Forca␣gel␣inicial␣antes␣do␣envelhecimento:␣";
1654    cin>>nome;
1655    cin.get();
1656
1657    if(nome=="-")
1658    {
1659      num=999999;
1660    }
1661    else
1662    {
1663      if(atof(nome.c_str()))
1664      {
1665        num=atof(nome.c_str());
1666      }
1667      else
1668      {
1669        num=999999;
1670      }
1671    }
1672    fout<<num<<endl;
1673
1674    fout<<"-Forca_gel_final_antes_do_envelhecimento:␣"<<endl;
1675
1676    cout<<"Forca␣gel␣final␣antes␣do␣envelhecimento:␣";
1677    cin>>nome;
1678    cin.get();
1679
1680    if(nome=="-")
1681    {
1682      num=999999;
1683    }
1684    else
1685    {
1686      if(atof(nome.c_str()))
1687      {
1688        num=atof(nome.c_str());
1689      }
1690      else
1691      {
1692        num=999999;
1693      }
1694    }
1695    fout<<num<<endl;
1696
1697    fout<<"-Forca_gel_inicial_depois_do_envelhecimento:␣"<<endl;
1698
```

```
1699    cout<<"Forca␣gel␣inicial␣depois␣do␣envelhecimento:␣";
1700    cin>>nome;
1701    cin.get();
1702
1703    if(nome=="-")
1704    {
1705      num=999999;
1706    }
1707    else
1708    {
1709      if(atof(nome.c_str()))
1710      {
1711        num=atof(nome.c_str());
1712      }
1713      else
1714      {
1715        num=999999;
1716      }
1717    }
1718    fout<<num<<endl;
1719
1720    fout<<"-Forca_gel_final_depois_do_envelhecimento:␣"<<endl;
1721
1722    cout<<"Forca␣gel␣final␣depois␣do␣envelhecimento:␣";
1723    cin>>nome;
1724    cin.get();
1725
1726    if(nome=="-")
1727    {
1728      num=999999;
1729    }
1730    else
1731    {
1732      if(atof(nome.c_str()))
1733      {
1734        num=atof(nome.c_str());
1735      }
1736      else
1737      {
1738        num=999999;
1739      }
1740    }
1741    fout<<num<<endl;
1742
1743
1744    fout<<"-Estabilidade_eletrica_antes_do_envelhecimento:␣"<<endl;
1745
1746    cout<<"Estabilidade␣eletrica␣antes␣do␣envelhecimento:␣";
```

```
1747    cin>>nome;
1748    cin.get();
1749
1750    if(nome=="-")
1751    {
1752      num=999999;
1753    }
1754    else
1755    {
1756      if(atof(nome.c_str()))
1757      {
1758        num=atof(nome.c_str());
1759      }
1760      else
1761      {
1762        num=999999;
1763      }
1764    }
1765    fout<<num<<endl;
1766
1767    fout<<"-Estabilidade_eletrica_depois_do_envelhecimento:␣"<<endl;
1768
1769    cout<<"Estabilidade␣eletrica␣depois␣do␣envelhecimento:␣";
1770    cin>>nome;
1771    cin.get();
1772
1773    if(nome=="-")
1774    {
1775      num=999999;
1776    }
1777    else
1778    {
1779      if(atof(nome.c_str()))
1780      {
1781        num=atof(nome.c_str());
1782      }
1783      else
1784      {
1785        num=999999;
1786      }
1787    }
1788    fout<<num<<endl;
1789
1790    fout<<"-Coeficiente_de_lubricidade:␣"<<endl;
1791
1792    cout<<"Coeficiente␣de␣Lubricidade:␣";
1793    cin>>nome;
1794    cin.get();
```

```
1795
1796    if(nome=="-")
1797    {
1798      num=999999;
1799    }
1800    else
1801    {
1802      if(atof(nome.c_str()))
1803      {
1804        num=atof(nome.c_str());
1805      }
1806      else
1807      {
1808        num=999999;
1809      }
1810    }
1811    fout<<num<<endl;
1812
1813    fout<<"-Volume_de_Filtrado:␣"<<endl;
1814
1815    cout<<"Volume␣de␣Filtrado:␣";
1816    cin>>nome;
1817    cin.get();
1818
1819    if(nome=="-")
1820    {
1821      num=999999;
1822    }
1823    else
1824    {
1825      if(atof(nome.c_str()))
1826      {
1827        num=atof(nome.c_str());
1828      }
1829      else
1830      {
1831        num=999999;
1832      }
1833    }
1834    fout<<num<<endl;
1835
1836    fout<<"-Teor_de_solidos:␣"<<endl;
1837
1838    cout<<"Teor␣de␣solidos:␣";
1839    cin>>nome;
1840    cin.get();
1841
1842    if(nome=="-")
```

```
1843    {
1844      num=999999;
1845    }
1846    else
1847    {
1848      if(atof(nome.c_str()))
1849      {
1850        num=atof(nome.c_str());
1851      }
1852      else
1853      {
1854        num=999999;
1855      }
1856    }
1857    fout<<num<<endl;
1858
1859    fout<<"-Salinidade:␣"<<endl;
1860
1861    cout<<"Salinidade:␣";
1862    cin>>nome;
1863    cin.get();
1864
1865    if(nome=="-")
1866    {
1867      num=999999;
1868    }
1869    else
1870    {
1871      if(atof(nome.c_str()))
1872      {
1873        num=atof(nome.c_str());
1874      }
1875      else
1876      {
1877        num=999999;
1878      }
1879    }
1880    fout<<num<<endl;
1881
1882    fout<<"_____________________________________________________________
           "<<endl;
1883    fout<<"PROPRIEDADES_REOLOGICAS"<<endl;
1884    fout<<"-Viscosidade_aparente_antes_do_envelhecimento:␣"<<endl;
1885
1886    cout<<"Viscosidade␣aparente␣antes␣do␣envelhecimento:␣";
1887    cin>>nome;
1888    cin.get();
1889
```

```
1890    if(nome=="-")
1891    {
1892      num=999999;
1893    }
1894    else
1895    {
1896      if(atof(nome.c_str()))
1897      {
1898        num=atof(nome.c_str());
1899      }
1900      else
1901      {
1902        num=999999;
1903      }
1904    }
1905    fout<<num<<endl;
1906
1907    fout<<"-Viscosidade_aparente_depois_do_envelhecimento:␣"<<endl;
1908
1909    cout<<"Viscosidade␣aparente␣depois␣do␣envelhecimento:␣";
1910    cin>>nome;
1911    cin.get();
1912
1913    if(nome=="-")
1914    {
1915      num=999999;
1916    }
1917    else
1918    {
1919      if(atof(nome.c_str()))
1920      {
1921        num=atof(nome.c_str());
1922      }
1923      else
1924      {
1925        num=999999;
1926      }
1927    }
1928    fout<<num<<endl;
1929
1930    fout<<"-Viscosidade_plastica_antes_do_envelhecimento:␣"<<endl;
1931
1932    cout<<"Viscosidade␣plastica␣antes␣do␣envelhecimento:␣";
1933    cin>>nome;
1934    cin.get();
1935
1936    if(nome=="-")
1937    {
```

```
1938        num=999999;
1939      }
1940      else
1941      {
1942        if(atof(nome.c_str()))
1943        {
1944          num=atof(nome.c_str());
1945        }
1946        else
1947        {
1948          num=999999;
1949        }
1950      }
1951      fout<<num<<endl;
1952
1953      fout<<"-Viscosidade_plastica_depois_do_envelhecimento:␣"<<endl;
1954
1955      cout<<"Viscosidade␣plastica␣depois␣do␣envelhecimento:␣";
1956      cin>>nome;
1957      cin.get();
1958
1959      if(nome=="-")
1960      {
1961        num=999999;
1962      }
1963      else
1964      {
1965        if(atof(nome.c_str()))
1966        {
1967          num=atof(nome.c_str());
1968        }
1969        else
1970        {
1971          num=999999;
1972        }
1973      }
1974      fout<<num<<endl;
1975
1976      fout<<"-Limite_de_escoamento_antes_do_envelhecimento:␣"<<endl;
1977
1978      cout<<"Limite␣de␣Escoamento␣antes␣do␣envelheciemnto:␣␣";
1979      cin>>nome;
1980      cin.get();
1981
1982      if(nome=="-")
1983      {
1984        num=999999;
1985      }
```

```
1986      else
1987      {
1988        if(atof(nome.c_str()))
1989        {
1990          num=atof(nome.c_str());
1991        }
1992        else
1993        {
1994          num=999999;
1995        }
1996      }
1997      fout<<num<<endl;
1998
1999      fout<<"-Limite_de_escoamento_depois_do_envelhecimento:␣"<<endl;
2000
2001      cout<<"Limite␣de␣Escoamento␣depois␣do␣envelheciemnto:␣␣";
2002      cin>>nome;
2003      cin.get();
2004
2005      if(nome=="-")
2006      {
2007        num=999999;
2008      }
2009      else
2010      {
2011        if(atof(nome.c_str()))
2012        {
2013          num=atof(nome.c_str());
2014        }
2015        else
2016        {
2017          num=999999;
2018        }
2019      }
2020      fout<<num<<endl;
2021
2022      fout<<"_____________________________________________________________
             "<<endl;
2023      fout<<"ADITIVOS"<<endl;
2024      fout<<"-Adensante:␣"<<endl;
2025      cout<<"Adensante␣utilizado␣(9␣caracteres):␣";
2026      cin>>nome;
2027      cin.get();
2028      fout<<nome<<endl;
2029
2030      fout<<"-Concentracao_do_adensante:␣"<<endl;
2031
2032      cout<<"␣Concentracao␣do␣adensante:␣";
```

```
2033    cin>>nome;
2034    cin.get();
2035
2036    if(nome=="-")
2037    {
2038      num=999999;
2039    }
2040    else
2041    {
2042      if(atof(nome.c_str()))
2043      {
2044        num=atof(nome.c_str());
2045      }
2046      else
2047      {
2048        num=999999;
2049      }
2050    }
2051
2052    fout<<num<<endl;
2053
2054    fout<<"-Inibidor_de_formacoes_ativas:␣"<<endl;
2055    cout<<"Inibidor␣de␣formacoes␣ativas␣(9␣caracteres):␣";
2056    cin>>nome;
2057    cin.get();
2058    fout<<nome<<endl;
2059
2060    fout<<"-Concentração_do_inibidor_de_formacoes_ativas:␣"<<endl;
2061
2062    cout<<"Concentração␣do␣Inibidor␣de␣formacoes␣ativas:␣";
2063    cin>>nome;
2064    cin.get();
2065
2066    if(nome=="-")
2067    {
2068      num=999999;
2069    }
2070    else
2071    {
2072      if(atof(nome.c_str()))
2073      {
2074        num=atof(nome.c_str());
2075      }
2076      else
2077      {
2078        num=999999;
2079      }
2080    }
```

```
2081
2082      fout<<num<<endl;
2083
2084      fout<<"-Redutor_de_filtrado:␣"<<endl;
2085      cout<<"Redutor␣de␣filtrado␣(9␣caracteres):␣";
2086      cin>>nome;
2087      cin.get();
2088      fout<<nome<<endl;
2089
2090      fout<<"-Concentracao_do_redutor_de_filtrado:␣"<<endl;
2091
2092      cout<<"Concentracao␣do␣redutor␣de␣filtrado:␣";
2093      cin>>nome;
2094      cin.get();
2095
2096      if(nome=="-")
2097      {
2098        num=999999;
2099      }
2100      else
2101      {
2102        if(atof(nome.c_str()))
2103        {
2104          num=atof(nome.c_str());
2105        }
2106        else
2107        {
2108          num=999999;
2109        }
2110      }
2111
2112      fout<<num<<endl;
2113
2114      fout<<"-Biopolimero:␣"<<endl;
2115      cout<<"Biopolimero␣(9␣caracteres):␣";
2116      cin>>nome;
2117      cin.get();
2118      fout<<nome<<endl;
2119
2120      fout<<"-Concentracao_do_biopolimero:␣"<<endl;
2121
2122      cout<<"Concentracao␣do␣biopolimero␣␣:";
2123      cin>>nome;
2124      cin.get();
2125
2126      if(nome=="-")
2127      {
2128        num=999999;
```

```
2129    }
2130    else
2131    {
2132      if(atof(nome.c_str()))
2133      {
2134        num=atof(nome.c_str());
2135      }
2136      else
2137      {
2138        num=999999;
2139      }
2140    }
2141
2142    fout<<num<<endl;
2143
2144    fout<<"-Viscosificante:␣"<<endl;
2145    cout<<"Viscosificante␣(9␣caracteres):␣";
2146    cin>>nome;
2147    cin.get();
2148    fout<<nome<<endl;
2149
2150    fout<<"-Concentracao_do_viscosificante:␣"<<endl;
2151
2152    cout<<"Concentracao␣do␣viscosificante:␣";
2153    cin>>nome;
2154    cin.get();
2155
2156    if(nome=="-")
2157    {
2158      num=999999;
2159    }
2160    else
2161    {
2162      if(atof(nome.c_str()))
2163      {
2164        num=atof(nome.c_str());
2165      }
2166      else
2167      {
2168        num=999999;
2169      }
2170    }
2171
2172    fout<<num<<endl;
2173
2174    fout<<"-Dispersante:␣"<<endl;
2175    cout<<"Dispersante␣(9␣caracteres):␣";
2176    cin>>nome;
```

```
2177    cin.get();
2178    fout<<nome<<endl;
2179
2180    fout<<"-Concentracao_do_dispersante:␣"<<endl;
2181
2182    cout<<"Concentracao␣do␣dispersante:␣";
2183    cin>>nome;
2184    cin.get();
2185
2186    if(nome=="-")
2187    {
2188      num=999999;
2189    }
2190    else
2191    {
2192      if(atof(nome.c_str()))
2193      {
2194        num=atof(nome.c_str());
2195      }
2196      else
2197      {
2198        num=999999;
2199      }
2200    }
2201
2202    fout<<num<<endl;
2203
2204    fout<<"-Defloculante:␣"<<endl;
2205    cout<<"Defloculante␣(9␣caracteres):␣";
2206    cin>>nome;
2207    cin.get();
2208    fout<<nome<<endl;
2209
2210    fout<<"-Concentracao_do_defloculante:␣"<<endl;
2211
2212    cout<<"Concentracao␣do␣defloculante:␣";
2213    cin>>nome;
2214    cin.get();
2215
2216    if(nome=="-")
2217    {
2218      num=999999;
2219    }
2220    else
2221    {
2222      if(atof(nome.c_str()))
2223      {
2224        num=atof(nome.c_str());
```

```
2225        }
2226        else
2227        {
2228          num=999999;
2229        }
2230      }
2231
2232      fout<<num<<endl;
2233
2234      fout<<"-Emulsificante:␣"<<endl;
2235      cout<<"Emulsificante␣(9␣caracteres):␣";
2236      cin>>nome;
2237      cin.get();
2238      fout<<nome<<endl;
2239
2240      fout<<"-Concentracao_do_emulsificante:␣"<<endl;
2241
2242      cout<<"Concentracao␣do␣emulsificante:␣";
2243      cin>>nome;
2244      cin.get();
2245
2246      if(nome=="-")
2247      {
2248        num=999999;
2249      }
2250      else
2251      {
2252        if(atof(nome.c_str()))
2253        {
2254          num=atof(nome.c_str());
2255        }
2256        else
2257        {
2258          num=999999;
2259        }
2260      }
2261
2262      fout<<num<<endl;
2263
2264      fout<<"-Biocida:␣"<<endl;
2265      cout<<"Biocida␣(9␣caracteres):␣";
2266      cin>>nome;
2267      cin.get();
2268      fout<<nome<<endl;
2269
2270      fout<<"-Concentracao_do_biocida:␣"<<endl;
2271
2272      cout<<"Concentracao␣do␣biocida:␣";
```

```
2273     cin>>nome;
2274     cin.get();
2275 
2276     if(nome=="-")
2277     {
2278       num=999999;
2279     }
2280     else
2281     {
2282       if(atof(nome.c_str()))
2283       {
2284         num=atof(nome.c_str());
2285       }
2286       else
2287       {
2288         num=999999;
2289       }
2290     }
2291 
2292     fout<<num<<endl;
2293 
2294     fout<<"-Lubrificante:␣"<<endl;
2295     cout<<"Lubrificante␣(9␣caracteres):␣";
2296     cin>>nome;
2297     cin.get();
2298     fout<<nome<<endl;
2299 
2300     fout<<"-Concentracao_do_lubrificante:␣"<<endl;
2301 
2302     cout<<"Concentracao␣do␣lubrificante:␣";
2303     cin>>nome;
2304     cin.get();
2305 
2306     if(nome=="-")
2307     {
2308       num=999999;
2309     }
2310     else
2311     {
2312       if(atof(nome.c_str()))
2313       {
2314         num=atof(nome.c_str());
2315       }
2316       else
2317       {
2318         num=999999;
2319       }
2320     }
```

```
2321
2322    fout<<num<<endl;
2323
2324    fout<<"-Inibidor_de_corrosao:␣"<<endl;
2325    cout<<"Inibidor␣de␣corrosao␣(9␣caracteres):␣";
2326    cin>>nome;
2327    cin.get();
2328    fout<<nome<<endl;
2329
2330    fout<<"-Concentracao_do_inibidor_de_corrosao:␣"<<endl;
2331
2332    cout<<"Concentracao␣do␣inibidor␣de␣corrosao:␣";
2333    cin>>nome;
2334    cin.get();
2335
2336    if(nome=="-")
2337    {
2338      num=999999;
2339    }
2340    else
2341    {
2342      if(atof(nome.c_str()))
2343      {
2344        num=atof(nome.c_str());
2345      }
2346      else
2347      {
2348        num=999999;
2349      }
2350    }
2351
2352    fout<<num<<endl;
2353
2354    fout<<"-Controlador_de_pH:␣"<<endl;
2355    cout<<"Controlador␣de␣pH␣(9␣caracteres):␣";
2356    cin>>nome;
2357    cin.get();
2358    fout<<nome<<endl;
2359
2360    fout<<"-Concentracao_do_controlador_de_pH:␣"<<endl;
2361    cout<<"Concentracao␣do␣controlador␣de␣pH:␣";
2362    cin>>nome;
2363    cin.get();
2364
2365    if(nome=="-")
2366    {
2367      num=999999;
2368    }
```

```
2369      else
2370      {
2371        if(atof(nome.c_str()))
2372        {
2373          num=atof(nome.c_str());
2374        }
2375        else
2376        {
2377          num=999999;
2378        }
2379      }
2380
2381      fout<<num<<endl;
2382
2383      fout.close();
2384
2385
2386      ////////////////////////////////////
2387      //fim da inclusao do arquivo basico
2388      ////////////////////////////////////
2389    }
```

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

Listing 6.5: Arquivo de cabeçalho da classe CConfiguracao.

```
2391 #ifndef CConfiguracao_h
2392 #define CConfiguracao_h
2393 #include <string>
2394 #include <sstream>
2395 #include <iostream>
2396 #include <fstream>
2397 using namespace std;
2398
2399 class CConfiguracao
2400 {
2401 public:
2402
2403   ifstream fin;
2404   ofstream fout;
2405
2406   stringstream caminho_ss;
2407   string caminho_str;
2408   string nome_programa;
2409   string nome_explorador_linux;
2410   string nome_explorador_windows;
2411   string nome_editor_texto_linux;
2412   string nome_editor_texto_windows;
2413   string tipo_programa;
2414   string str;
```

```
2415
2416   int opcao;
2417
2418 public:
2419
2420   int opcao_abrir_DT();
2421   void opcao_abrir_DT(int);
2422   void opcao_programa(string, string);
2423   string opcao_programa(string);
2424
2425 };
2426
2427 #endif
```

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

Listing 6.6: Arquivo de implementação da classe CConfiguracao.

```
2428 #include <iostream>
2429 #include <stdio.h>
2430 //verifica sistema operacional
2431 #ifdef __linux
2432
2433 #include <cstdlib>
2434
2435 #elif _WIN32
2436
2437 #include <windows.h>
2438 #include <stdlib.h>//para system
2439 #include <conio.h>
2440
2441 #else
2442
2443 #endif
2444
2445 #include <string>
2446 #include <sstream>
2447 #include <cstdlib>
2448 #include <fstream>
2449
2450 #include "CConfiguracao.h"
2451
2452 using namespace std;
2453
2454 int CConfiguracao::opcao_abrir_DT()
2455 {
2456   caminho_ss.str("");
2457   #ifdef __linux
2458   caminho_ss<<"config"<<char(47)<<"DT.ini";
2459   #elif _WIN32
```

```
2460   caminho_ss<<"config"<<char(92)<<"DT.ini";
2461   #else
2462
2463   #endif
2464   fin.open(caminho_ss.str().c_str());
2465   if(fin.fail())
2466   {
2467     cout<<"Não␣foi␣possivel␣ler."<<endl;
2468   }
2469   caminho_ss.str("");
2470   fin>>opcao;
2471   fin.close();
2472   return opcao;
2473 }
2474
2475
2476 void CConfiguracao::opcao_abrir_DT(int opcao_)
2477 {
2478   opcao=opcao_;
2479   caminho_ss.str("");
2480   #ifdef __linux
2481   caminho_ss<<"config"<<char(47)<<"DT.ini";
2482   #elif _WIN32
2483   caminho_ss<<"config"<<char(92)<<"DT.ini";
2484   #else
2485
2486   #endif
2487   fout.open(caminho_ss.str().c_str());
2488   if(fout.fail())
2489   {
2490     cout<<"Não␣foi␣possivel␣editar."<<endl;
2491   }
2492   caminho_ss.str("");
2493   fout<<opcao;
2494   fout.close();
2495 }
2496
2497 void CConfiguracao::opcao_programa(string tipo_programa_, string
       nome_programa_)
2498 {
2499   tipo_programa=tipo_programa_;
2500   nome_programa=nome_programa_;
2501   caminho_ss.str("");
2502   #ifdef __linux
2503   caminho_ss<<"config"<<char(47)<<"programas.ini";
2504   #elif _WIN32
2505   caminho_ss<<"config"<<char(92)<<"programas.ini";
2506   #else
```

```
2507
2508  #endif
2509  //le o arquivo todo
2510  fin.open(caminho_ss.str().c_str());
2511  if(fin.fail())
2512  {
2513    cout<<"Não foi possivel ler."<<endl;
2514  }
2515  caminho_ss.str("");
2516  fin>>str;
2517  fin>>str;
2518  fin>>str;
2519  fin>>str;
2520
2521  fin>>nome_explorador_windows;
2522  fin>>nome_explorador_linux;
2523  fin>>str;
2524
2525  fin>>nome_editor_texto_windows;
2526  fin>>nome_editor_texto_linux;
2527
2528  fin.close();
2529
2530  caminho_ss.str("");
2531  #ifdef __linux
2532  caminho_ss<<"config"<<char(47)<<"programas.ini";
2533  #elif _WIN32
2534  caminho_ss<<"config"<<char(92)<<"programas.ini";
2535  #else
2536
2537  #endif
2538
2539  //escreve o arquivo
2540  fout.open(caminho_ss.str().c_str());
2541  if(fout.fail())
2542  {
2543    cout<<"Não foi possivel editar."<<endl;
2544  }
2545  caminho_ss.str("");
2546  fout<<"Tipo-do-Programa:"<<endl;
2547  fout<<"nome-windows"<<endl;
2548  fout<<"nome-linux"<<endl;
2549  fout<<"explorador:"<<endl;
2550  if(tipo_programa=="explorador")
2551  {
2552    #ifdef __linux
2553    fout<<nome_explorador_windows<<endl;
2554    fout<<nome_programa<<endl;
```

```
2555      #elif _WIN32
2556      fout<<nome_programa<<endl;
2557      fout<<nome_explorador_linux<<endl;
2558      #else
2559
2560      #endif
2561    }
2562    else
2563    {
2564      fout<<nome_explorador_windows<<endl;
2565      fout<<nome_explorador_linux<<endl;
2566    }
2567
2568
2569
2570    fout<<"editor-texto:"<<endl;
2571    if(tipo_programa=="editortexto")
2572    {
2573      #ifdef __linux
2574      fout<<nome_editor_texto_windows<<endl;
2575      fout<<nome_programa<<endl;
2576      #elif _WIN32
2577      fout<<nome_programa<<endl;
2578      fout<<nome_editor_texto_linux<<endl;
2579      #else
2580
2581      #endif
2582    }
2583    else
2584    {
2585      fout<<nome_editor_texto_windows<<endl;
2586      fout<<nome_editor_texto_linux<<endl;
2587    }
2588
2589
2590    fout<<endl;
2591    fout<<"#OBS:␣O␣nome␣do␣programa␣é␣o␣mesmo␣que␣você␣digita␣no␣cmd␣do␣
          windows"<<endl;
2592    fout<<"#ou␣no␣terminal␣do␣linux."<<endl;
2593    fout.close();
2594
2595}
2596
2597string CConfiguracao::opcao_programa(string tipo_programa_)
2598{
2599    tipo_programa=tipo_programa_;
2600    caminho_ss.str("");
2601    #ifdef __linux
```

```
2602  caminho_ss<<"config"<<char(47)<<"programas.ini";
2603  #elif _WIN32
2604  caminho_ss<<"config"<<char(92)<<"programas.ini";
2605  #else
2606
2607  #endif
2608  //le o arquivo todo
2609  fin.open(caminho_ss.str().c_str());
2610  if(fin.fail())
2611  {
2612    cout<<"Não foi possivel ler."<<endl;
2613  }
2614  caminho_ss.str("");
2615  fin>>str;
2616  fin>>str;
2617  fin>>str;
2618  fin>>str;
2619
2620  fin>>nome_explorador_windows;
2621  fin>>nome_explorador_linux;
2622
2623  fin>>str;
2624
2625  fin>>nome_editor_texto_windows;
2626  fin>>nome_editor_texto_linux;
2627
2628  fin.close();
2629
2630
2631  if(tipo_programa=="explorador")
2632  {
2633    #ifdef __linux
2634    nome_programa = nome_explorador_linux;
2635    #elif _WIN32
2636    nome_programa = nome_explorador_windows;
2637    #else
2638
2639    #endif
2640  }
2641
2642  if(tipo_programa=="editortexto")
2643  {
2644    #ifdef __linux
2645    nome_programa = nome_editor_texto_linux;
2646    #elif _WIN32
2647    nome_programa = nome_editor_texto_windows;
2648    #else
2649
```

```
2650     #endif
2651   }
2652   return nome_programa;
2653
2654 }
```

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

Listing 6.7: Arquivo de cabeçalho da classe CHeader.

```
2655 #ifndef CHeader_h
2656 #define CHeader_h
2657 #include <iostream>
2658
2659 class CHeader
2660 {
2661 public:
2662
2663   int i;//vetor i contagem de for
2664   int largura;//vetor de largura do retangulo do titulo
2665
2666 public:
2667    void header_titulo();//Escreve titulo do programa
2668   void header_inicio();//escreve subtitulo do programa
2669   void header_banco_de_dados();//escreve subtitulo de listar amostras
2670   void header_banco_de_dados_lista_de_amostras();//escreve subtitulo de
        amostras listadas
2671   void header_banco_de_dados_exibir_amostras();//escreve subtitulo de
        exibir amostras
2672   void header_banco_de_dados_pesquisa_de_amostras();//escreve subtitulo
        de amostras listadas
2673   void header_banco_de_dados_inserir_amostras();//escreve subtitulo de
        inserir amostras
2674   void header_banco_de_dados_excluir_amostras();//escreve subtitulo de
        excluir amostras
2675   void header_banco_de_dados_exportar_amostras();//escreve subtitulo de
        excluir amostras
2676   void header_configuracoes();//escreve subtitulo do menu 2
2677   void header_menu_abre_DT();//escreve subtitulo do menu
2678   void header_menu_abre_bd();//escreve subtitulo do menu
2679
2680
2681
2682
2683 };
2684
2685
2686 #endif
```

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

Listing 6.8: Arquivo de implementação da classe CHeader.

```
//verifica sistema operacional
#ifdef __linux
#include <cstdlib>
//  #include <unistd.h>
#elif _WIN32

#include <windows.h>
#include <stdlib.h>//para system
#include <conio.h>
#else
#endif
#include <iostream>
#include <stdio.h>
#include <locale>
#include "CHeader.h"
using namespace std;


void CHeader::header_titulo()//escreve titulo do programa
{

  //Titulo Programa
  //verifica sistema operacional
  #ifdef __linux
  //system("clear");

  {
    largura=45;
    cout<<"\u2554";//canto sup esq
    for(i=0;i<largura;i++)//barra sup
      cout<<"\u2550";
    cout<<"\u2557"<<endl;//canto su dir
    cout<<"\u2551"<<"                                             "<<"\u2551"<<endl;;//Margem vert acim Titulo
    cout<<"\u2551";//Margem vert Titulo
    cout<<"   Banco de Dados de Fluidos de Perfuracao   ";
    cout<<"\u2551"<<endl;//Margem vert Titulo
    cout<<"\u255A";//canto inf dir
    for(i=0;i<largura;i++)//barra inf
      cout<<"\u2550";
    cout<<"\u255D"<<endl;//canto inf esq
  }
  #elif _WIN32
  //system("cls");

  {

```

```
    largura=30;
    cout<<(char)201;//canto sup esq
    for(i=0;i<largura;i++)//barra sup
      cout<<(char)205;
    cout<<(char)187<<endl;//canto su dir
    cout<<(char)186<<"                                           "<<(
       char)186<<endl;;//Margem vert acim Titulo
    cout<<(char)186;//Margem vert Titulo
    cout<<"   Banco de Dados de Fluidos de Perfuração   ";
    cout<<endl;
    cout<<(char)186<<endl;//Margem vert Titulo
    cout<<(char)200;//canto inf dir
    for(i=0;i<largura;i++)//barra inf
      cout<<(char)205;
    cout<<(char)188<<endl;//canto inf esq
  }





  #else

  #endif


}



    void CHeader::header_inicio()//escreve subtitulo do menu inicio
    {
      #ifdef __linux
      {

        largura=30;
        cout<<"\u250C";//canto sup esq
        for(i=0;i<largura;i++)//barra sup
          cout<<"\u2500";
        cout<<"\u2510"<<endl;//canto su dir
        cout<<"\u2502";//Margem vert Titulo
        cout<<"            In\u00EDcio            ";
        cout<<"\u2502"<<endl;//Margem vert Titulo
        cout<<"\u2514";//canto inf dir
        for(i=0;i<largura;i++)//barra inf
          cout<<"\u2500";
        cout<<"\u2518"<<endl;
        cout<<endl;
```

```
2780
2781        }
2782
2783
2784
2785
2786
2787        #elif _WIN32
2788        {
2789          largura=30;
2790          cout<<(char)218;//canto sup esq
2791          for(i=0;i<largura;i++)//barra sup
2792            cout<<(char)196;
2793          cout<<(char)191<<endl;//canto su dir
2794          cout<<(char)179;//Margem vert Titulo
2795          cout<<"␣␣␣␣␣␣␣␣␣␣␣␣␣␣Início␣␣␣␣␣␣␣␣␣␣␣␣␣␣";
2796          cout<<(char)179<<endl;//Margem vert Titulo
2797          cout<<(char)192;//canto inf dir
2798          for(i=0;i<largura;i++)//barra inf
2799            cout<<(char)196;
2800          cout<<(char)217<<endl;
2801          cout<<endl;
2802
2803
2804        }
2805        #else
2806
2807        #endif
2808
2809      }
2810
2811      void CHeader::header_banco_de_dados()//escreve subtitulo para banco
          de dados
2812      {
2813        #ifdef __linux
2814        {
2815          largura=30;
2816          cout<<"\u250C";//canto sup esq
2817          for(i=0;i<largura;i++)//barra sup
2818            cout<<"\u2500";
2819          cout<<"\u2510"<<endl;//canto su dir
2820          cout<<"\u2502";//Margem vert Titulo
2821          cout<<"␣␣␣␣␣␣␣␣Banco␣de␣Dados␣␣␣␣␣␣␣␣";
2822          cout<<"\u2502"<<endl;//Margem vert Titulo
2823          cout<<"\u2514";//canto inf dir
2824          for(i=0;i<largura;i++)//barra inf
2825            cout<<"\u2500";
2826          cout<<"\u2518"<<endl;
```

```
2827          cout<<endl;
2828
2829        }
2830        #elif _WIN32
2831        {
2832          largura=30;
2833          cout<<(char)218;//canto sup esq
2834          for(i=0;i<largura;i++)//barra sup
2835            cout<<(char)196;
2836          cout<<(char)191<<endl;//canto su dir
2837          cout<<(char)179;//Margem vert Titulo
2838
2839          cout<<"␣␣␣␣␣␣␣␣Banco␣de␣Dados␣␣␣␣␣␣␣␣";
2840
2841          cout<<(char)179<<endl;//Margem vert Titulo
2842          cout<<(char)192;//canto inf dir
2843          for(i=0;i<largura;i++)//barra inf
2844            cout<<(char)196;
2845          cout<<(char)217<<endl;
2846          cout<<endl;
2847        }
2848
2849        #else
2850
2851        #endif
2852
2853      }
2854
2855      void CHeader::header_banco_de_dados_lista_de_amostras()//escreve
           subtitulo para lista de fluidos
2856      {
2857        #ifdef __linux
2858
2859        {
2860          largura=40;
2861          cout<<"\u250C";//canto sup esq
2862          for(i=0;i<largura;i++)//barra sup
2863            cout<<"\u2500";
2864          cout<<"\u2510"<<endl;//canto su dir
2865          cout<<"\u2502";//Margem vert Titulo
2866          cout<<"␣␣␣␣␣␣Lista␣de␣Fluidos␣de␣Perfuracao␣␣␣␣";
2867          cout<<"\u2502"<<endl;//Margem vert Titulo
2868          cout<<"\u2514";//canto inf dir
2869          for(i=0;i<largura;i++)//barra inf
2870            cout<<"\u2500";
2871          cout<<"\u2518"<<endl;
2872          cout<<endl;
2873        }
```

```
2874
2875        #elif _WIN32
2876
2877        {
2878          largura=30;
2879          cout<<(char)218;//canto sup esq
2880          for(i=0;i<largura;i++)//barra sup
2881            cout<<(char)196;
2882          cout<<(char)191<<endl;//canto su dir
2883          cout<<(char)179;//Margem vert Titulo
2884          cout<<"␣␣␣␣␣␣␣Lista␣de␣Fluidos␣de␣Perfuração␣␣␣␣␣␣␣␣";
2885          cout<<(char)179<<endl;//Margem vert Titulo
2886          cout<<(char)192;//canto inf dir
2887          for(i=0;i<largura;i++)//barra inf
2888            cout<<(char)196;
2889          cout<<(char)217<<endl;
2890          cout<<endl;
2891        }
2892
2893
2894        #else
2895
2896        #endif
2897      }
2898
2899      void CHeader::header_banco_de_dados_exibir_amostras()//escreve
          subtitulo das amostras listadas
2900      {
2901        #ifdef __linux
2902
2903
2904        largura=30;
2905        cout<<"\u250C";//canto sup esq
2906        for(i=0;i<largura;i++)//barra sup
2907          cout<<"\u2500";
2908        cout<<"\u2510"<<endl;//canto su dir
2909        cout<<"\u2502";//Margem vert Titulo
2910
2911
2912        cout<<"␣␣␣␣␣␣␣␣␣␣␣␣␣Exibir␣␣␣␣␣␣␣␣␣␣␣␣␣";
2913
2914        cout<<"\u2502"<<endl;//Margem vert Titulo
2915        cout<<"\u2514";//canto inf dir
2916        for(i=0;i<largura;i++)//barra inf
2917          cout<<"\u2500";
2918        cout<<"\u2518"<<endl;
2919        cout<<endl;
2920
```

```
2921
2922        #elif _WIN32
2923
2924        {
2925          largura=30;
2926          cout<<(char)218;//canto sup esq
2927          for(i=0;i<largura;i++)//barra sup
2928            cout<<(char)196;
2929          cout<<(char)191<<endl;//canto su dir
2930          cout<<(char)179;//Margem vert Titulo
2931
2932          cout<<"␣␣␣␣␣␣␣␣␣␣␣␣␣Exibir␣␣␣␣␣␣␣␣␣␣␣␣␣";
2933
2934
2935          cout<<(char)179<<endl;//Margem vert Titulo
2936          cout<<(char)192;//canto inf dir
2937          for(i=0;i<largura;i++)//barra inf
2938            cout<<(char)196;
2939          cout<<(char)217<<endl;
2940          cout<<endl;
2941        }
2942
2943        #else
2944
2945        #endif
2946
2947      }
2948
2949      void CHeader::header_banco_de_dados_pesquisa_de_amostras()//escreve
          subtitulo para pesquisa de amostras
2950      {
2951
2952        #ifdef __linux
2953
2954        {
2955          largura=40;
2956          cout<<"\u250C";//canto sup esq
2957          for(i=0;i<largura;i++)//barra sup
2958            cout<<"\u2500";
2959          cout<<"\u2510"<<endl;//canto su dir
2960          cout<<"\u2502";//Margem vert Titulo
2961
2962          cout<<"␣␣␣␣␣Pesquisa␣de␣Fluidos␣de␣Perfuracao␣␣";
2963
2964          cout<<"\u2502"<<endl;//Margem vert Titulo
2965          cout<<"\u2514";//canto inf dir
2966          for(i=0;i<largura;i++)//barra inf
2967            cout<<"\u2500";
```

```
2968          cout<<"\u2518"<<endl;
2969          cout<<endl;
2970        }
2971        #elif _WIN32
2972        {
2973          largura=30;
2974          cout<<(char)218;//canto sup esq
2975          for(i=0;i<largura;i++)//barra sup
2976            cout<<(char)196;
2977          cout<<(char)191<<endl;//canto su dir
2978          cout<<(char)179;//Margem vert Titulo
2979
2980          cout<<"␣␣␣␣␣Pesquisa␣de␣Fluidos␣de␣Perfuração␣␣␣";
2981
2982          cout<<(char)179<<endl;//Margem vert Titulo
2983          cout<<(char)192;//canto inf dir
2984          for(i=0;i<largura;i++)//barra inf
2985            cout<<(char)196;
2986          cout<<(char)217<<endl;
2987          cout<<endl;
2988        }
2989
2990        #else
2991
2992        #endif
2993
2994      }
2995      void CHeader::header_banco_de_dados_inserir_amostras()//escreve
             subtitulo para inserir fluidos
2996      {
2997
2998        #ifdef __linux
2999
3000        largura=40;
3001        cout<<"\u250C";//canto sup esq
3002        for(i=0;i<largura;i++)//barra sup
3003          cout<<"\u2500";
3004        cout<<"\u2510"<<endl;//canto su dir
3005        cout<<"\u2502";//Margem vert Titulo
3006
3007        cout<<"␣␣␣␣␣␣␣Inserir␣Fluidos␣de␣Perfuracao␣␣␣␣";
3008
3009        cout<<"\u2502"<<endl;//Margem vert Titulo
3010        cout<<"\u2514";//canto inf dir
3011        for(i=0;i<largura;i++)//barra inf
3012          cout<<"\u2500";
3013        cout<<"\u2518"<<endl;
3014        cout<<endl;
```

```
3015
3016        #elif _WIN32
3017
3018        {
3019          largura=30;
3020          cout<<(char)218;//canto sup esq
3021          for(i=0;i<largura;i++)//barra sup
3022            cout<<(char)196;
3023          cout<<(char)191<<endl;//canto su dir
3024          cout<<(char)179;//Margem vert Titulo
3025
3026          cout<<"␣␣␣␣␣␣␣␣Inserir␣Fluidos␣de␣Perfuracao␣␣␣␣";
3027
3028          cout<<(char)179<<endl;//Margem vert Titulo
3029          cout<<(char)192;//canto inf dir
3030          for(i=0;i<largura;i++)//barra inf
3031            cout<<(char)196;
3032          cout<<(char)217<<endl;
3033          cout<<endl;
3034        }
3035
3036        #else
3037
3038        #endif
3039      }
3040
3041      void CHeader::header_banco_de_dados_excluir_amostras()//escreve
           subtitulo das amostras listadas
3042      {
3043
3044        #ifdef __linux
3045
3046        largura=30;
3047        cout<<"\u250C";//canto sup esq
3048        for(i=0;i<largura;i++)//barra sup
3049          cout<<"\u2500";
3050        cout<<"\u2510"<<endl;//canto su dir
3051        cout<<"\u2502";//Margem vert Titulo
3052
3053
3054        cout<<"␣␣␣␣␣␣␣␣Excluir␣Amostras␣␣␣␣␣␣␣␣";
3055
3056        cout<<"\u2502"<<endl;//Margem vert Titulo
3057        cout<<"\u2514";//canto inf dir
3058        for(i=0;i<largura;i++)//barra inf
3059          cout<<"\u2500";
3060        cout<<"\u2518"<<endl;
3061        cout<<endl;
```

```
3062
3063        #elif _WIN32
3064
3065        {
3066          largura=30;
3067          cout<<(char)218;//canto sup esq
3068          for(i=0;i<largura;i++)//barra sup
3069            cout<<(char)196;
3070          cout<<(char)191<<endl;//canto su dir
3071          cout<<(char)179;//Margem vert Titulo
3072          cout<<"␣␣␣␣␣␣␣␣Excluir␣Amostras␣␣␣␣␣␣␣␣";
3073          cout<<(char)179<<endl;//Margem vert Titulo
3074          cout<<(char)192;//canto inf dir
3075          for(i=0;i<largura;i++)//barra inf
3076            cout<<(char)196;
3077          cout<<(char)217<<endl;
3078          cout<<endl;
3079        }
3080
3081        #else
3082
3083        #endif
3084
3085
3086      }
3087
3088
3089      void CHeader::header_banco_de_dados_exportar_amostras()//escreve
            subtitulo para exportar
3090      {
3091        #ifdef __linux
3092        {
3093
3094          largura=40;
3095          cout<<"\u250C";//canto sup esq
3096          for(i=0;i<largura;i++)//barra sup
3097            cout<<"\u2500";
3098          cout<<"\u2510"<<endl;//canto su dir
3099          cout<<"\u2502";//Margem vert Titulo
3100
3101          cout<<"␣␣␣␣␣␣␣␣Exportar␣Fluidos␣de␣Perfuracao␣␣␣␣";
3102
3103          cout<<"\u2502"<<endl;//Margem vert Titulo
3104          cout<<"\u2514";//canto inf dir
3105          for(i=0;i<largura;i++)//barra inf
3106            cout<<"\u2500";
3107          cout<<"\u2518"<<endl;
3108          cout<<endl;
```

```
3109        }
3110        #elif _WIN32
3111
3112        {
3113          largura=30;
3114          cout<<(char)218;//canto sup esq
3115          for(i=0;i<largura;i++)//barra sup
3116            cout<<(char)196;
3117          cout<<(char)191<<endl;//canto su dir
3118          cout<<(char)179;//Margem vert Titulo
3119
3120          cout<<"␣␣␣␣␣␣␣␣Exportar␣Fluidos␣de␣Perfuracao␣␣␣␣";
3121
3122          cout<<(char)179<<endl;//Margem vert Titulo
3123          cout<<(char)192;//canto inf dir
3124          for(i=0;i<largura;i++)//barra inf
3125            cout<<(char)196;
3126          cout<<(char)217<<endl;
3127          cout<<endl;
3128        }
3129
3130
3131        #else
3132
3133        #endif
3134
3135      }
3136
3137
3138
3139      void CHeader::header_configuracoes()//subtitulo do menu 2
3140      {
3141
3142
3143        #ifdef __linux
3144
3145        largura=30;
3146        cout<<"\u250C";//canto sup esq
3147        for(i=0;i<largura;i++)//barra sup
3148          cout<<"\u2500";
3149        cout<<"\u2510"<<endl;//canto su dir
3150        cout<<"\u2502";//Margem vert Titulo
3151        cout<<"␣␣␣␣␣␣␣␣␣Configura\u00E7\u00F5es␣␣␣␣␣␣␣␣␣";
3152        cout<<"\u2502"<<endl;//Margem vert Titulo
3153        cout<<"\u2514";//canto inf dir
3154        for(i=0;i<largura;i++)//barra inf
3155          cout<<"\u2500";
3156        cout<<"\u2518"<<endl;
```

```
3157        cout<<endl;
3158
3159        #elif _WIN32
3160        {
3161          largura=30;
3162          cout<<(char)218;//canto sup esq
3163          for(i=0;i<largura;i++)//barra sup
3164            cout<<(char)196;
3165          cout<<(char)191<<endl;//canto su dir
3166          cout<<(char)179;//Margem vert Titulo
3167
3168          cout<<"␣␣␣␣␣␣␣␣␣Configurações␣␣␣␣␣␣␣␣";
3169
3170          cout<<(char)179<<endl;//Margem vert Titulo
3171          cout<<(char)192;//canto inf dir
3172          for(i=0;i<largura;i++)//barra inf
3173            cout<<(char)196;
3174          cout<<(char)217<<endl;
3175          cout<<endl;
3176        }
3177
3178
3179        #else
3180
3181        #endif
3182      }
3183
3184      void CHeader::header_menu_abre_DT()//subtitulo do menu 3
3185      {
3186        #ifdef __linux
3187
3188        largura=30;
3189        cout<<"\u250C";//canto sup esq
3190        for(i=0;i<largura;i++)//barra sup
3191          cout<<"\u2500";
3192        cout<<"\u2510"<<endl;//canto su dir
3193        cout<<"\u2502";//Margem vert Titulo
3194
3195        cout<<"␣Abrindo␣Diret\u00F3rio␣de␣Trablho␣";
3196
3197        cout<<"\u2502"<<endl;//Margem vert Titulo
3198        cout<<"\u2514";//canto inf dir
3199        for(i=0;i<largura;i++)//barra inf
3200          cout<<"\u2500";
3201        cout<<"\u2518"<<endl;
3202        cout<<endl;
3203
3204        #elif _WIN32
```

```
3205        {
3206          largura=30;
3207          cout<<(char)218;//canto sup esq
3208          for(i=0;i<largura;i++)//barra sup
3209            cout<<(char)196;
3210          cout<<(char)191<<endl;//canto su dir
3211          cout<<(char)179;//Margem vert Titulo
3212
3213          cout<<"␣Abrindo␣Diretório␣de␣Trablho␣";
3214
3215          cout<<(char)179<<endl;//Margem vert Titulo
3216          cout<<(char)192;//canto inf dir
3217          for(i=0;i<largura;i++)//barra inf
3218            cout<<(char)196;
3219          cout<<(char)217<<endl;
3220          cout<<endl;
3221        }
3222
3223
3224        #else
3225
3226        #endif
3227      }
3228
3229      void CHeader::header_menu_abre_bd()//subtitulo do menu 3
3230      {
3231
3232        #ifdef __linux
3233
3234        largura=30;
3235        cout<<"\u250C";//canto sup esq
3236        for(i=0;i<largura;i++)//barra sup
3237          cout<<"\u2500";
3238        cout<<"\u2510"<<endl;//canto su dir
3239        cout<<"\u2502";//Margem vert Titulo
3240
3241        cout<<"␣␣Abrindo␣Diret\u00F3rio␣de␣Dados␣␣";
3242
3243        cout<<"\u2502"<<endl;//Margem vert Titulo
3244        cout<<"\u2514";//canto inf dir
3245        for(i=0;i<largura;i++)//barra inf
3246          cout<<"\u2500";
3247        cout<<"\u2518"<<endl;
3248        cout<<endl;
3249
3250        #elif _WIN32
3251        {
3252          largura=30;
```

```
3253          cout<<(char)218;//canto sup esq
3254          for(i=0;i<largura;i++)//barra sup
3255            cout<<(char)196;
3256          cout<<(char)191<<endl;//canto su dir
3257          cout<<(char)179;//Margem vert Titulo
3258
3259          cout<<"   Abrindo Diretório de Dados   ";
3260
3261          cout<<(char)179<<endl;//Margem vert Titulo
3262          cout<<(char)192;//canto inf dir
3263          for(i=0;i<largura;i++)//barra inf
3264            cout<<(char)196;
3265          cout<<(char)217<<endl;
3266          cout<<endl;
3267        }
3268
3269
3270
3271        #else
3272
3273        #endif
3274      }
```

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

Listing 6.9: Arquivo de cabeçalho da classe CInterface.

```
3277 #ifndef CInterface_h
3278 #define CInterface_h
3279 #include <iostream>
3280
3281 #ifdef __linux
3282
3283 #include <cstdlib>
3284
3285 #elif _WIN32
3286 #include <windows.h>//para configurar a janela
3287 #include <stdlib.h>//para system
3288 #else
3289
3290 #endif
3291
3292 #include "CMenu.h"
3293 #include "CHeader.h"
3294 #include "CBancodeDados.h"
3295 #include "CFluidodePerfuracao.h"
3296 #include "CRelatorio.h"
3297 #include "CConfiguracao.h"
3298
3299
```

```
3300 #include <string>
3301 #include <sstream>
3302 #include <vector>
3303
3304 class CInterface
3305 {
3306 public:
3307   int resposta;
3308   int int_chave;
3309   int item;
3310   unsigned int c;
3311
3312   CHeader Head;
3313   CMenu Menu;
3314   CBancodeDados bancodedados;
3315   CRelatorio relatorio;
3316   CConfiguracao configuracao;
3317
3318   stringstream comando_abrir;
3319   stringstream titulo_ss;
3320   string titulo;
3321   string informacao_str;
3322   string programa;
3323
3324   char ch;
3325
3326   vector<CFluidodePerfuracao> amostras;
3327   vector<double> vx;
3328   vector<double> vy;
3329   vector<int> vetor_resposta;
3330
3331 public:
3332   void inicia();
3333   void abre_DT();
3334   void abre_bd();
3335   void abre_relatorio(int);
3336   void ajusta_janela();
3337   void limpa_janela();
3338
3339
3340 };
3341
3342 #endif
```

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

Listing 6.10: Arquivo de implementação da classe CInterface.

```
3343 #include <iostream>
3344 //verifica sistema operacional
```

```
3345 #ifdef __linux
3346     #include <cstdlib>
3347     #include <stdio.h>
3348    // #include <unistd.h>
3349 #elif _WIN32
3350         #include <windows.h>//para configurar a janela
3351         #include <stdlib.h>//para system
3352 #else
3353
3354 #endif
3355
3356 #include <sstream>
3357 #include <cstring>//biblioteca para trab com os comandos
3358 #include <string>
3359 #include "CInterface.h"//inclui classe
3360 #include "CMenu.h"//Inclui classes utilizadas
3361 #include "CHeader.h"
3362 #include "CFluidodePerfuracao.h"
3363
3364
3365 using namespace std;
3366
3367 //usuario simples
3368
3369 void CInterface::inicia()//inicia o menu
3370 {
3371
3372
3373   ajusta_janela();
3374   do{
3375
3376     limpa_janela();
3377     Head.header_titulo();//Coloca titulo principal do programa
3378     Head.header_inicio();//Coloca subtitulo Inicio
3379     resposta=Menu.menu_inicio();//Escreve o menu inicio e obtem a
          resposta
3380
3381     switch (resposta)//testa a resposta
3382     {
3383       case 1://Banco de dados
3384         do
3385         {
3386           limpa_janela();
3387           Head.header_titulo();//Coloca titulo pricipal do programa
3388           Head.header_banco_de_dados();//Coloca subtitulo editar
                amostras
3389           resposta=Menu.menu_banco_de_dados(); //obtem resposta do menu
                editar amostra
```

```
3390
3391            switch (resposta)//faz a funçao respctiva do submenu editar
                    amostra
3392            {
3393              case 1://listar
3394                limpa_janela();
3395                Head.header_titulo();
3396                Head.header_banco_de_dados_lista_de_amostras();
3397                amostras=bancodedados.ler_amostra_basico();
3398                bancodedados.listar_amostra(amostras);
3399                cout<<endl;
3400                amostras.clear();
3401                  do
3402                  {
3403                  cout<<endl;
3404                  resposta=Menu.menu_exibir_voltar();
3405                  if(resposta==0)break;
3406                      do
3407                      {
3408                      limpa_janela();
3409                      Head.header_titulo();
3410                      Head.header_banco_de_dados_exibir_amostras();
3411                      amostras=bancodedados.ler_amostra_basico();
3412                      bancodedados.listar_amostra(amostras);
3413                      amostras.clear();
3414                      cout<<"CHAVE:␣";
3415                      cin>>int_chave;
3416                      //limpa tudo e reescreve
3417                      limpa_janela();
3418                      Head.header_titulo();
3419                      Head.header_banco_de_dados_exibir_amostras();
3420                      amostras=bancodedados.ler_amostra_basico();
3421                      bancodedados.listar_amostra(amostras);
3422                      amostras.clear();
3423                      bancodedados.exibir_amostra(int_chave);
3424                      cout<<endl;
3425                      cout<<"Exibir␣Outra␣Amostra?";
3426                      resposta=Menu.menu_sim_nao();
3427                      }while(resposta!=0);
3428                  }while(resposta!=0);
3429                //exibe lista pergunta chave para visualizaca
3430
3431                resposta=2;
3432                break;
3433
3434              case 2://pesquisar
3435                limpa_janela();
3436                Head.header_titulo();
```

```
3437                Head.header_banco_de_dados_pesquisa_de_amostras();
3438                resposta=Menu.menu_banco_de_dados_pesquisar();
3439                if(resposta==2)
3440                {
3441                  resposta=1;
3442                  break;
3443                }
3444
3445                switch(resposta)
3446                {
3447
3448                  case 1:
3449
3450                    do
3451                    {
3452                      limpa_janela();
3453                      Head.header_banco_de_dados_pesquisa_de_amostras();
3454                      amostras=bancodedados.ler_amostra_basico();//lista
                             todas as amostras
3455                      cout<<"Itens de Pesquisa"<<endl<<endl;
3456                      resposta=Menu.menu_filtro();//escreve menu filtro e
                             retorna seu valor
3457                      if(resposta==16)break;//volta
3458                      limpa_janela();
3459                      Head.header_banco_de_dados_pesquisa_de_amostras();
3460                      amostras=bancodedados.ler_amostra_basico(resposta);
                             //vetor de amostra recebe amostra ja filtradas
3461                      bancodedados.listar_amostra(amostras);
3462                      cout<<endl;
3463                      resposta=Menu.menu_exibir_voltar();
3464                      if(resposta==0)break;
3465
3466
3467                      do
3468                      {
3469                        //limpa tudo e reescreve
3470                        limpa_janela();
3471                        Head.header_titulo();
3472                        Head.header_banco_de_dados_exibir_amostras();
3473                        bancodedados.listar_amostra(amostras);
3474                        cout<<endl;
3475                        resposta=Menu.menu_exibir_voltar();
3476                        if(resposta==0)break;
3477                        //inicia exibicao amostra
3478                        cout<<"CHAVE: ";
3479                        cin>>int_chave;
3480                        limpa_janela();//acha a chave digitada e reescreve
                               tudo novamente
```

```
3481                    Head.header_titulo();
3482                    Head.header_banco_de_dados_exibir_amostras();
3483                    bancodedados.listar_amostra(amostras);
3484                    bancodedados.exibir_amostra(int_chave);
3485                    cout<<"Deseja␣Pesquisar␣Novamente?";
3486                    resposta=Menu.menu_sim_nao();
3487                    amostras.clear();
3488
3489                  }
3490                  while(resposta==1);
3491                }
3492                while(resposta==1);
3493                break;
3494            }
3495            resposta=2;
3496            break;//break da pesquisa
3497
3498            case 3://inserir
3499              limpa_janela();
3500              Head.header_titulo();
3501              Head.header_banco_de_dados_inserir_amostras();
3502              if(configuracao.opcao_abrir_DT()==1)
3503              {
3504                abre_DT();
3505              }
3506
3507              do
3508              {
3509                bancodedados.inserir_amostra();
3510                cout<<endl;
3511                cout<<endl;
3512                cout<<"Deseja␣Inserir␣Novamente?"<<endl;
3513                resposta=Menu.menu_sim_nao();
3514              }
3515              while(resposta==1);
3516          resposta=2;//retorna ao menu Banco de DADOS
3517
3518          break;
3519
3520             case 4://excluir
3521               do
3522               {
3523                 limpa_janela();
3524                 Head.header_titulo();
3525                 Head.header_banco_de_dados_excluir_amostras();
3526                 amostras=bancodedados.ler_amostra_basico();
3527                 bancodedados.listar_amostra(amostras);
3528                 cout<<"Deseja␣realmente␣Excluir?";
```

```
3529                    resposta=Menu.menu_sim_nao();
3530                    if(resposta==0)break;
3531                    cout<<"CHAVE:␣";
3532                    while(!(cin>>int_chave));//correcao de erro caso
                            entre alguma resposta errada
3533                    {
3534                      cin.clear() ;
3535                      cin >> ch;
3536                      cin.get();
3537                    }
3538
3539
3540                    cin>>int_chave;
3541                    cin.get();
3542                    system("pause");
3543                    bancodedados.excluir_amostra(int_chave);
3544                    amostras.clear();
3545                    cout<<endl;
3546                    cout<<endl;
3547
3548                    //atualiza tela
3549                    limpa_janela();
3550                    Head.header_titulo();
3551                    Head.header_banco_de_dados_excluir_amostras();
3552                    amostras=bancodedados.ler_amostra_basico();
3553                    bancodedados.listar_amostra(amostras);
3554                    amostras.clear();
3555                    cout<<"Deseja␣Excluir␣Outra␣Amostra?"<<endl;
3556                    resposta=Menu.menu_sim_nao();
3557                  }
3558                  while(resposta==1);
3559                  resposta=2;//retorna ao menu Banco de DADOS
3560
3561
3562                  break;
3563
3564
3565              case 5://Exportar
3566                limpa_janela();
3567                Head.header_titulo();
3568                Head.header_banco_de_dados_exportar_amostras();
3569                resposta=Menu.menu_exportar_dados();
3570                if(resposta==5)
3571                {
3572                  resposta=2;
3573                  break;
3574                }
3575                switch(resposta)
```

```
3576                {
3577                  case 1:
3578
3579                    cout<<"CHAVE:␣";
3580                    cin>>int_chave;
3581                    cin.get();
3582                    relatorio.exportar_amostra(int_chave);
3583                    abre_relatorio(int_chave);
3584                    break;
3585
3586                  default:
3587                    resposta=2;
3588                    break;
3589                    //abre diretorio com relatorio(os)
3590                }
3591                resposta=2;
3592                break;
3593
3594                case 6://voltar
3595                  resposta=1;
3596                  break;
3597
3598                   default:
3599                  {
3600                    #ifdef __linux
3601                    setlocale(LC_ALL, "Portuguese");
3602                    cout<<"Op\u00E7\u00E3o␣invalida."<<endl;
3603                    locale::global(locale(""));
3604                    #elif _WIN32
3605                    setlocale(LC_ALL, "Portuguese");
3606                    cout<<"Opcão␣invalida."<<endl;
3607                    locale::global(locale(""));
3608                    #else
3609
3610                    #endif
3611
3612                    limpa_janela();
3613
3614                  }
3615                  resposta=1;
3616                  break;
3617                 }
3618          }
3619          while(resposta==2);
3620
3621          break;//fechando o caso 1:Banco de dados
3622
3623
```

```
3624          case 2://configuracoes
3625            limpa_janela();
3626            Head.header_titulo();
3627            Head.header_configuracoes();
3628            resposta=Menu.menu_configuracoes();
3629
3630            switch(resposta)
3631            {
3632              case 1:
3633                cout<<endl;
3634                cout<<"Explorador␣Padr\u00E3o:␣"<<configuracao.
                       opcao_programa("explorador")<<endl;
3635                cout<<"Explorador␣Padr\u00E3o:␣";
3636                cin>>informacao_str;
3637                cin.get();
3638                configuracao.opcao_programa("explorador",informacao_str);
3639                break;
3640
3641              case 2:
3642                cout<<endl;
3643                cout<<"Editor␣de␣Texto␣Padr\u00E3o:␣"<<configuracao.
                       opcao_programa("editortexto")<<endl;
3644                cout<<"Editor␣de␣Texto␣Padr\uE003o:␣";
3645                cin>>informacao_str;
3646                cin.get();
3647                configuracao.opcao_programa("editortexto",informacao_str);
3648                break;
3649
3650              case 3:
3651                cout<<endl;
3652                cout<<"Op\u00E7\u00E3o␣Atual:␣"<<configuracao.
                       opcao_abrir_DT()<<endl;
3653                cout<<endl;
3654                cout<<"Deseja␣sempre␣abrir␣o␣Diret\u00F3rio␣de␣Trabalho␣(
                       DT)?"<<endl;
3655                resposta=Menu.menu_sim_nao();
3656                configuracao.opcao_abrir_DT(resposta);//0 para nao abrir e
                        1 para abrir
3657                break;
3658
3659              case 4:
3660                resposta=1;
3661                break;
3662
3663
3664              default:
3665                resposta=1;
3666                break;
```

```
3667            }
3668
3669            break;
3670
3671
3672          case 3://abrir diretorio de trabalho
3673            limpa_janela();
3674
3675            Head.header_menu_abre_DT();
3676            //verifica sistema operacional
3677            #ifdef __linux
3678            //sleep(100);
3679            #elif _WIN32
3680            Sleep(100);
3681            #else
3682
3683            #endif
3684
3685            abre_DT();
3686            resposta=1;
3687
3688            break;
3689
3690
3691          case 4://abrir diretorio de dados
3692            limpa_janela();
3693
3694            Head.header_menu_abre_bd();
3695            //verifica sistema operacional
3696            #ifdef __linux
3697            //sleep(100);
3698            #elif _WIN32
3699            Sleep(100);
3700            #else
3701
3702            #endif
3703
3704            abre_bd();
3705            resposta=1;
3706
3707
3708            break;
3709
3710
3711           case 5://sair
3712             limpa_janela();
3713             Head.header_titulo();
3714             cout<<"Realmente␣deseja␣sair?"<<endl;
```

```
3715            resposta=Menu.menu_sim_nao();
3716            if(resposta==1)
3717              exit(0);
3718            else
3719              resposta=1;
3720            break;
3721
3722
3723          default:
3724
3725            //verifica sistema operacional
3726            #ifdef __linux
3727            setlocale(LC_ALL, "Portuguese");
3728            cout<<"Op\u00E7\u00E3o␣inv\u00E1lida."<<endl;
3729            locale::global(locale(""));
3730            // sleep(1000);
3731            #elif _WIN32
3732            setlocale(LC_ALL, "Portuguese");
3733            cout<<"Opção␣inválida."<<endl;
3734            locale::global(locale(""));
3735            Sleep(1000);
3736            #else
3737
3738            #endif
3739
3740            resposta=1;
3741            break;
3742
3743     }
3744   }
3745   while(resposta==1);
3746 }
3747
3748
3749 void CInterface::abre_DT()
3750 {
3751   //inicia diretorio de trabalho
3752   //comando: "move bd\0.txt bd\lixo" Comando para mover o arquivo da
          amostra
3753   //verifica sistema operacional
3754   programa=configuracao.opcao_programa("explorador");//define
          explorardor
3755   comando_abrir.str("");
3756   #ifdef __linux
3757   comando_abrir<<programa<<"␣"<<"./DT";
3758   #elif _WIN32
3759   comando_abrir<<"start"<<"␣"<<programa<<"␣"<<"DT";
3760
```

```
3761   #else
3762
3763   #endif
3764   system(comando_abrir.str().c_str());//Manda o comando pro prompt para
           mover o arquivo p lixeira
3765   comando_abrir.str("");
3766 }
3767
3768 void CInterface::abre_relatorio(int int_chave_)
3769 {
3770   //inicia diretorio de trabalho
3771   //comando: "move bd\0.txt bd\lixo" Comando para mover o arquivo da
           amostra
3772   //verifica sistema operacional
3773   programa=configuracao.opcao_programa("explorador");//define explorador
3774   int_chave=int_chave_;
3775   comando_abrir.str("");
3776   #ifdef __linux
3777   comando_abrir<<programa<<"␣"<<"./DT/relatorios/"<<int_chave;
3778   #elif _WIN32
3779   comando_abrir<<"start"<<"␣"<<programa<<"␣"<<"DT"<<char(92)<<"
           relatorios"<<char(92)<<int_chave;
3780   cout<<comando_abrir.str();
3781   #else
3782
3783   #endif
3784   if(configuracao.opcao_abrir_DT()==1)
3785   {
3786     system(comando_abrir.str().c_str());//Manda o comando pro prompt
           para mover o arquivo p lixeira
3787   }
3788
3789   comando_abrir.str("");
3790 }
3791
3792 void CInterface::abre_bd()
3793 {
3794   //inicia diretorio de dados
3795   //comando: "move bd\0.txt bd\lixo" Comando para mover o arquivo da
           amostra
3796   programa=configuracao.opcao_programa("explorador");//define explorador
3797   //verifica sistema operacional
3798   comando_abrir.str("");
3799   #ifdef __linux
3800   comando_abrir<<programa<<"␣"<<"./bd";
3801   #elif _WIN32
3802   comando_abrir<<"start"<<"␣"<<programa<<"␣"<<"bd";
3803
```

```
3804   #else
3805
3806   #endif
3807   system(comando_abrir.str().c_str());//Manda o comando pro prompt para
           mover o arquivo p lixeira
3808   comando_abrir.str("");
3809 }
3810
3811 void CInterface::ajusta_janela()
3812 {
3813   //verifica sistema operacional
3814   #ifdef __linux
3815   //sleep(100);
3816   #elif _WIN32
3817   //ajusta cor do programa(comando de MS-DOS)
3818   system("color F0");
3819   //Maximiza Programa
3820   {//Referencia(http://forum.clubedohardware.com.br/full-screen-c/567207
         em 15/11/2013)
3821     keybd_event ( VK_MENU, 0x38, 0, 0 );
3822     keybd_event ( VK_SPACE, 0x39, 0, 0 );
3823     keybd_event(0x58,0,0,0);
3824     keybd_event ( VK_MENU, 0x38, KEYEVENTF_KEYUP, 0 );
3825     keybd_event ( VK_SPACE, 0x39, KEYEVENTF_KEYUP, 0 );
3826     keybd_event(0x58,0,KEYEVENTF_KEYUP,0);
3827   }
3828   //Chama Animacao de carregamento
3829   CHeader Header;
3830   Header.header_carregando();
3831   //Escreve Versao do Windows
3832   cout<<"Versao do sistema: ";
3833   system("VER");
3834   Sleep(100);
3835   #else
3836
3837   #endif
3838
3839 }
3840
3841 void CInterface::limpa_janela()
3842 {
3843   #ifdef __linux
3844   system("clear");//limpa tela
3845   #elif _WIN32
3846   system("cls");
3847   #else
3848
3849   #endif
```

```
}
```

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

Listing 6.11: Arquivo de cabeçalho da classe CMenu.

```
#ifndef CMenu_h
#define CMenu_h

#include <iostream>
#ifdef __linux

    #include <cstdlib>

#elif _WIN32
        #include <windows.h>//para configurar a janela
        #include <stdlib.h>//para system
#else

#endif

#include <string>
#include <sstream>
#include <vector>

using namespace std;

    class CMenu
                {
                        public:
                             int resposta;
                             string resposta_str;
                                char ch;
                        public:
                            int menu_inicio();
                            int menu_banco_de_dados();
                int menu_exportar_dados();
                int menu_banco_de_dados_pesquisar();
                int menu_filtro();
                int menu_configuracoes();
                            int menu_sim_nao();
                            int menu_exibir_voltar();

                //menus ocultos
                int menu_usuario();
                //sobrecarga_operador
                };
```

```
3896
3897 #endif
```

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

Listing 6.12: Arquivo de implementação da classe CMenu.

```
3898 //verifica sistema operacional
3899         #ifdef __linux
3900         //#include <unistd.h>
3901         #include <cstdlib>//para system
3902         #elif _WIN32
3903
3904         #include <windows.h>
3905         #include <stdlib.h>//para system
3906         #include <conio.h>
3907
3908         #else
3909
3910         #endif
3911 #include <iostream>
3912 #include <string>
3913 #include <sstream>
3914 #include <locale>
3915
3916 #include <vector>
3917 #include "CMenu.h"
3918 #include <stdio.h>
3919
3920
3921
3922 using namespace std;
3923
3924 int CMenu::menu_inicio()
3925 {
3926
3927   #ifdef __linux
3928   cout<<"(1)Banco de Dados"<<endl;
3929   cout<<"(2)Configura\u00E7\u00F5es"<<endl;
3930   cout<<"(3)Abrir Diret\u00F3rio de Trabalho"<<endl;
3931   cout<<"(4)Abrir Diret\u00F3rio de Dados"<<endl;
3932   cout<<"(5)SAIR"<<endl;
3933   cout<<"Informe a Op\u00E7\u00E3o: ";
3934   #elif _WIN32
3935   cout<<"(1)Banco de Dados"<<endl;
3936   cout<<"(2)Configurações"<<endl;
3937   cout<<"(3)Abrir Diretório de Trabalho"<<endl;
3938   cout<<"(4)Abrir Diretório de Dados"<<endl;
3939   cout<<"(5)SAIR"<<endl;
3940   cout<<"Informe a Opção: ";
```

```
3941  #else
3942
3943  #endif
3944
3945  cin>>resposta_str;
3946  cin.get();
3947  //testa se foi digitado um numero maior ou menor
3948  if(atoi(resposta_str.c_str()))
3949  {
3950    if(atoi(resposta_str.c_str())<1 || atoi(resposta_str.c_str()) >4)
3951    {
3952      resposta=5;//se resposta for fora do intervalo, define 1 como
           padrao
3953    }
3954    else
3955    {
3956      resposta=atoi(resposta_str.c_str());
3957    }
3958
3959  }
3960  else
3961  {
3962    resposta=5;//Permanece no MENU
3963  }
3964  return resposta;
3965}
3966
3967
3968int CMenu::menu_banco_de_dados()
3969{
3970  #ifdef __linux
3971
3972  cout<<"(1)Listar"<<endl;
3973  cout<<"(2)Pesquisar"<<endl;
3974  cout<<"(3)Inserir"<<endl;
3975  cout<<"(4)Excluir"<<endl;
3976  cout<<"(5)Exportar"<<endl;
3977  cout<<"(6)VOLTAR"<<endl;
3978  //cout<<"(8)SAIR"<<endl;
3979  cout<<endl;
3980  cout<<"Infome␣a␣Op\u00E7\u00E3o:␣";
3981
3982  #elif _WIN32
3983
3984  cout<<"(1)Listar"<<endl;
3985  cout<<"(2)Pesquisar"<<endl;
3986  cout<<"(3)Inserir"<<endl;
3987  cout<<"(4)Excluir"<<endl;
```

```
3988   cout<<"(5)Exportar"<<endl;
3989   cout<<"(6)VOLTAR"<<endl;
3990   //cout<<"(8)SAIR"<<endl;
3991   cout<<endl;
3992   cout<<"Infome a Opção: ";
3993
3994   #else
3995
3996   #endif
3997
3998   while(!(cin>>resposta))//correcao de erro caso entre alguma resposta
           errada
3999   {
4000     cin.clear() ;
4001     cin >> ch;
4002     if(ch==120)break;//se caracter for x termina processo
4003   }
4004   cin.get();
4005
4006   if(resposta<1 || resposta >7)
4007   {
4008     resposta=7;//se resposta for fora do intervalo, define 1 como padrao
4009   }
4010
4011   return resposta;
4012 }
4013
4014 int CMenu::menu_banco_de_dados_pesquisar()
4015
4016 {
4017   cout<<"(1)Pesquisa Simples"<<endl;
4018   cout<<"(2)VOLTAR"<<endl;
4019   cout<<endl;
4020
4021
4022   #ifdef __linux
4023   cout<<"Infome a Op\u00E7\u00E3o: ";
4024   #elif _WIN32
4025
4026   cout<<"Infome a Opção: ";
4027   #else
4028
4029   #endif
4030
4031   while(!(cin>>resposta))//correcao de erro caso entre alguma resposta
           errada
4032   {
4033     cin.clear() ;
```

```
4034      cin >> ch;
4035      if(ch==120)break;//se caracter for x termina processo
4036    }
4037    cin.get();
4038
4039    if(resposta<1 || resposta >2)
4040    {
4041      resposta=2;//se resposta for fora do intervalo, define 1 como padrao
4042    }
4043
4044    return resposta;
4045  }
4046
4047  int CMenu::menu_configuracoes()
4048  {
4049    #ifdef __linux
4050    cout<<"(1)Explorador"<<endl;
4051    cout<<"(2)Editor de Texto"<<endl;
4052    cout<<"(3)Op\u00E7\u00E3o Abrir Diret\u00F3rio de Trabalho (DT)"<<endl
         ;
4053    cout<<endl;
4054    cout<<"(4)Voltar"<<endl;
4055    cout<<endl;
4056    cout<<"Infome a Op\u00E7\u00E3o: ";
4057    #elif _WIN32
4058    cout<<"(1)Explorador"<<endl;
4059    cout<<"(2)Editor de Texto"<<endl;
4060    cout<<"(3)Opção Abrir Diretorio de Trabalho (DT)"<<endl;
4061    cout<<endl;
4062    cout<<"(4)Voltar"<<endl;
4063    cout<<endl;
4064    cout<<"Infome a Opção: ";
4065    #else
4066
4067    #endif
4068
4069    while(!(cin>>resposta))//correcao de erro caso entre alguma resposta
         errada
4070    {
4071      cin.clear() ;
4072      cin >> ch;
4073      if(ch==120)break;//se caracter for x termina processo
4074    }
4075    cin.get();
4076    if(resposta<0 || resposta >4)
4077    {
4078      resposta=4;//se resposta for fora do intervalo, define 1 como padrao
4079    }
```

```
4080   return resposta;
4081 }
4082
4083 int CMenu::menu_filtro()
4084 {
4085   #ifdef __linux
4086
4087   cout<<"(-)Chave"<<endl;
4088   cout<<"(2)Nome␣do␣fluido"<<endl;
4089   cout<<"(3)Base"<<endl;
4090   cout<<"(4)pH␣m\u00EDnimo"<<endl;
4091   cout<<"(5)pH␣m\u00C1ximo"<<endl;
4092   cout<<"(6)Teor␣de␣\u00C1gua"<<endl;
4093   cout<<"(7)Peso␣espec\u00EDfico␣m\u00EDnimo"<<endl;
4094   cout<<"(8)Peso␣espec\u00EDfico␣m\u00C1ximo"<<endl;
4095   cout<<"(9)Temperatura␣de␣envelhecimento"<<endl;
4096   cout<<"(10)Salinidade"<<endl;
4097   cout<<"(11)Filtrado"<<endl;
4098   cout<<"|DATA␣DE␣CADASTRO|"<<endl;
4099   cout<<"(12)Dia"<<"␣␣␣␣␣␣␣␣␣␣␣(13)M\u00EAs"<<"␣␣␣␣␣␣␣␣␣(14)Ano"<<endl;
4100   cout<<"(15)Listar␣todos"<<endl;
4101   cout<<"(16)VOLTAR"<<endl;
4102   cout<<endl;
4103   cout<<"Infome␣a␣Op\u00E7\u00E3o:␣";
4104
4105
4106   #elif _WIN32
4107
4108   cout<<"(-)Chave"<<endl;
4109   cout<<"(2)Nome␣do␣fluido"<<endl;
4110   cout<<"(3)Base"<<endl;
4111   cout<<"(4)pH␣m\u00EDnimo"<<endl;
4112   cout<<"(5)pH␣m\u00C1ximo"<<endl;
4113   cout<<"(6)Teor␣de␣\u00C1gua"<<endl;
4114   cout<<"(7)Peso␣espec\u00EDfico␣m\u00EDnimo"<<endl;
4115   cout<<"(8)Peso␣espec\u00EDfico␣m\u00C1ximo"<<endl;
4116   cout<<"(9)Temperatura␣de␣envelhecimento"<<endl;
4117   cout<<"(10)Salinidade"<<endl;
4118   cout<<"(11)Filtrado"<<endl;
4119   cout<<"|DATA␣DE␣CADASTRO|"<<endl;
4120   cout<<"(12)Dia"<<"␣␣␣␣␣␣␣␣␣␣␣(13)M\u00EAs"<<"␣␣␣␣␣␣␣␣␣(14)Ano"<<endl;
4121   cout<<"(15)Listar␣todos"<<endl;
4122   cout<<"(16)VOLTAR"<<endl;
4123   cout<<endl;
4124   cout<<"Infome␣a␣Op\u00E7\u00E3o:␣";
4125
4126   #else
4127
```

```
4128   #endif
4129
4130   while(!(cin>>resposta))//correcao de erro caso entre alguma resposta
           errada
4131   {
4132     cin.clear() ;
4133     cin >> ch;
4134     if(ch==120)break;//se caracter for x termina processo
4135   }
4136   cin.get();
4137
4138   if(resposta<1 || resposta >16)
4139   {
4140     resposta=16;
4141   }
4142
4143   return resposta;
4144 }
4145
4146
4147 int CMenu::menu_sim_nao()
4148 {
4149
4150 #ifdef __linux
4151
4152     cout<<endl;
4153
4154     setlocale(LC_ALL, "Portuguese");
4155     cout<<"(1)Sim"<<endl;
4156     cout<<"(0)N\u00E3o"<<endl;
4157     cout<<endl;
4158     cout<<"Infome␣a␣Op\u00E7\u00E3o:␣";
4159     locale::global(locale(""));
4160
4161 #elif _WIN32
4162     setlocale(LC_ALL, "Portuguese");
4163     cout<<endl;
4164     setlocale(LC_ALL, "Portuguese");
4165     cout<<"(1)Sim"<<endl;
4166     cout<<"(0)Não"<<endl;
4167     cout<<endl;
4168     cout<<"Infome␣a␣Opção:␣";
4169     locale::global(locale(""));
4170
4171 #else
4172
4173 #endif
4174
```

```
4175     while(!(cin>>resposta))//correcao de erro caso entre alguma resposta
             errada
4176     {
4177       cin.clear() ;
4178       cin >> ch;
4179       if(ch==120)break;//se caracter for x termina processo
4180     }
4181     cin.get();
4182 
4183     if(resposta<0 || resposta >1)
4184     {
4185       resposta=0;//se resposta for fora do intervalo, define 0 como
               padrao
4186     }
4187 
4188     return resposta;
4189 }
4190 
4191 int CMenu::menu_exibir_voltar()
4192 {
4193 
4194   cout<<endl;
4195   cout<<"(0)Voltar"<<endl;
4196   cout<<"(1)Exibir"<<endl;
4197 
4198 
4199   cout<<endl;
4200 
4201   #ifdef __linux
4202   setlocale(LC_ALL, "Portuguese");
4203   cout<<"Infome␣a␣Op\u00E7\u00E3o:␣";
4204   locale::global(locale(""));
4205   #elif _WIN32
4206   setlocale(LC_ALL, "Portuguese");
4207   cout<<"Infome␣a␣Opção:␣";
4208   locale::global(locale(""));
4209   #else
4210 
4211   #endif
4212 
4213 
4214   while(!(cin>>resposta))//correcao de erro caso entre alguma resposta
         errada
4215   {
4216     cin.clear() ;
4217     cin >> ch;
4218     if(ch==120)break;//se caracter for x termina processo
4219   }
```

```
4220   cin.get();
4221
4222   if(resposta<0 || resposta >1)
4223   {
4224     resposta=0;//se resposta for fora do intervalo, define 0 como padrao
4225   }
4226
4227   return resposta;
4228 }
4229
4230 int CMenu::menu_exportar_dados()
4231 {
4232
4233 #ifdef __linux
4234     setlocale(LC_ALL, "Portuguese");
4235     cout<<"(1)Completo"<<endl;
4236     cout<<"(2)VOLTAR"<<endl;
4237     cout<<endl;
4238     cout<<"Infome␣a␣Op\u00E7\u00E3o:␣";
4239
4240     locale::global(locale(""));
4241 #elif _WIN32
4242     setlocale(LC_ALL, "Portuguese");
4243     cout<<"(1)Completo"<<endl;
4244     cout<<"(2)VOLTAR"<<endl;
4245     cout<<endl;
4246     cout<<"Infome␣a␣Opção:␣";
4247
4248     locale::global(locale(""));
4249 #else
4250
4251 #endif
4252
4253     while(!(cin>>resposta))//correcao de erro caso entre alguma resposta
          errada
4254     {
4255       cin.clear() ;
4256       cin >> ch;
4257       if(ch==120)break;//se caracter for x termina processo
4258     }
4259     cin.get();
4260     if(resposta<1 || resposta >3)
4261     {
4262       resposta=3;//se resposta for fora do intervalo, define 3 como
            padrao
4263     }
4264     return resposta;
4265 }
```

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

Listing 6.13: Arquivo de cabeçalho da classe CRelatorio.

```
4267 #ifndef CRelatorio_h
4268 #define CRelatorio_h
4269 #include <string>
4270 #include <vector>
4271 #include <iostream>
4272 #include <fstream>
4273
4274 #include "CFluidodePerfuracao.h"
4275 #include "CBancodeDados.h"
4276
4277 using namespace std;
4278
4279 class CRelatorio
4280 {
4281
4282 public:
4283
4284   int chave;
4285   int int_chave;
4286   int resposta;
4287   unsigned int c;
4288
4289   vector<CFluidodePerfuracao> v_amostra;
4290
4291   ofstream fout;
4292
4293   stringstream ss;
4294
4295   CBancodeDados lerbasico;
4296
4297 public:
4298   void exportar_amostra(int);
4299 };
4300
4301 #endif
```

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

Listing 6.14: Arquivo de implementação da classe CRelatorio.

```
4302 #include <iostream>
4303 #include <stdio.h>
4304 #include <fstream>
4305 #include <string>
4306
4307 //verifica sistema operacional
4308 #ifdef __linux
```

```
4309
4310     #include <cstdlib>
4311
4312 #elif _WIN32
4313
4314     #include <windows.h>
4315     #include <stdlib.h>//para system
4316     #include <conio.h>
4317
4318 #else
4319
4320 #endif
4321
4322
4323 #include "CRelatorio.h"
4324 #include "CFluidodePerfuracao.h"
4325 #include "CBancodeDados.h"
4326
4327 void CRelatorio::exportar_amostra(int chave_)
4328 {
4329   chave=chave_;
4330
4331   //criando o diretorio para salvar o relatorio de chave n
4332   #ifdef __linux
4333   ss<<"mkdir "<<"DT"<<char(47)<<"relatorios"<<char(47)<<chave;//DT/
         relatorios/1
4334   #elif _WIN32
4335   ss<<"mkdir "<<"DT"<<char(92)<<"relatorios"<<char(92)<<chave;//DT\
         relatorios\1
4336   #else
4337
4338   #endif
4339
4340   system(ss.str().c_str());//cria o diretorio a ser gravado os arquivos
4341   ss.str("");//apaga stringstream
4342
4343   ////////////////////////////////////////
4344   /////inicia leitura basica das amostras
4345   ////////////////////////////////////////
4346
4347   //pesquisa amostra de chave n
4348   int_chave=chave_;
4349   CFluidodePerfuracao amostra_aux;
4350   vector<CFluidodePerfuracao> v_amostra;
4351   unsigned int c;
4352   //lendo todas as amostras basico
4353   v_amostra=lerbasico.ler_amostra_basico();
4354   //procurando amostra correspontende a chave informada
```

```
for(c=0;c<v_amostra.size();c++)
{
  if (int_chave==v_amostra[c].chave)
  {
    amostra_aux=v_amostra[c];
    break;
  }
}
///////////////////////////////////////
/////Termina leitura Basica das amostras
///////////////////////////////////////
ss.str("");//apaga stringstream

#ifdef __linux
ss<<"DT"<<char(47)<<"relatorios"<<char(47)<<chave<<char(47)<<"Amostra-"<<chave<<"-completa.txt";//
#elif _WIN32
ss<<"DT"<<char(92)<<"relatorios"<<char(92)<<chave<<char(92)<<"Amostra-"<<chave<<"-completa.txt";
#else

#endif

fout.open(ss.str().c_str());//abre arquivo para escrita
ss.str("");
//////////////////////////////////////////////////////////////////////////

////////////INICIO DA ESCRITA DO CODIGO
//////////////////////////////////////////////////////////////////////////


//inicia das exibicoes basica das amostras
fout<<"_______________________________________________"<<endl;
fout<<"␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣"<<endl;
fout<<"␣␣␣␣␣␣Relatório␣da␣Amostra␣de␣Chave␣"<<int_chave<<"␣␣␣␣␣"<<endl;
fout<<"_______________________________________________"<<endl;
fout<<endl<<endl;
fout<<"Data␣de␣Cadastro␣----␣:␣"<<amostra_aux.dia<<"/"<<amostra_aux.mes<<"/"<<amostra_aux.ano<<endl;
fout<<"Nome␣do␣Fluido␣-------:␣"<<amostra_aux.nome_do_fluido<<endl;
fout<<"Base␣----------------␣:␣"<<amostra_aux.base<<endl;
fout<<"Teor␣de␣Base␣--------␣:␣"<<lerbasico.teste_vazio(amostra_aux.teorbase)<<endl;
fout<<"Teor␣de␣Agua␣--------␣:␣"<<lerbasico.teste_vazio(amostra_aux.teoragua)<<endl;
```

```
4393 fout<<"␣␣␣␣-----Propriedades␣Físicas␣e␣químicas-----␣␣␣␣␣␣"<<endl;
4394 fout<<"pH␣Mínimo␣----------␣:␣"<<lerbasico.teste_vazio(amostra_aux.
       pH_min)<<endl;
4395 fout<<"pH␣Máximo␣----------␣:␣"<<lerbasico.teste_vazio(amostra_aux.
       pH_max)<<endl;
4396 fout<<"Peso␣Específico␣Mínimo:␣"<<lerbasico.teste_vazio(amostra_aux.
       pe_min)<<endl;
4397 fout<<"Peso␣Específico␣Máximo:␣"<<lerbasico.teste_vazio(amostra_aux.
       pe_max)<<endl;
4398 fout<<"Temperatura␣de␣envelhecimento:␣"<<lerbasico.teste_vazio(
       amostra_aux.temp_e)<<endl;
4399 fout<<"Força␣gel␣inicial␣antes␣do␣envelhecimento:␣"<<lerbasico.
       teste_vazio(amostra_aux.gel_i_ae)<<endl;
4400 fout<<"Força␣gel␣inicial␣depois␣do␣envelhecimento:␣"<<lerbasico.
       teste_vazio(amostra_aux.gel_i_de)<<endl;
4401 fout<<"Força␣gel␣final␣antes␣do␣envelhecimento:␣"<<lerbasico.
       teste_vazio(amostra_aux.gel_f_ae)<<endl;
4402 fout<<"Força␣gel␣final␣depois␣do␣envelhecimento:␣"<<lerbasico.
       teste_vazio(amostra_aux.gel_f_de)<<endl;
4403 fout<<"Estabilidade␣elétrica␣antes␣do␣envelhecimento:␣"<<lerbasico.
       teste_vazio(amostra_aux.est_el_ae)<<endl;
4404 fout<<"Estabilidade␣elétrica␣depois␣do␣envelhecimento:␣"<<lerbasico.
       teste_vazio(amostra_aux.est_el_de)<<endl;
4405 fout<<"Coeficiente␣de␣lubricidade:␣"<<lerbasico.teste_vazio(
       amostra_aux.c_lubricidade)<<endl;
4406 fout<<"Volume␣de␣filtrado:␣"<<lerbasico.teste_vazio(amostra_aux.
       filtrado)<<endl;
4407 fout<<"Teor␣de␣sólidos␣-----␣:␣"<<lerbasico.teste_vazio(amostra_aux.
       teor_solidos)<<endl;
4408 fout<<"Salinidade␣----------␣:␣"<<lerbasico.teste_vazio(amostra_aux.
       temp_e)<<endl;
4409 fout<<"␣␣␣␣----Propriedades␣Reológicas----␣␣␣␣␣␣"<<endl;
4410 fout<<"Viscosidade␣Aparente␣antes␣do␣envelhecimento:␣"<<lerbasico.
       teste_vazio(amostra_aux.VA_ae)<<endl;
4411 fout<<"Viscosidade␣Aparente␣depois␣do␣envelhecimento:␣"<<lerbasico.
       teste_vazio(amostra_aux.VA_de)<<endl;
4412 fout<<"Viscosidade␣Plástica␣antes␣do␣envelhecimento:␣"<<lerbasico.
       teste_vazio(amostra_aux.VP_ae)<<endl;
4413 fout<<"Viscosidade␣Plástica␣depois␣do␣envelhecimento:␣"<<lerbasico.
       teste_vazio(amostra_aux.VP_de)<<endl;
4414 fout<<"␣␣␣␣----Aditivos----␣␣␣␣␣␣"<<endl;
4415 fout<<"Adensante␣----------␣:␣"<<amostra_aux.adensante<<endl;
4416 fout<<"Concentração␣de␣Adensante␣----------␣:␣"<<lerbasico.teste_vazio
       (amostra_aux.conc_adensante)<<endl;
4417 fout<<"Inibidor␣de␣Formações␣Ativas␣-------␣:␣"<<amostra_aux.
       inibidor_fa<<endl;
4418 fout<<"Concentração␣do␣Inibidor␣de␣Formações␣Ativas␣----------␣:␣"<<
       lerbasico.teste_vazio(amostra_aux.conc_inibidor_fa)<<endl;
```

```
4419  fout<<"Redutor␣de␣Filtrado␣-␣:␣"<<amostra_aux.redutor_f<<endl;
4420  fout<<"Concentração␣do␣Redutor␣de␣Filtrado␣----------␣:␣"<<lerbasico.
          teste_vazio(amostra_aux.conc_redutor_f)<<endl;
4421  fout<<"Biopolimero␣-␣:␣"<<amostra_aux.biopolimero<<endl;
4422  fout<<"Concentração␣do␣Biopolimero␣----------␣:␣"<<lerbasico.
          teste_vazio(amostra_aux.conc_biopolimero)<<endl;
4423  fout<<"Viscosificante␣-␣:␣"<<amostra_aux.viscosificante<<endl;
4424  fout<<"Concentração␣do␣Viscosificante----------␣:␣"<<lerbasico.
          teste_vazio(amostra_aux.conc_viscosificante)<<endl;
4425  fout<<"Dispersante␣-␣:␣"<<amostra_aux.dispersante<<endl;
4426  fout<<"Concentração␣do␣Dispersante----------␣:␣"<<lerbasico.
          teste_vazio(amostra_aux.conc_dispersante)<<endl;
4427  fout<<"Defloculante␣-␣:␣"<<amostra_aux.defloculante<<endl;
4428  fout<<"Concentração␣do␣Defloculante----------␣:␣"<<lerbasico.
          teste_vazio(amostra_aux.conc_defloculante)<<endl;
4429  fout<<"Emulsificante␣-␣:␣"<<amostra_aux.emulsificante<<endl;
4430  fout<<"Concentração␣do␣Emulsificante----------␣:␣"<<lerbasico.
          teste_vazio(amostra_aux.conc_emulsificante)<<endl;
4431  fout<<"Biocida␣-␣:␣"<<amostra_aux.biocida<<endl;
4432  fout<<"Concentração␣do␣Biocida----------␣:␣"<<lerbasico.teste_vazio(
          amostra_aux.conc_biocida)<<endl;
4433  fout<<"Lubrificante␣-␣:␣"<<amostra_aux.lubrificante<<endl;
4434  fout<<"Concentração␣do␣Lubrificante----------␣:␣"<<lerbasico.
          teste_vazio(amostra_aux.conc_lubrificante)<<endl;
4435  fout<<"Inibidor␣de␣corrosão␣-␣:␣"<<amostra_aux.inibidor_c<<endl;
4436  fout<<"Concentração␣do␣Inibidor␣de␣Corrosão----------␣:␣"<<lerbasico.
          teste_vazio(amostra_aux.conc_inibidor_c)<<endl;
4437  fout<<"Controlador␣de␣pH␣-␣:␣"<<amostra_aux.controlador_pH<<endl;
4438  fout<<"Concentração␣do␣Controlador␣de␣pH----------␣:␣"<<lerbasico.
          teste_vazio(amostra_aux.conc_controlador_pH)<<endl;
4439  fout<<endl<<endl;
4440
4441
4442  ////////////////////////////////////////////////////////
4443  ////////FIM DA ESCRITA DO CODIGO
4444  ////////////////////////////////////////////////////////
4445
4446  fout.close();
4447 }
```

Apresentam-se na listagem ?? o programa que usa as classes.

Listing 6.15: Arquivo de implementação da função `main()`.

```
4452 #include <iostream>
4453
4454     #ifdef __linux__
4455         #elif _WIN32
4456 #include <windows.h>
4457 #include <stdlib.h>//para system
```

```
4458          #else
4459          #endif
4460
4461 #include "CInterface.h"
4462
4463 using namespace std;
4464
4465 int main(int argc, char** argv)
4466 {
4467
4468   //Inicia a interface do programa
4469   CInterface Interface;
4470   Interface.inicia();
4471   return(0);
4472 }
```

Capítulo 7

Teste

Todo projeto de engenharia passa por uma etapa de testes. Neste capítulo apresentamos alguns testes do software desenvolvido.

7.1 Teste 1: Acesso ao Banco de Dados

No seguinte teste apresentado na imagem mostraremos o menu inicial, onde o usuário pode ter acesso ao Banco de Dados e às funções de Configurações, Abrir Diretório de Trabalho e Abrir Diretório de Dados através de uma interface em modo texto.

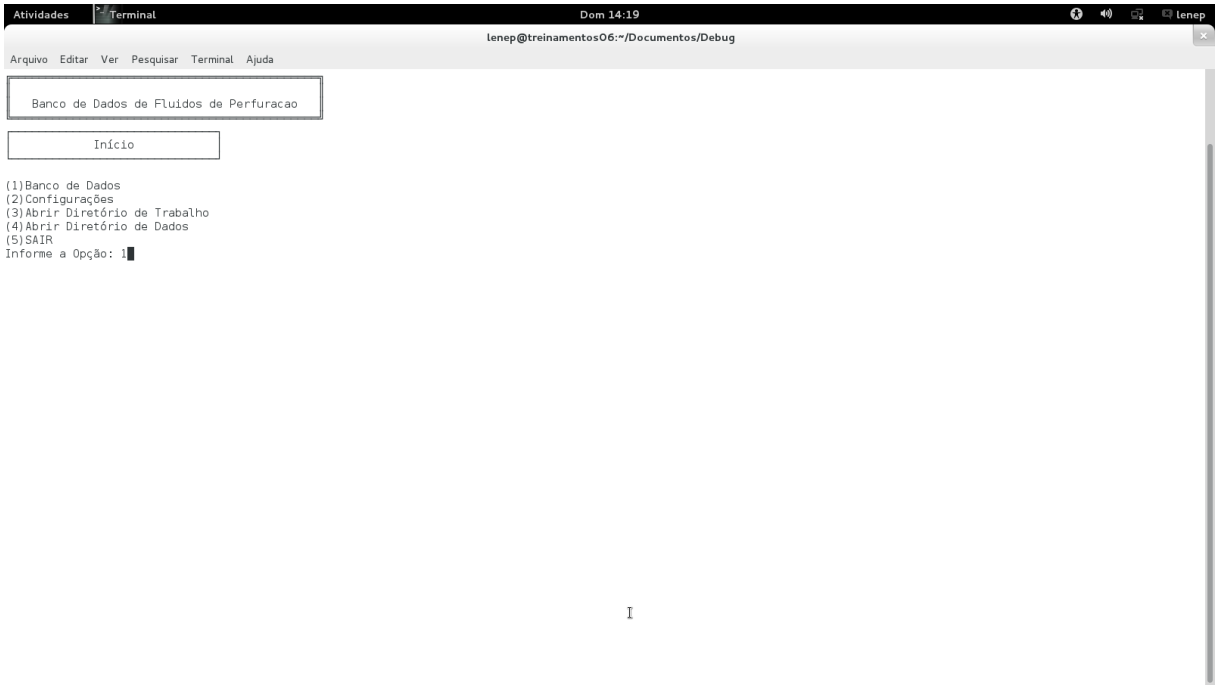

Figura 7.1: Tela do programa mostrando o menu inicial

Figura 7.2: Tela do programa mostrando o menu banco de dados

7.2 Teste 2: Listar fluidos de perfuração

Após acessar o Banco de Dados, o usuário terá acesso ao menu com diferentes opções. Após escolher listar os fluidos, o programa apresentará a seguinte tela mostrada na imagem abaixo.

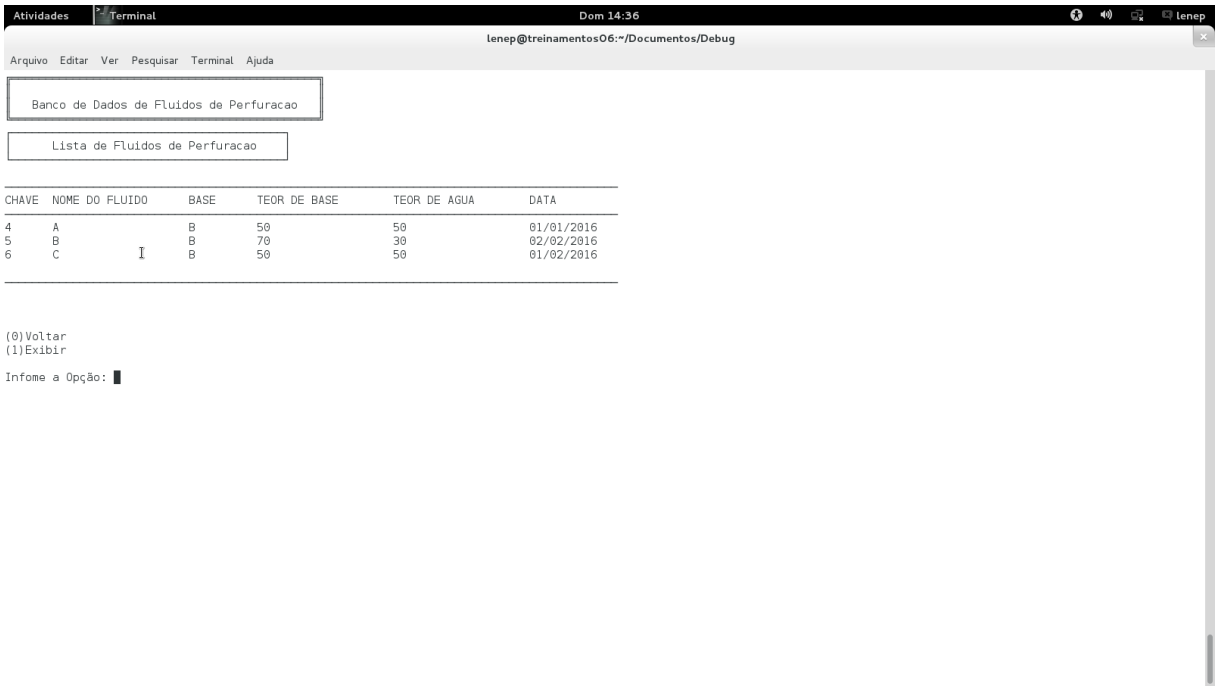

Figura 7.3: Tela do programa mostrando a lista de fluidos de perfuração

Após o programa ter exibido as amostras selecionadas, aparecerá para o usuário a

opção de exibir alguma amostra. Abaixo apresenta-se uma amostra exibida.

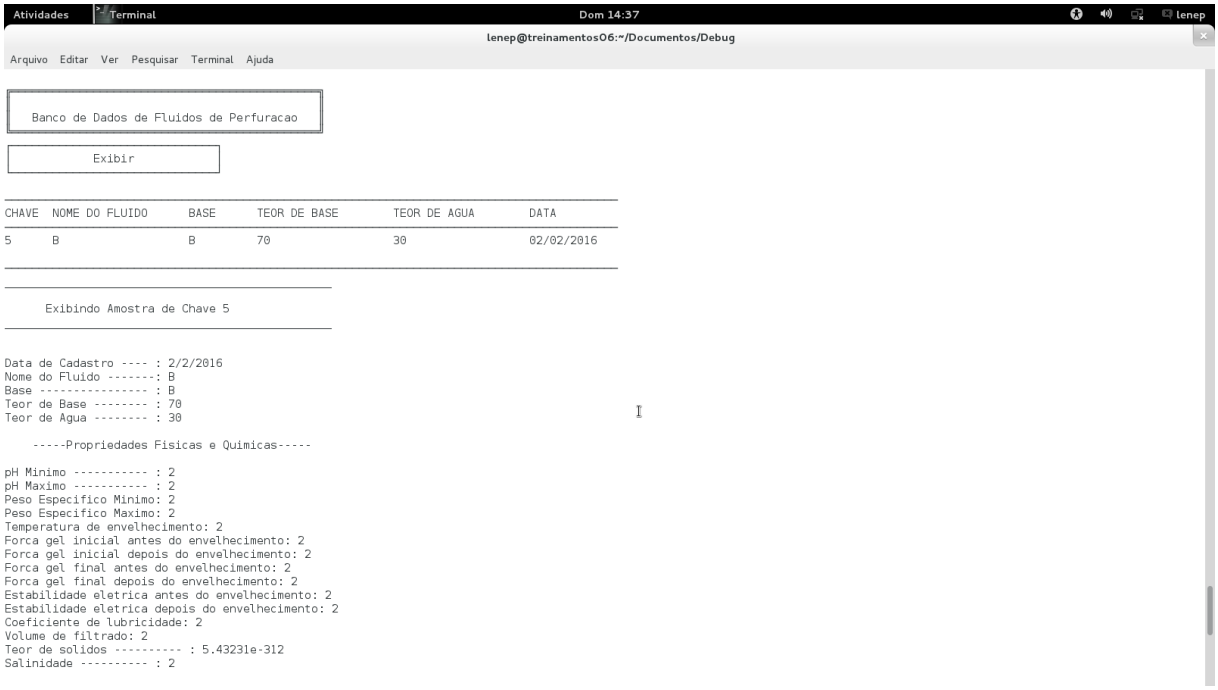

Figura 7.4: Tela do programa mostrando um fluido de perfuração sendo exibido

7.3 Teste 3: Pesquisar fluidos de perfuração

Após acessar o Banco de Dados, o usuário terá acesso ao menu com diferentes opções. Após escolher pesquisar os fluidos, o programa apresentará a seguinte tela mostrada na imagem abaixo. O usuário poderá escolher pesquisar os fluidos por diferentes tipos de filtros apresentados na captura de tela abaixo.

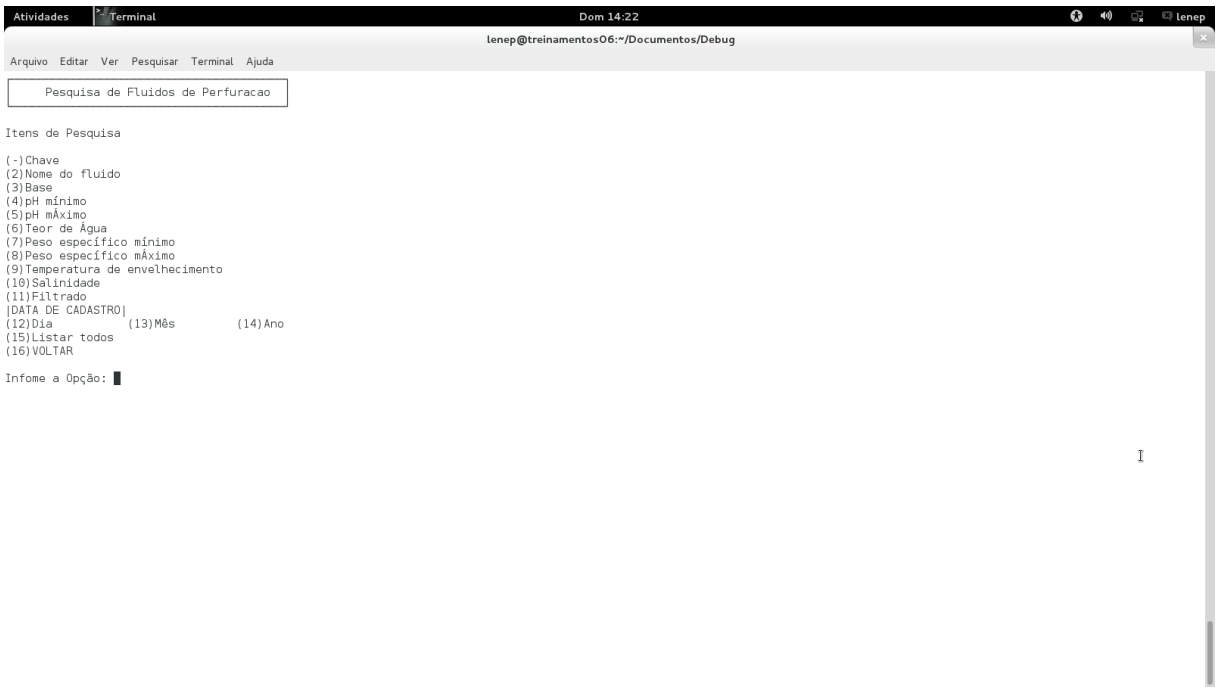

Figura 7.5: Tela do programa mostrando os filtros que podem ser utilizados para a pesquisa de fluidos de perfuração

Depois de escolher o filtro, o usuário terá como resposta os fluidos pesquisados com o filtro selecionado. Segue um exemplo de filtro na imagem abaixo.

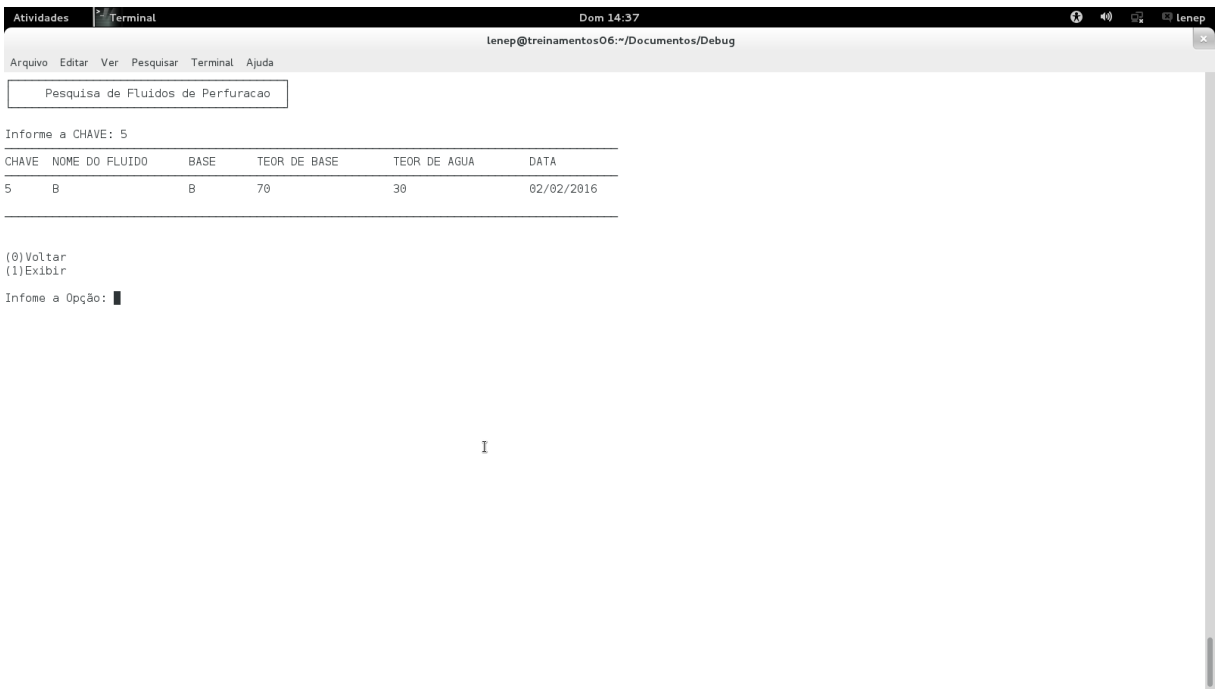

Figura 7.6: Tela do programa mostrando a pesquisa de fluidos de perfuração

7.4 Teste 4: Inserir fluidos de perfuração

Após acessar o Banco de Dados, o usuário terá acesso ao menu com diferentes opções. Após escolher inserir fluidos, o usuário deverá ir caracterizando o fluido a ser inserido com alguns de seus dados, como os apresentados na imagem abaixo.

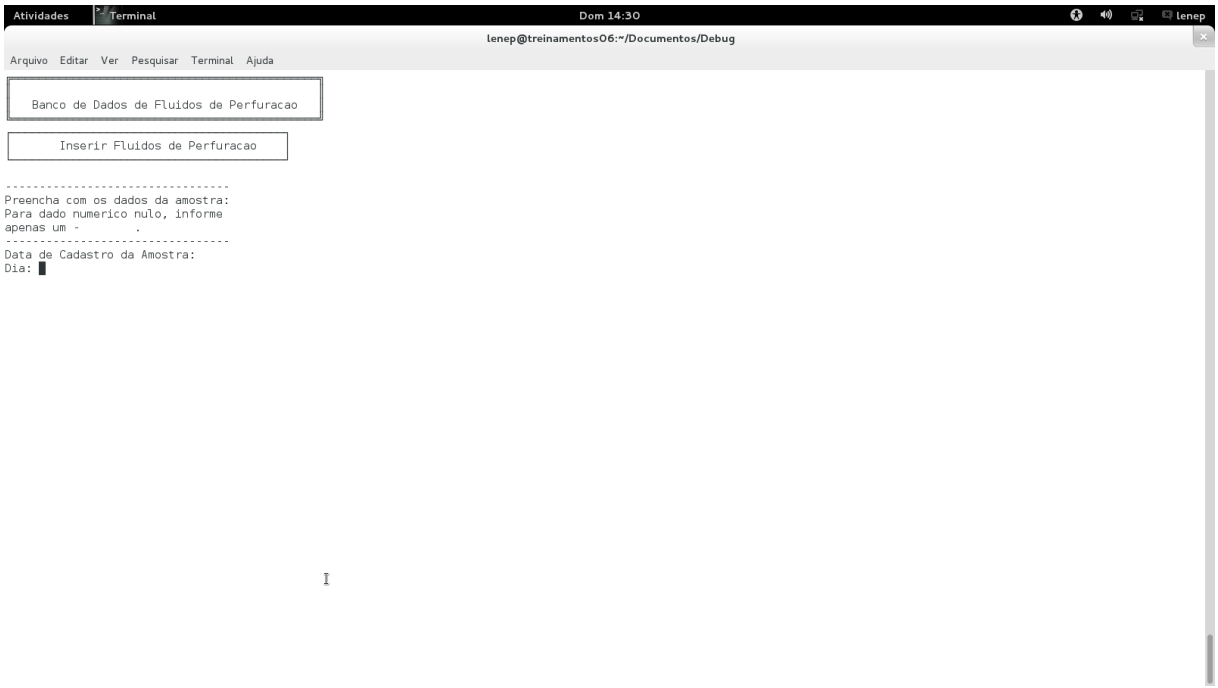

Figura 7.7: Tela do programa mostrando alguns dados a serem respondidos pelo usuário na inserção de fluidos de perfuração

7.5 Teste 5: Exportar fluidos de perfuração

Após acessar o Banco de Dados, o usuário terá acesso ao menu com diferentes opções. Após escolher exportar fluidos, o usuário terá a opção de exportar a amostra de forma completa ou de voltar. Escolhendo a opção de exportar de forma completa, o usuário deverá escolher a chave a ser exportada e depois ele terá como resposta um arquivo no formato txt na pasta relatorios. A imagem abaixo mostra uma captura de tela de um relatório gerado.

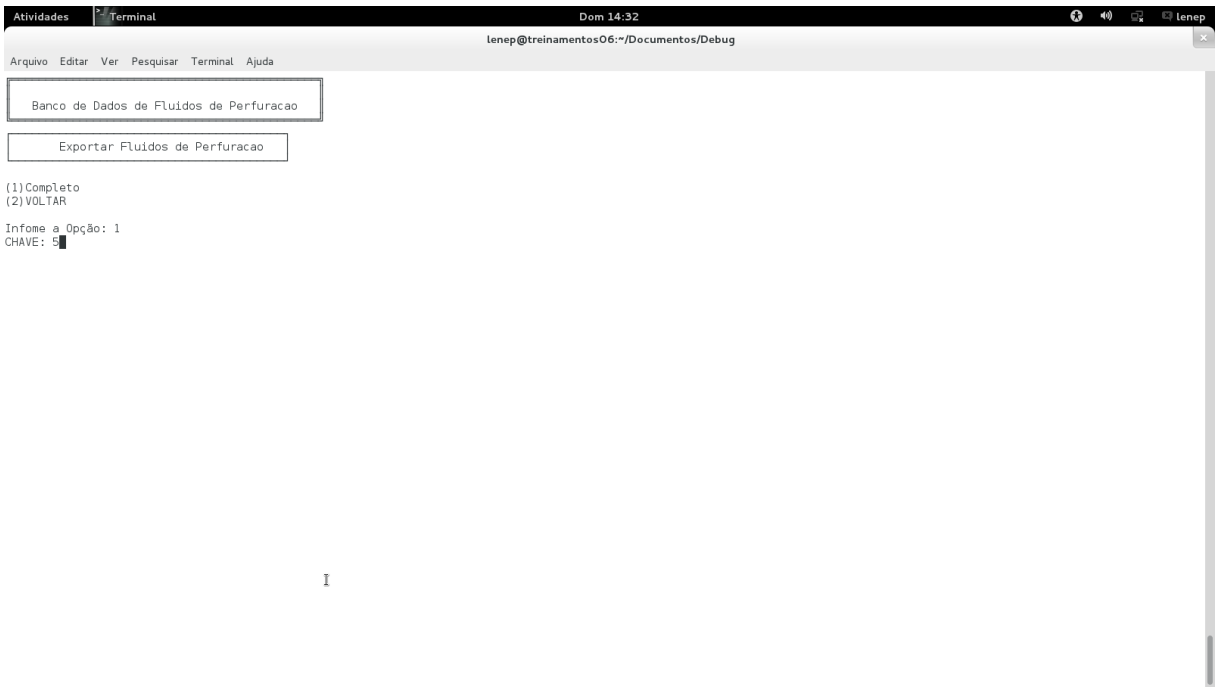

Figura 7.8: Tela do programa mostrando o menu para exportar fluidos de perfuração

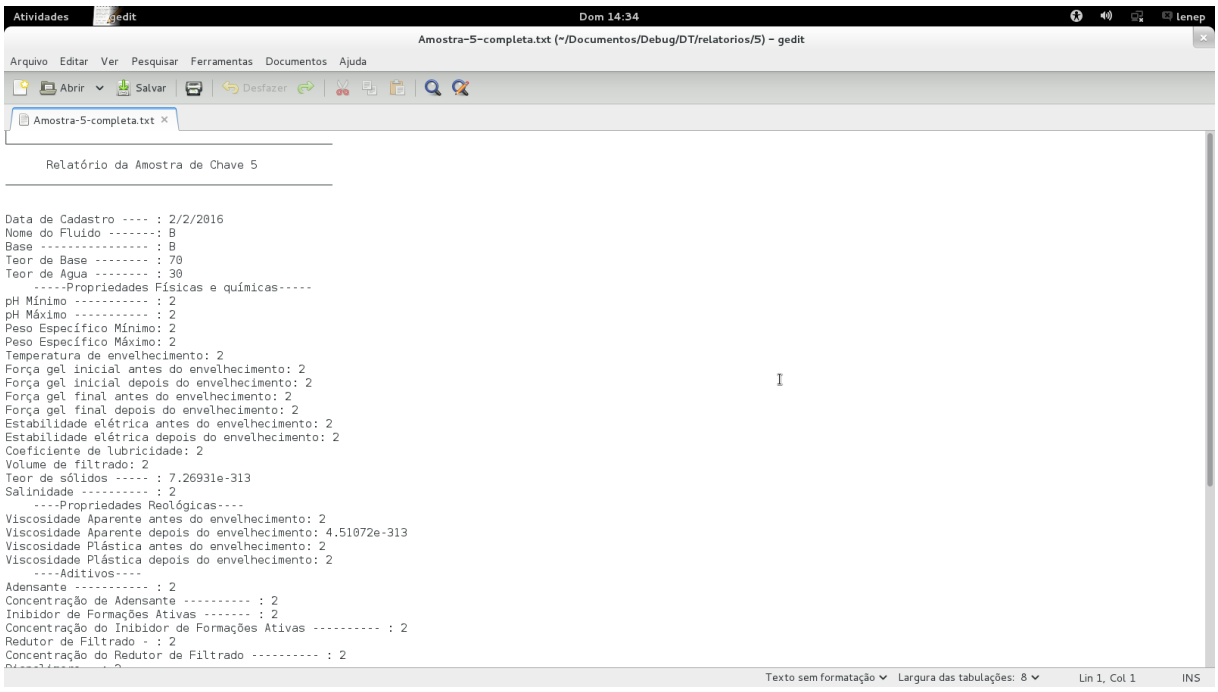

Figura 7.9: Tela do programa mostrando o relatório gerado de um fluido de perfuração

7.6 Teste 6: Excluir fluidos de perfuração

Após acessar o Banco de Dados, o usuário terá acesso ao menu com diferentes opções. Após escolher excluir fluidos, o usuário deverá escolher a chave a ser excluída. A imagem abaixo mostra uma captura de tela do menu de exclusão do fluido.

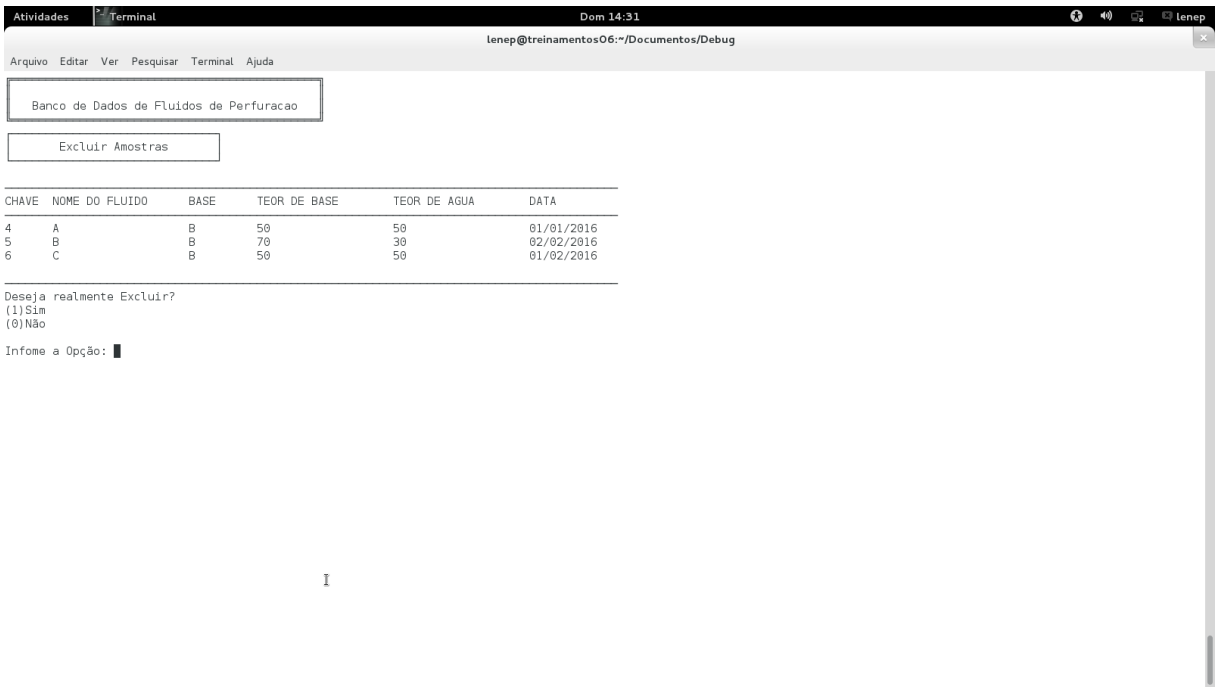

Figura 7.10: Tela do programa mostrando o menu de exclusão de fluidos de perfuração

Capítulo 8

Documentação

A presente documentação refere-se ao uso do Banco de Dados de Fluidos de Perfuração.

8.1 Documentação do usuário

Abaixo encontra-se uma explicação de como o usuário pode utilizar o programa.

8.1.1 Como utilizar o software

Após abrir o terminal, compilar o programa e, depois, executá-lo, o usuário terá um menu inicial com as seguintes opções:

1. Banco de Dados
2. Configurações
3. Abrir Diretório de Trabalho
4. Abrir Diretório de Dados

Se o usuário escolher a opção Banco de Dados, um menu referente ao Banco de Dados será apresentado, com as seguintes opções:

1. Listar
2. Pesquisar
3. Inserir
4. Excluir
5. Exportar
6. Voltar

Uma vez escolhida a opção de Listar, o usuário terá os fluidos de perfuração presentes no Banco de Dados listados com algumas características principais, além da opção de poder exibir fluidos selecionados.

Se a opção escolhida for a de pesquisar, o usuário poderá pesquisar fluidos com os seguintes filtros:

1. Chave
2. Nome do fluido
3. Base
4. pH mínimo
5. pH máximo
6. Teor de água
7. Peso específico mínimo
8. Peso específico máximo
9. Temperatura de envelhecimento
10. Salinidade
11. Filtrado
12. Dia
13. Mês
14. Ano
15. Listar todos
16. VOLTAR

Se o usuário selecionar a opção Inserir, ele deverá fornecer alguns dados para a caracterização dos diferentes fluidos.

Escolhendo a opção excluir, o usuário deverá selecionar a chave do fluido de perfuração a ser excluída.

Por fim, ao selecionar a opção exportar, o usuário terá um relatório do fluido selecionado na forma de um arquivo no formato txt.

Ao escolher a opção Configurações no menu inicial, o usuário poderá configurar o programa explorador e o editor de textos.

Na Opção Abir Diretório de Trabalho e Abrir Diretório de Dados, o usuário poderá abrir os respectivos diretórios.

8.2 Documentação para desenvolvedor

A documentação abaixo é apresentada 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/.

8.3 Referências Bibliográficas

- Estudo preliminar sobre o uso de glicerina proveniente da produção de biodiesel como base para fluidos de perfuração dos poços de petróleo.

Índice Remissivo