UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE CIÊNCIAS E TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO
EM ENGENHARIA QUÍMICA (MESTRADO)
LABORATÓRIO DE REFERÊNCIA EM DESSALINIZAÇÃO

ELÓI DUARTE DE MÉLO

NORMALIZAÇÃO DE UM SISTEMA DE DESSALINIZAÇÃO PILOTO UTILIZANDO UM MÓDULO DE GERÊNCIA DISTRIBUÍDA COM CAPACIDADE DE MONITORAÇÃO REMOTA

CAMPINA GRANDE - PB
2004

ELÓI DUARTE DE MÉLO

NORMALIZAÇÃO DE UM SISTEMA DE DESSALINIZAÇÃO PILOTO UTILIZANDO UM MÓDULO DE GERÊNCIA DISTRIBUÍDA COM CAPACIDADE DE MONITORAÇÃO REMOTA

Dissertação em Engenharia Química apresentada a Universidade Federal de Campina Grande – UFCG, em cumprimento dos requisitos necessários para a obtenção do grau de Mestre em *Engenharia Química*, área de concentração *Operações e Processos,* linha de pesquisa *Transferência de Massa,* elaborada após integralização curricular do Programa de Pós-Graduação em Engenharia Química.

Orientador: Prof. Kepler B. França (Ph.D)

CAMPINA GRANDE - PB
2004

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL DA UFCG

M528n
2004

Melo, Eloi Duarte de
Normalização de um sistema de dessalinização piloto utilizando um módulo de gerência distribuída com capacidade de monitoração remota / Eloi Duarte de Melo . — Campina Grande: UFCG, 2004.
156p.: il.

Inclui Bibliografia

Dissertação (Mestrado em Engenharia Química) — Centro de Ciências e Tecnologia, Universidade Federal de Campina Grande.

1— Osmose Inversa – Normalização 2— Monitoração Remota I— Título

CDU 66.081.63

ELÓI DUARTE DE MÉLO

NORMALIZAÇÃO DE UM SISTEMA DE DESSALINIZAÇÃO PILOTO UTILIZANDO UM MÓDULO DE GERÊNCIA DISTRIBUÍDA COM CAPACIDADE DE MONITORAÇÃO REMOTA

Dissertação em Engenharia Química apresentada a Universidade Federal de Campina Grande – UFCG, em cumprimento dos requisitos necessários para a obtenção do grau de Mestre em *Engenharia Química.*

Aprovada em: 26 **de** 11 **de** 2004.

COMISSÃO EXAMINADORA

Prof. Dr. Kepler Borges França – UFCG/DEQ

Prof. Dr. Walfredo da Costa Cirne Filho – UFCG/DSC

Prof. Dr. Gilmar Trindade de Araújo – UFCG/DEQ

Prof. Dr. Michel François Fossy – UFCG/DEQ

À Edinho (in memorian).

AGRADECIMENTOS

A Deus por ter permitido a realização desse trabalho e por sempre ter me dado força e humildade para a concretização dessa pesquisa.

A minha querida, especial e linda esposa, Francisca Maria de Melo, por sua compreensão e companheirismo nos momentos de dificuldades e de alegrias e, principalmente, pelo seu grande amor por mim manifestado quando mais preciso.

Aos meus pais, Antônio e Margarida, e irmãos, Edilson e Eliene, pelo amor e compreensão nos momentos em que me ausentei.

Ao professor Kepler B. França, por ter compartilhado comigo seus conhecimentos científicos.

A todos os professores e ao coordenador do Programa de Pós-Graduação em Engenharia Química por contribuírem para a minha formação.

A todos os funcionários do LABDES e do Programa de Pós-Graduação em Engenharia Química pela a atenção e a ajuda prestada a mim nos momentos oportunos.

Aos amigos de luta do LABDES Fox, Aécio, Raniere, Renê, Francinaldo, Damião e Maurício pelo compartilhamento de informações, conhecimentos e papo furado que em muito contribuiram para a realização deste trabalho.

Ao **Fundo de Recursos Hídricos (CT-HIDRO)** do **CNPq** pelo apoio financeiro.

MÉLO, Elói Duarte de. Normalização de um sistema de dessalinização piloto utilizando um módulo de gerência distribuída com capacidade de monitoração remota. 106p, 2004. Mestrado em Engenharia Química (Programa de Pós-Graduação em Engenharia Química – área de concentração Operações e Processos, linha de pesquisa Transferência de Massa) – Universidade Federal de Campina Grande – Campina Grande – Paraíba.

Resumo

A maioria dos sistemas de dessalinização via osmose inversa instalados no nordeste brasileiro vem enfrentando problemas como a falta de capacitação de operação, monitoração e manutenção, o que diminui a vida útil das membranas dos equipamentos devido a formação de incrustações durante o processo. Nesse sentido, o presente trabalho visa introduzir no software desenvolvido (SISMODES), um conjunto de equações do processo de osmose inversa para realizar a normalização de um sistema de dessalinização piloto (SDP) e obter, a partir dos parâmetros normalizados, um diagnóstico do tipo de incrustação que está ocorrendo. Para isso, o SDP possui um conjunto de sensores que enviam os valores das variáveis de medidas, através do computador embarcado, para o banco de dados da máquina servidor através da linha telefônica. O SGBD (Sistema de Gestão de Banco de Dados) utilizado foi o SGBD Interbase 6.0 da Borland que é free software e o sistema operacional usado foi o Linux Mandrake 8.1. Todo o sistema web foi implementado em linguagem Java e o servidor utilizado para exibir as páginas HTML dinâmicas foi o servidor tomcat do projeto Jakarta. Toda a interface do projeto foi desenvolvida em JSP. Além de possibilitar o gerenciamento remoto do sistema de dessalinização piloto (SDP), o software desenvolvido faz uma normalização do SDP em tempo real, permitindo a exibição de gráficos que mostram o comportamento das variáveis de medidas, bem como dos parâmetros normalizados, em função do tempo de operação do SDP. O software também é capaz de exibir um diagnóstico de incrustação em função dos parâmetros normalizados. Além de indicar o tipo e a localização da incrustação no sistema, SISMODES indica uma série de ferramentas experimentais que podem ser usadas para se confirmar o diagnóstico, bem como aponta as ações corretivas que devem ser tomadas pelos operadores / técnicos para acabar ou inibir a incrutação.

PALAVRAS-CHAVE: Normalização, incrustação, monitoração remota.

MÉLO, Elói Duarte. *Desalination system pilot normalization using a distributed management module with monitoring remote capacity*. 106p, 2004. Mastership of a military order in Chemical Engineering (concentration area: *Operations and Processes* and line of research: Mass Transference) - Federal University of Campina Grande - Campina Grande – Paraíba – Brazil.

Abstract

The majority of the desalination systems via reverse osmosis installed in the brazilian northeast comes facing problems as the lack of operation qualification, monitoring and maintenance. As consequence of this, the membranes useful life of the equipment comes decrease due the fouling formation during the desalination process. In this direction, the present work aims at to introduce in developed software (SISMODES), a set of equations of the reverse osmosis process to carry through the pilot desalination system (SDP) normalization from the measures variable that are sent to the management server and to get, from the normalized parameters, a diagnosis of the incrustation type that is occurring. For this, the SDP possess a set sensors that send the values of the measures variable, through the embarked computer, for the serving machine data base through the telephonic line. The SGBD (Data base System Management) used was the SGBD Interbase 6.0 of the Borland that is free softwares and the used operational system was the Linux Mandrake 8.1. All the system web was implemented in Java language and the used server to show dynamic pages HTML was the server tomcat of the Jakarta project. All the project interface was developed in JSP. Besides making possible the remote management of the pilot desalination system (SDP), developed software makes a SDP normalization in real time, allowing the graphs exhibition that show the measures variable behavior, as well as, of the normalized parameters, in function of the SDP running time. The software also is capable to show a incrustation diagnosis in function of the normalized parameters. Besides indicating the type and the localization of the system incrustation, SISMODES indicates a series of experimental tools that can be used to confirm the diagnosis, as well as points the corrective actions that must be taken by the operators/technician to finish or to inhibit the incrustation.

KEY-WORDS: Normalization, incrustation, remote monitoring.

SIMBOLOGIA

A = área total das membranas (m^2)

C_a = concentração inicial de sais dissolvidos na corrente de alimentação (mg/L)

C_{ac} = média da concentração alimentação-concentrado (mg/L)

C_c = concentração de sais dissolvidos na corrente do concentrado (mg/L)

c_i = concentração molar do soluto (mol/L)

C_p = concentração de soluto na corrente de permeado (mg/L)

C_{Pn} = Sólidos totais dissolvidos do permeado normalizado (mg/L)

C_{STD} = concentração de sólidos totais dissolvidos

CTM = Coeficiente de transferência de massa ($m/h.kgf.cm^{-2}$)

DDL = Linguagem de Definição de Dados

FC = fator de concentração (adimensional)

FCT = fator de correção de temperatura (adimensional)

FC_{lm} = fator de concentração média logarítmica (adimensional)

ISL = Índice de Saturação de Langelier (adimensional)

Ip_c = produto iônico do concentrado

J2SE = Java 2 Standard Edition

JDBC = Java Database Connectivity

J_p = fluxo do permeado (L/m^2h)

JSP = Java Server Pages

k_{os} = constante osmótica

Ks_c = produto de solubilidade do concentrado

K_w = coeficiente de permeação de água ($L/m^2h.kgf.cm^{-2}$)

NDP = gradiente médio de pressão efetivo da membrana (kgf/cm^2)

OI = Osmose inversa

p1 = pressão de entrada dos filtros (kgf/cm^2)

p2 = pressão de saída dos filtros (kgf/cm^2)

p3 = pressão de entrada das membranas (kgf/cm^2)

p4 = pressão de saída das membranas (kgf/cm^2)

P_a = pressão de alimentação (kgf/cm^2)

pH: potencial hidrogeniônico da alimentação

pH_s: potencial hidrogeniônico no qual a alimentação fica saturado com $CaCO_3$

P_p = pressão do permeado (kgf/cm^2)

PS = passagem de sais (%)

PS_n = Passagem de sal normalizada (%)

Q_a = vazão de alimentação (m^3/h)

Q_c = vazão do concentrado (m^3/h)

Q_n = vazão do permeado normalizada (m^3/h)

Q_p = vazão do permeado (m^3/h)

r = razão de recuperação (%)

R = constante dos gases ($kgf.L/cm^2mol.K$)

RS = rejeição de sais (%)

SDP = Sistema de Dessalinização Piloto

STD = sólidos totais dissolvidos (mg/L)

SQL = Structured Query Language

T = temperatura da alimentação (°C)

tdsA = total de sólidos da solução de alimentação (mg/L)

T_{ref} = temperatura de referencia (°C)

v1 = vazão do concentrado (LPM)

v2 = vazão do permeado (LPM)

ΔP = diferencial de pressão (kgf/cm^2)

$\Delta\pi$ = gradiente de pressão osmótica (kgf/cm^2)

π = pressão osmótica da solução iônica (kgf/cm^2)

π_a = pressão osmótica da alimentação (kgf/cm^2)

π_c = pressão osmótica do concentrado (kgf/cm^2)

π_P = pressão osmótica do permeado (kgf/cm^2)

ν_i = nº de íons formados na dissociação do soluto

γ = coeficiente de atividade dos componentes do sal na solução

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

SUMÁRIO

INTRODUÇÃO

O Laboratório de Referência em Dessalinização (LABDES) da Universidade Federal de Campina Grande, juntamente com a Secretaria de Recursos Hídricos do Ministério do Meio Ambiente através do Programa Água Boa, investiram no processo de dessalinização, via osmose inversa, para atender pequenas comunidades com água de boa qualidade a partir de águas salobras e salinas de poços tubulares existentes nas regiões semi-áridas do Nordeste. Apesar dos benefícios sociais que o processo de dessalinização de águas fornece às comunidades contempladas, diversos tipos de problemas vêm sendo enfrentados com os equipamentos instalados no campo: a falta de capacitação de operação, monitoração e manutenção.

Como conseqüência disso, as membranas dos equipamentos vêm apresentando um tempo de vida útil menor do que 3 anos. Segundo levantamento do CTHidro, existem vários dessalinizadores instalados no Nordeste e poucos funcionam devido a problemas diversos normalmente relacionados à falta de manutenção. É essencial que estes equipamentos tenham uma boa manutenção para que os mesmos possam operar com o mesmo conjunto de membranas por no mínimo 5 anos, produzindo água de boa qualidade.

O fato de existir uma grande quantidade de dessalinizadores instalados sob uma mesma administração, torna-se inviável realizar a gerência desses equipamentos tomando como base apenas os dados coletados, através de vistorias presenciais, realizadas por operadores no campo (Almeida, 2002). Para uma boa monitoração surge então a necessidade de automação das vistorias e centralização dos dados obtidos através delas.

Uma forma de solucionar este problema é através da realização de atividades de gerência remota dos dessalinizadores, onde estes, dentro de intervalos de tempos pré-estabelecidos, coletam e armazenam informações e, periodicamente, submetem as informações obtidas nessas coletas para estações servidoras de gerência. Os dados coletados dos vários dessalinizadores distribuídos no campo podem ser então tratados convenientemente. O software executando na estação gerente pode, por exemplo, verificar se as informações atingem um valor crítico, que possa gerar mau-funcionamento, ou mesmo danificar o dessalinizador e avisar ao responsável pela administração para que as medidas cabíveis possam ser tomadas. Além disso, a partir das informações das variáveis de medidas em tempo de operação, pode-se fazer a normalização do sistema de dessalinização através de um conjunto de equações do processo de osmose inversa introduzidas no software. A

normalização do sistema vem sendo uma ferramenta eficiente para avaliar o potencial de incrustação e a projeção do custo de manutenção do processo. Em termos gerais, a normalização consiste em comparar os valores atuais dessas variáveis com os valores de referências. Com isso, pode-se chegar a uma série de diagnósticos, em função dos parâmetros normalizados, que indicam a presença de incrustações, bem como o seu tipo e sua localização no sistema de membranas.

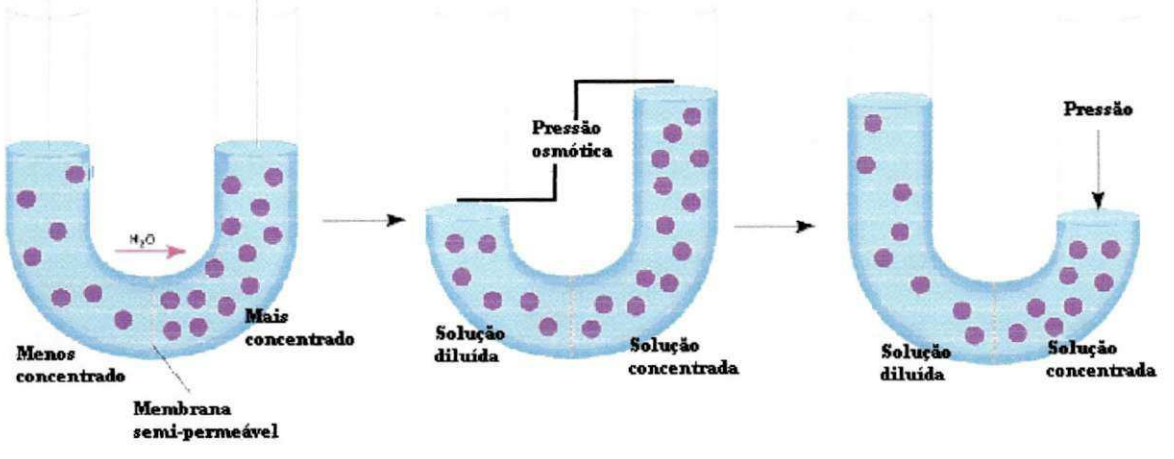
Desse modo, o presente trabalho visa introduzir no software desenvolvido, denominado de SISMODES, um conjunto de equações do processo de osmose inversa para realizar a normalização do sistema de dessalinização piloto (SDP) a partir das variáveis de medidas que são enviadas ao servidor de gerência e obter, a partir dos parâmetros normalizados, um diagnóstico do tipo de incrustação que está ocorrendo. Com isso, as ações corretivas para acabar e/ou diminuir a incrustação são indicadas, aumentando desse modo a vida útil das membranas e otimizando o processo de dessalinização.

CAPÍTULO I

Revisão Bibliográfica

1.1. Processo de osmose inversa

A osmose ocorre quando duas soluções salinas de concentrações diferentes encontram-se separadas por uma membrana semipermeável. Neste caso, conforme pode ser observado na Figura 1.1, a água (solvente) da solução menos concentrada tenderá a passar para o lado da solução de maior salinidade. Com isto, esta solução mais concentrada, ao receber mais solvente, se dilui, num processo impulsionado por uma grandeza chamada *pressão osmótica*, até que as duas soluções atinjam concentrações iguais (Joyce *et al.*, 2001).



(a) (b) (c)

Figura 1.1: Representação do processo de osmose e osmose inversa: (a) duas soluções, uma salina e outra sem sal , separadas por uma membrana semipermeável; (b) a água pura dilui a salgada até que seja atingido o equilíbrio osmótico; (c) a aplicação de uma pressão superior à diferença de pressão hidrostática inverte o processo. (Fonte: Kerr & Mchale, 2001).

A osmose inversa é utilizada para dessalinizar águas salinas, salobras e de superfície, utilizando membranas semipermeáveis sintéticas. A pressão aplicada deve superar a pressão osmótica da solução para separar os sais da água (Figura 1.1). Na prática, a pressão de operação deve superar também a resistência da membrana, a resistência da zona de polarização de concentração e a resistência interna do equipamento. As pressões de operação reais são, portanto, mais elevadas do que a pressão osmótica da solução. A principal função

das membranas é a rejeição de sais, que depende da temperatura, pressão, pH, concentração de sal e rendimento (Schneider & Tsutiya, 2001).

A osmose inversa (OI) é, portanto, uma operação unitária que através de membranas semipermeáveis e com auxílio de um gradiente de pressão, pode rejeitar sais inorgânicos de baixo peso molecular, como também pequenas moléculas orgânicas numa faixa menor que 200 Daltons (Ozaki e Li, 2002). As moléculas de água, por outro lado, passam livremente através da superfície da membrana, criando uma corrente de água purificada. A parcela restante da água de alimentação que não atravessa a membrana, conhecida como concentrado ou rejeito, leva consigo os compostos rejeitados pela mesma. As rejeições típicas de sais dissolvidos atingem a marca de 95 a 99%.

Os sistemas de separação por membranas ocorrem através de um método denominado de fluxo cruzado e compreende a utilização de uma corrente de alimentação pressurizada fluindo paralelamente a superfície da membrana. Como podemos observar na Figura 1.2, esta corrente de alimentação é dividida em duas correntes de saída: a solução que passou através da superfície da membrana (permeado) e a remanescente denominada de concentrado (Habert *et al.*, 1997).

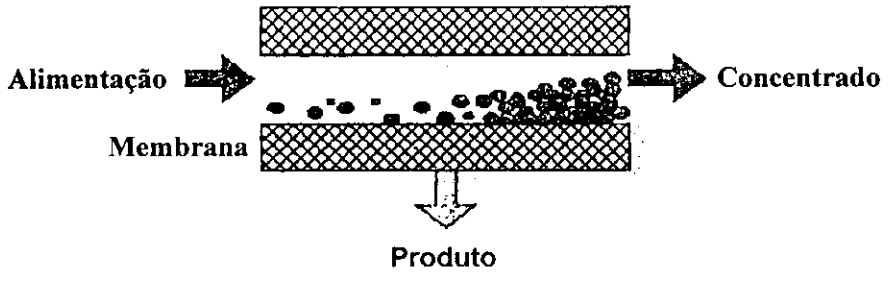


Figura 1.2: Modelo de filtração de fluxo cruzado (Fonte: Sousa, 2001).

1.2. Principais parâmetros do processo

1.2.1. Rejeição de sais (RS)

A rejeição de solutos pelas membranas é influenciada por uma grande variedade de fatores, tais como as dimensões do soluto, a morfologia dos componentes retidos pela membrana, o tamanho dos poros da membrana e as propriedades químicas da solução a ser filtrada. A eficácia de processos de separação é geralmente medida pelo parâmetro denominado rejeição de sais (RS). Este parâmetro é calculado de acordo com a Equação 1, (Wiesner & Aptel, 1996):

$$RS(\%) = \left(1 - \frac{C_p}{C_a}\right) * 100 \quad (1)$$

Onde: C_a = concentração inicial de sais dissolvidos na corrente de alimentação (mg/L); C_p = concentração de soluto na corrente de permeado (mg/L).

A rejeição de sais indica a efetividade de remoção de sais e outras espécies químicas pela membrana, possuindo valores que variam de 90 a 99,8 % para a maioria dos íons existentes na água.

A determinação exata deste parâmetro em sistemas de membranas é praticamente impossível, pois seria necessário medir com precisão a concentração do soluto rejeitado na faixa da subcamada laminar em contato com a superfície da membrana. Além disso, o valor exato da rejeição varia ao longo do módulo em função do processo de permeabilização e do conseqüente aumento da concentração dos materiais retidos (Schneider & Tsutiya, 2001).

1.2.2. Passagem de sais (PS)

A passagem de sais é um parâmetro que se comporta de maneira oposta à rejeição de sais, sendo definido como a porcentagem do sal na água de alimentação que atravessa a membrana. Segundo Brandt & Leitner (1992), este parâmetro é calculado de acordo com a Equação 2:

$$PS(\%) = \left(\frac{C_p}{C_a}\right) * 100 \quad (2)$$

Onde: PS = passagem de sais (%); C_a = concentração inicial de sais dissolvidos na corrente de alimentação (mg/l); C_p = concentração de soluto na corrente de permeado (mg/L).

1.2.3. Razão de recuperação

A razão de recuperação do sistema é definida como sendo a porcentagem da água de alimentação que é convertida em água purificada e depende de vários fatores, como a formação de incrustação na superfície das membranas, a pressão osmótica e a qualidade da água de alimentação do sistema. Segundo Taylor & Jacobs (1996), a recuperação de um sistema pode ser definido de acordo com a Equação 3:

$$r(\%) = \frac{Q_p}{Q_a} * 100 = \frac{Q_p}{Q_p + Q_c} * 100 \quad (3)$$

Onde: r = razão de recuperação (%); Q_p = vazão do permeado (m^3/h); Q_a = vazão de alimentação (m^3/h) e Q_c = vazão do concentrado (m^3/h).

1.2.4. Concentração de sais dissolvidos

A concentração de sais dissolvidos na corrente do concentrado é matematicamente estimado, baseando-se nos resultados analíticos obtidos das águas de alimentação, através da Equação 4 (El-Manharawy & Hafez, 2001):

$$C_c = C_a * FC \tag{4}$$

Onde: C_c = concentração de sais dissolvidos na corrente do concentrado (mg/L); C_a = concentração de sais dissolvidos na corrente de alimentação (mg/L) e FC = fator de concentração.

O fator de concentração é obtido através da Equação 5 (Bradley, 1992; Taylor & Jacobs, 1996; El-Manharawy & Hafez, 2001).

$$FC = \frac{1}{1-r} \tag{5}$$

Onde: r = razão de recuperação do sistema.

A Equação 5 é válida desde que seja assumido 100% de rejeição. Sabe-se que alguns íons atravessam a membrana e saem no fluxo do permeado (Taylor & Jacobs, 1996; El-Manharawy & Hafez, 2001).

1.2.5. Fluxo do permeado

Segundo Bradley (1992), o fluxo do permeado através de uma membrana de osmose inversa é inversamente proporcional à área da membrana e proporcional à variação de pressão osmótica e hidráulica, sendo dado pela Equação 6:

$$J_p = K_w * (\Delta P - \Delta\pi) = \frac{Q_P}{A} \tag{6}$$

Onde: J_p = fluxo do permeado (L/m^2h); K_w = coeficiente de permeação de água ($L/m^2h.kgf.cm^{-2}$); A = área da membrana (m^2); ΔP = diferencial de pressão hidráulica (kgf/cm^2) e $\Delta\pi$ = gradiente de pressão osmótica (kgf/cm^2).

1.2.6. Balanço de massa

A Equação 7 apresenta o balanço de massa (em regime permanente) para um sistema de dessalinização.

$$Q_aC_a = Q_cC_c + Q_pC_p \quad (7)$$

Onde: Q_a = vazão de alimentação (m^3/h); C_a = concentração inicial de sais dissolvidos na corrente de alimentação (mg/L); Q_p = vazão do permeado (m^3/h); C_p = concentração de sais dissolvidos na corrente de permeado (mg/L); Q_c = vazão do concentrado (m^3/h) e C_c= concentração de sais dissolvidos na corrente do concentrado (mg/L).

Da definição da razão de recuperação (Equação 3), temos que:

$$Q_P = rQ_a \quad (8)$$

$$Q_c = \frac{Q_P(1-r)}{r} \quad (9)$$

Substituindo as Equações 8 e 9 na Equação 7, temos:

$$Q_aC_a = C_PrQ_a + C_c\frac{Q_P(1-r)}{r} \Rightarrow C_a = C_Pr + C_c(1-r) \Rightarrow C_c = \frac{C_a - rC_P}{(1-r)} \quad (10)$$

Se $rC_P <<< C_a$, obtem-se a Equação 4:

$$C_c = \frac{C_a}{(1-r)} \Rightarrow C_c = C_a * FC$$

Se $C_P = 0$, das Equações 7 e 10 obtem-se a Equação 5:

$$Q_aC_a = C_cQ_c \Rightarrow \frac{Q_a}{Q_c} = \frac{C_c}{C_a} = \frac{1}{(1-r)} = FC$$

1.2.7. Pressão osmótica

A pressão osmótica é função da concentração de sais e do tipo das moléculas orgânicas contidas na água de alimentação. Quanto maior for a concentração da solução, maior será o valor da pressão osmótica dessa solução (Brandt *et al.*, 1992). Ela pode ser calculada pela equação de Van't Hoff:

$$\pi = \sum \nu_i c_i RT \quad (11)$$

Onde: π = pressão osmótica da solução iônica (kgf/cm^2); ν_i = nº de íons formados na dissociação do soluto; c_i = concentração molar do soluto (mol/L); R = constante dos gases ($kgf.L/cm^2mol.K$) e T = temperatura (K).

Na prática, a pressão osmótica pode ser aproximada a partir da concentração total de sais (totais de sólidos dissolvidos):

$$\pi = k_{os} R C_{STD} \quad (12)$$

Onde k_{os} = constante osmótica; R = constante de gases e C_{STD} é a concentração de sólidos totais dissolvidos. O fator k_{os} varia entre 0,0063 e 0,0115.

1.3. Incrustações em sistemas de osmose inversa

As incrustações podem ser entendidas como todo o material em suspensão na água ou dissolvida na mesma que se deposita na superfície da membrana. Os problemas operacionais em instalações com membranas de osmose inversa usadas na dessalinização de água podem ser causados por uma variedade de tipos de incrustações. A incrustação das membranas causa um aumento nos custos operacionais devido a maior demanda de energia necessária, ao aumento das limpezas e a redução da vida útil dos elementos de membranas.

Segundo Amjad (1992), os mecanismos de incrustações das membranas de osmose inversa incluem biofouling, incrustação orgânica, incrustação inorgânica (incluindo scaling) e a incrustação particulada. A incrustação deteriora o desempenho das membranas. Entretanto, o lugar onde ocorre a deterioração depende do tipo de incrustação (Tabela 1).

Tabela 1: Posição da incrustação nas plantas de osmose inversa (Huiting *et al.*, 2001).

Incrustação	Onde ele ocorre primeiro
Scaling/sílica	Última membrana do último arranjo
Óxidos metálicos	Primeira membrana do primeiro arranjo
Colóides	Primeira membrana do primeiro arranjo
Orgânico	Primeira membrana do primeiro arranjo
Biofouling (rápido)	Primeira membrana do primeiro arranjo
(lento)	Em todo conjunto de instalação

Como a incrustação é progressiva, ela pode, se não for controlada inicialmente, prejudicar o desempenho das membranas em um tempo relativamente curto.

Os depósitos inorgânicos são sais normalmente solúveis na água e que precipitam e se cristalizam na superfície das membranas. Isso é facilmente entendido quando se verifica que na área externa próxima à membrana a concentração de sais é bem superior à concentração do

sal na água de entrada, podendo atingir o limite de solubilidade e assim precipitar (Benret, 1996).

A precipitação de sais em superfície de membranas é um problema que ocorre com freqüência na OI, onde fatores como altos índices de rejeição de sais (até 99%) e a remoção de até 90% de sais da água aumentam a concentração de sais no concentrado e na superfície da membrana. Um outro fator muito importante que contribui para a precipitação de sais é a polarização de concentração, fenômeno no qual a concentração de sais dissolvidos próxima à superfície da membrana é maior do que a concentração média da água que flui em volta da superfície considerada. O controle dos processos de precipitação de sais é um dos principais condicionantes de projeto em sistemas de OI, onde deve ser precedido de uma análise dos componentes da água de alimentação que podem precipitar na superfície da membrana. Os componentes que oferecem maior risco para a formação de precipitados na superfície de membranas são a sílica, o carbonato de cálcio e os sulfatos de cálcio, estrôncio e bário. Um sal precipita quando a concentração dos seus componentes ultrapassa o valor limite correspondente ao produto de solubilidade (Kps) e geralmente a precipitação por scaling ocorre nos elementos de membranas instalados na saída dos vasos de pressão localizados na última bancada do sistema (Schneider & Tsutiya, 2001).

O depósito de material biológico ("biofouling") é definido como o acúmulo, crescimento e/ou depósito de biomassa na superfície da membrana, ocasionando problemas operacionais (Vrouwenvelder & Kooij, 2001). Na osmose inversa, a ocorrência de biofilme é indicada pela contínua redução do fluxo através das membranas ou pelo aumento da pressão de operação, necessária para manter uma determinada taxa de fluxo. Em casos extremos, o biofilme pode causar o colapso telescópico de elementos de membranas em espirais pelo deslocamento lateral de canais adjacentes (Schneider & Tsutiya, 2001).

O Biofouling é a incrustação mais comum das membranas nas plantas de dessalinização via osmose inversa seguido por scaling/óxidos metálicos, colóides e incrustação orgânica (Huiting *et al.*, 2001). No entanto, este tipo de incrustação ainda não recebe um pré-tratamento adequado em sistemas instalados no Brasil. Além disso, segundo Vrouwenvelder *et al.* (2003), a qualidade da água de alimentação pode ser influenciada pelas variações (climáticas) da qualidade da água anterior ao pré-tratamento, em que o material biológico passa a não ser completamente eliminado. Um outro ponto é que a dosagem de produtos químicos (floculantes e anti-incrustantes) à água de alimentação pode causar biofouling sério, conduzindo à substituição antecipada das membranas (Van der Hoek *et al.*, 1997). Vrouwenvelder *et al.* (2000), pesquisando anti-incrustantes baseados em polímeros,

demonstrou que os produtos químicos usados para impedir a incrustação diferem extremamente em sua habilidade de promover o crescimento dos microrganismos.

1.4. Normalização do sistema de osmose inversa

Durante a operação de um sistema de osmose inversa, as condições de operação tais como: a pressão, a temperatura, a recuperação do sistema e a concentração de alimentação podem variar, causando variações na produtividade e na qualidade de água do permeado. Por exemplo, uma diminuição da temperatura de alimentação de 4°C causará uma diminuição no fluxo de permeado de aproximadamente 10%. Para avaliar eficazmente o desempenho do sistema, é necessário comparar o desempenho nas mesmas condições. Conseqüentemente, é necessário converter os dados operacionais obtidos em condições reais em uma série de condições padronizadas selecionadas, normalizando desse modo o desempenho dos dados (Huiting *et al.*, 2001).

A fim de acompanhar o desempenho de um sistema de membrana, recomenda-se que todos os dados relevantes sejam coletados e gravados. Uma vez que os dados são recolhidos, necessitam ser analisado para determinar o desempenho do sistema de membrana em função do tempo de operação. A normalização é desse modo um processo que compara o desempenho da dessalinização calculado de hoje, com o desempenho da partida do sistema tida como uma base de referência determinada previamente, considerando tempo e parâmetros operacionais. A normalização é realizada para ler tendências periódicas para qualidade e quantidade de permeado, e não valores diários isolados do sistema.

As mudanças nos parâmetros normalizados indicam um possível problema na operação da planta. Com uma normalização apropriada pode ser detectada onde e quando o problema pode ocorrer (Huiting *et al.*, 2001). O monitoramento do desempenho do sistema de OI demonstra ser uma ferramenta para descobrir possíveis problemas de formação de incrustações nas membranas. A normalização dos dados operacionais dá condições de prever a formação de incrustações e através dos cálculos identificar a sua natureza, sem a necessidade de interromper a operação do sistema. Com a normalização podem-se otimizar as condições de operação, os procedimentos e a frequência de limpeza química (Sousa, 2003).

Os parâmetros requeridos para a normalização de um sistema de dessalinização via osmose inversa são:

- tempo de operação (excluindo as paradas programadas);

- temperatura e pressão de alimentação;
- salinidade da alimentação ou condutividade elétrica;
- diferencial de pressão (perda de carga que corresponde a diferença de pressão da alimentação e do concentrado);
- pressão do permeado, se o valor final disponível for adequado;
- fluxo final do concentrado e do permeado.

Com a normalização os dados do desempenho do sistema de osmose inversa são convertidos a:

- Vazão normalizada da água (Q_n);
- O gradiente médio de pressão efetivo da membrana, NDP;
- A passagem de sal normalizada (PSN).

1.4.1. Vazão do permeado normalizada (Q_n)

O gradiente médio de pressão efetivo da membrana (NDP) e a temperatura influenciam na permeabilidade da membrana. O NDP é função da pressão de alimentação, diferença de pressão, pressão osmótica e pressão do permeado. Com o aumento do valor do NDP, há uma tendência da membrana diminuir a produção do permeado. No entanto, com o aumento de temperatura, a membrana fica mais permeável, e a vazão do permeado aumenta. Estes fatos foram comprovados por Sousa (2003) que observou a mesma dependência da vazão do permeado normalizada com o NDP e a temperatura. Isso demonstra que a vazão está diretamente relacionada com a variação de pressão dos sistemas, a qual também depende da concentração dos componentes presentes na água de alimentação e da temperatura. Desse modo, esses parâmetros estão sujeitos a variar em função das condições climáticas onde se encontram instalados os sistemas de OI. O fator de correção de temperatura (FCT) correlata à mudança na vazão. A vazão normalizada é encontrada através da seguinte equação (Mallevialle *et al.*, 1996):

$$Q_n = Q_{pt} * \left(\frac{NDP_r}{NDP_t}\right) * \left(\frac{FCT_r}{FCT_t}\right) \qquad (13)$$

Onde:

Q_p = Vazão do permeado, m^3/h; NDP = Gradiente médio de pressão efetivo da membrana (kgf/cm^2); FCT = Fator de correção de temperatura (adimensional); Subscrito n = Indica

condição normalizada; Subscrito r = Indica condição de referência, ou seja, condição de projeto; Subscrito t = Indica condição atual, ou seja, condição de operação.

A Equação 14 mostra como pode ser determinado o valor do NDP. Todas as unidades são unidades de pressão (van de Lisdonk *et al.*, 2000).

$$NDP = P_a - \frac{\Delta P}{2} - \left(\frac{\pi_a + \pi_b}{2}\right) - P_p + \pi_p \qquad (14)$$

Onde: P_a = Pressão de alimentação (kgf/cm^2); ΔP = Diferencial de pressão (queda de pressão da corrente de água que passa sobre a membrana, em kgf/cm^2); π_a = pressão osmótica da alimentação (kgf/cm^2); π_P = pressão osmótica do permeado (kgf/cm^2); π_c = pressão osmótica do concentrado (kgf/cm^2); P_p = Pressão do permeado (kgf/cm^2).

A pressão osmótica pode ser estimada pela medida da condutividade elétrica e temperatura da água, mas existem fórmulas diferentes disponíveis na literatura. Uma aproximação válida e prática, dada em kgf/cm^2, é (Hydranautics, 2002):

$$\pi = \frac{C_{ac} * (T + 320)}{481506{,}61} \qquad \text{para } C_{ac} < 20000 \text{ mg/L} \qquad (15)$$

e

$$\pi = \frac{[(0{,}0117 * C_{ac}) - 34] * (T + 320)}{4814{,}43} \qquad \text{para } C_{ac} > 20000 \text{ mg/L} \qquad (16)$$

Onde: $C_{ac} = C_a * FC_{lm}$ é a média da concentração alimentação-concentrado, mg/L; FC_{lm} = Fator de concentração média logarítmica (adimensional); T = Temperatura, em °C; C_a = Sólidos totais dissolvidos na água de alimentação, mg/L.

O fator de concentração média logarítmica (FC_{lm}) é determinada pela Equação 17 (Hydranautics, 2002):

$$FC_{lm} = \frac{\ln \frac{1}{(1-r)}}{r} \qquad (17)$$

Onde: r = razão de recuperação do sistema.

Finalmente, o fator de correção de temperatura é dado pela Equação 18 (Taylor & Jacobs, 1996; Valley, 2003):

$$FTC = \exp\left[U * \left(\frac{1}{T_{ref} + 273} - \frac{1}{T + 273}\right)\right] \qquad (18)$$

Onde U é o fator de temperatura que depende do tipo de membrana. Para T $\geq$ T$_{ref}$ (25°C) U é igual à 2.640 e para T $\leq$ T$_{ref}$ (25°C) U é 3.480.

1.4.2. Sólidos totais dissolvidos do permeado normalizado (C_{Pn})

O STD normalizado pode ser determinado através da Equação 19 (Filmtec, 1998):

$$C_{Pn} = C_{Pt} * \left(\frac{NDP_t + \pi_{Pt}}{NDP_r + \pi_{Pr}} \right) * \left(\frac{C_{acr}}{C_{act}} \right) \quad (19)$$

Onde: C_p = Concentração do permeado, mg/L; NDP = Gradiente médio de pressão efetivo da membrana (kgf/cm^2); π_P = pressão osmótica do permeado (kgf/cm^2); C_{ac} = Média da concentração alimentação-concentrado, mg/L; Subscrito n = Indica condição normalizada; Subscrito r = Indica condição de referência, ou seja, condição de projeto; Subscrito t = Indica condição atual, ou seja, condição de operação.

Segundo Sousa (2003), a concentração de sais no permeado aumenta em função do tempo. Esse aumento está relacionado com o percentual da passagem de sais para o permeado durante o processo.

1.4.3. Passagem de sal normalizada (PS_n)

A passagem de sal de um sistema pode ser normalizada pela Equação 20 (Valley, 2003):

$$\%PS_n = \left(\frac{C_{pt}}{C_{ar}} \right) * \left(\frac{FCT_r}{FCT_t} \right) * \left(\frac{Q_{pr} C_{acr}}{Q_n C_{act}} \right) * 100 \quad (20)$$

Onde: $\%PS_n$ = Porcentagem da passagem de sal normalizada; C_p = Concentração do permeado, mg/L; C_a = Concentração de alimentação, mg/L; C_{ac} = Média da concentração alimentação-concentrado, mg/L; Q_P = Vazão do permeado, m^3/h; Q_n = vazão normalizada, m^3/h; FCT = Fator de correção de temperatura (adimensional); Subscrito n = Indica condição normalizada; Subscrito r = Indica condição de referência, ou seja, condição de projeto; Subscrito t = Indica condição atual, ou seja, condição de operação.

Sousa (2003) observou em seu trabalho que a passagem de sais através das membranas aumenta com o tempo de operação do sistema de osmose inversa. Segundo Sridhar *et al.* (2002), a diminuição da rejeição de sais é devido ao aumento da polarização de concentração do soluto, que aumenta a pressão osmótica na superfície da membrana e causa uma perda da pressão efetiva transmembrana.

1.4.4. Coeficiente de transferência de massa (CTM)

O coeficiente de transferência de massa representa o fluxo corrigido para a variação da temperatura e do gradiente médio de pressão efetivo da membrana (NDP). Quando a incrustação ocorre o fluxo de permeado diminuirá e/ou a pressão requerida de alimentação aumentará causando uma queda no CTM (van de Lisdonk *et al.*, 2000). Segundo estes pesquisadores, o coeficiente de transferência de massa é definido como:

$$CTM = \frac{Q_p * FCT}{A * NDP} \tag{21}$$

Onde: CTM = Coeficiente de transferência de massa, m/h.kgf.cm^{-2}; Q_P = Vazão do permeado, m^3/h; NDP = Gradiente médio de pressão efetivo da membrana, kgf/cm^2; FCT = Fator de correção de temperatura (adimensional); A = Área total das membranas, m^2.

1.5. Diagnóstico de incrustações em sistemas de dessalinização

Um controle eficaz da incrustação requer um bom diagnóstico do incrustante presente. Um conjunto de ferramentas tem sido desenvolvido para determinar o potencial de incrustação da água de alimentação de sistemas de osmose inversa A Tabela 2 apresenta as principais ferramentas que são aplicadas na prática para cada tipo de incrustação.

Tabela 2: Visão geral das ferramentas disponíveis para determinar o potencial de incrutação da água de alimentação e o correspondente diagnóstico de incrutação das membranas de OI.

Ferramenta	Diagnóstico de incrustação	Comentário
Diagnóstico integral (autópsia)	Biofouling, componentes inorgânicos e partículas	Diagnóstico de incrustante em elementos de membranas
Monitor de biofilme e AOC	Biofouling	Predição e prevenção de biofouling para determinar o (crescimento) potencial da água
SOCR	Biofouling	Método não-destrutivo para determinar a biomassa ativa em sistemas de membranas
MFI-UF	Particulado	Potencial de incrustação particulado da água
ScaleGuard	Scaling	Otimização da recuperação, dosagem de ácido e anti-incrustante

1.5.1. Diagnóstico integral (autópsia)

A autópsia consiste em um método de diagnótico em que o elemento de membrana é destruído para se analisar todos os tipos de incrustação incorporados no mesmo. Com isso, é feita uma análise integral do problema, podendo-se fazer observações microscópicas dos compostos inorgânicos e do biofouling presente nas membranas, o que possibilita fazer um diagnóstico mais completo e preciso do potencial de incrustação da água que alimenta uma determinada planta de osmose inversa.

Os problemas operacionais podem ser causados por outros tipos de incrustações do que os esperados. Por exemplo, os produtos químicos dosados na alimentação para prevenir o scaling podem representar um risco para favorecer o desenvolvimento do biofouling (Vrouwenvelder *et al.*, 2003). Desse modo, a autópsia é uma ferramenta que identifica incrustações inesperadas. A análise inclui os parâmetros biológicos para a quantificação da biomassa (ATP), a caracterização da biomassa, observações microscópicas e parâmetros químicos para determinar a presença de compostos inorgânicos (ICP-MS).

1.5.2. Potencial de Biofouling da água

A monitoração do potencial de biofouling da água de alimentação das plantas de osmose inversa pode ser executada com os parâmetros de biomassa (ATP e TDC) e com os parâmetros do potencial de crescimento como AOC, taxa de formação de biofilme (BFR) e potencial de produção de biomassa (BPP). Estes testes podem também ser usados para determinar o potencial de biofouling dos produtos químicos usados para prevenir o scaling tal como anti-incrustantes ou ácidos minerais (Huiting *et al.*, 2001). Determinados anti-incrustantes têm um maior risco de biofouling, comparado a outros, nas dosagens usadas normalmente na água de alimentação de uma planta de osmose inversa.

Segundo Vrouwenvelder & van der Kooij (2002), o biofouling pode ser impedido (i) reduzindo a concentração dos microorganismos e/ou reduzindo a concentração dos nutrientes pelo pré-tratamento e/ou (ii) efetuando limpezas preventiva/curativa. No entanto, é difícil remover a biomassa da membrana através de limpezas (Vrouwenvelder & van der Kooij, 2001). Uma combinação do pré-tratamento e da limpeza pode ser a maneira de impedir o biofouling.

1.5.3. A taxa específica de consumo de oxigênio (SOCR)

A taxa específica de consumo de oxigênio (SOCR) é um parâmetro para determinar a presença da biomassa ativa em sistemas de membrana pela medida da demanda de oxigênio. O SOCR é determinado com um método não-destrutivo (in situ). Segundo Vrouwenvelder *et al.* (2003). um sistema de osmose inversa sem problemas operacionais apresenta uma baixa demanda de oxigênio e baixa densidade de biomassa nos elementos de membranas, enquanto que sistemas com problemas operacionais apresentam uma elevada demanda de oxigênio e elevada densidade de biomassa.

De acordo com esses pesquisadores, após uma limpeza química realizada em um sistema de osmose inversa com o objetivo de remover o biofouling, o valor do SOCR mostrou a presença de biomassa ativa, o que indica que a limpeza não foi muito eficaz na inativação e na remoção da biomassa presente nos elementos de membrana antes da limpeza. Análises realizadas 22 dias após a limpeza determinaram que o SOCR mostrou um valor similar ao valor apresentado antes da limpeza. Isto indica que ocorreu um crescimento rápido dos microrganismos nos elementos da membrana. A incrustação não foi controlada aparentemente pela limpeza aplicada. Desse modo, o SOCR pode ser usado como um sistema de advertência da formação de biofouling inicial, possibilitando assim se chegar a um diagnóstico rápido.

1.5.4. MFI-UF

Os índices de incrustação usados geralmente para medir o potencial de incrustação de particulados da água de alimentação são o SDI e o $MFI_{0,45}$. Estes índices não incluem partículas coloidais menores. Recentemente, foi desenvolvido o MFI usando membranas de ultrafiltração em substituição ao papel de filtro (Huiting *et al.*, 2001). O MFI-UF foi desenvolvido para ser uma ferramenta promissora para medir o potencial de incrustação de particulados da água de alimentação. Além disso, pode ser usado para avaliar a eficiência de processos de pré-tratamentos para a remoção de colóides e de material particulado.

1.5.5. ScaleGuard

Segundo van de Lisdonk (2000), uma ferramenta desenvolvida por Kiwa é o ScaleGuard®. Este monitor de scaling consiste de um elemento de membrana alimentado com o concentrado de um planta piloto ou de uma planta no campo. O ScaleGuard® pode, se

instalado corretamente, detectar o scaling antes dele ocorrer no último arranjo de uma planta de incrustação completa, sendo possível:

- Otimizar a operação de uma planta de incrustação completa, isto é, a recuperação e a dose de anti-incrustante ou ácido necessário;
- Identificar a natureza do scaling sem a necessidade de interromper a operação da planta;
- Otimizar o procedimento e a freqüência da limpeza química.

1.5.6. Prevenção de incrustação de $CaCO_3$

Os cálculos de formação de incrustação são ferramentas utilizadas a fim de determinar se um sal solúvel apresenta algum potencial de incrustação em um sistema de osmose inversa e, desse modo, auxiliar no diagnóstico de possíveis problemas no desempenho do sistema. A avaliação do risco de formação de precipitados de carbonato de cálcio e de sílica é mais complexa devido à influência do pH na estrutura química dos componentes destes sais.

Os índices de saturação são usados extensamente no tratamento de água na predição da formação de incrustação e como base de ajuste para as condições e controle de operação, tais como o pH e o pré-tratamento. Estes índices incluem o Índice de Saturação de Langelier (ISL) e o índice do Saturação de Stiff-Davis (S&DS1) para a predição de incrustação do carbonato de cálcio. Outros índices foram introduzidos mais tarde para outros tipos de incrustações mais comuns (Sulfato de Ca, Ba, Sr e sílica) que são baseados na solubilidade específica (El-Manharawy & Hafez, 2001).

O risco de formação de precipitados de carbonato de cálcio em águas salobras com STD até 10.000 mg/L é avaliado através do valor do Índice de Saturação de Langelier (ISL), enquanto que, o Índice de Estabilidade de Stiff e Davis (S&DSI) é utilizado em água de salinidades altas. Os dois índices são calculados pela mesma fórmula (Equação 22), mas diferem no fator de correção da salinidade, que no ISL é baseado no STD da solução e no S&DSI, no poder iônico da solução.

$$ISL = pH - pH_s \tag{22a}$$

$$S\,\&\,DSI = pH - pH_s \tag{22b}$$

onde ISL: Índice de Saturação de Langelier; S&DSI: Índice de Estabilidade de Stiff e Davis; pH: pH da alimentação; pH_s: pH no qual a alimentação fica saturado com $CaCO_3$.

O pH_s é calculado pelas seguintes equações (Mindler & Epstein, 1986; Ning & Netwig, 2002):

$$pH_s = (9{,}3 + A + B) - (C + D) \quad (23)$$

Onde:

$$A = \frac{(Log[STD] - 1)}{10} \quad (24)$$

$$B = -13{,}12 * Log[T + 273] + 34{,}55 \quad (25)$$

$$C = Log[Ca^{+2}\ como\ CaCO_3] - 0{,}4 \quad (26)$$

$$D = Log[Alcalinidade\ como\ CaCO_3] \quad (27)$$

Para evitar a precipitação de carbonato de cálcio, os índices ISL e S&SDI da água de alimentação devem ser negativos. Nesse caso, o pH da água está abaixo do pH de saturação calculado e a água tem um potencial muito limitado de incrustação. Se o pH exceder o pH_s, o ISL é positivo, e sendo supersaturada com $CaCO_3$, a água tem uma tendência para formar incrustação. Ao aumentar o valor positivo do índice, o potencial de incrustação aumenta. Quando é feita a correção do pH ou o adicionamento de anti-incrustante, os valores de ISL e S&SDI devem ficar abaixo dos valores estipulados para esses tratamentos, geralmente entre 1 e 1,5 (Schneider & Tsutiya, 2001). No caso do ISL igual a zero não haverá precipitação nem dissolução do $CaCO_3$, ou seja, a água está em equilíbrio com o carbonato de cálcio

O ISL e o S&SDI são usados por alguns fabricantes de membranas de OI para guiar o uso de produtos químicos no pré-tratamento da água de alimentação (Tabela 3):

Tabela 3: Valores limites para o ISL e S&DSI (Ning & Netwig, 2002).

	Hydranautics	Permasep	FilmTec
ISL e S&DSI, água sem inibidor de incrustação	< - 0,2	< 0,0	< 0,0
ISL e S&DSI, água com inibidor SHMP*	≤ 0,5	< 1,0	< 1,0
ISL e S&DSI, água com inibidor orgânico	≤ 1,8	< 2,3	< 1,8

* SHMP = inibidor de incrustação a base de hexametafosfato de sódio

1.5.7. Prevenção de incrustação de $CaSO_4$, $BaSO_4$ e $SrSO_4$

O potencial de incrustação de $CaSO_4$ é determinado comparando-se o valor do produto iônico da H_2O do concentrado Ip_c, com o produto de solubilidade, Ks_c, do concentrado nas mesmas condições. Segundo Kim *et al.* (2002), utilizam-se as seguintes equações:

$$Ip_c = \left[\left(Ca^{+2}\right)_a * \left(SO_4^{-2}\right)_a\right] * FC \quad (28)$$

$$Ks_c = \gamma_{Ca}\left[Ca^{+2}\right] * \gamma_{SO_4}\left[SO_4^{-2}\right] \quad (29)$$

Onde: $[Ca^{+2}]$ e $[SO_4^{-2}]$: concentração molar do respectivo componente; γ: coeficiente de atividade dos componentes do sal na solução e FC: fator de concentração (Equação 5).

Os coeficientes de atividade γ são unitários para solução com baixas concentrações de sais (água doce). Em águas salobras e salinas, estes coeficientes são corrigidos, por exemplo, através da fórmula de Debye-Hückel, que determina a relação entre o logaritmo do coeficiente de atividade, as cargas elétricas dos componentes do sal (z,y) e a força iônica da solução (I) (Stumm & Morgan, 1996; Van de Lisdonk, 2000).

$$\log \gamma_{ab} = -0{,}509 zy\sqrt{I} \quad (30)$$

$$I = \frac{1}{2}\sum\left(m_i z^2\right) \quad (31)$$

Se IPc ≥ Ks_c; ocorre formação de incrustação e um ajuste é requerido.

O cálculo do potencial de incrustação de sulfato bário e sulfato de estrôncio é análogo ao procedimento escrito para o sulfato de cálcio. Para o sulfato de estrôncio, se o IPc ≥ $0.8Ks_c$ ocorre formação de incrustação e um ajuste é requerido.

1.5.8. Prevenção de incrustação de Sílica

A incrustação de sílica nas membranas pode ocorrer por um dos três mecanismos: deposição monomérica na superfície da membrana, deposição coloidal por meio de colóides que se formam na solução e se acumulam na superfície da membrana, e sílica amorfa biogênica (Sheikholeslami & Bright, 2002). A concentração máxima permissível de SiO_2 no concentrado é baseada na sua solubilidade. Faz-se uma análise do concentrado para saber o teor de sílica e em seguida compara com o valor da solubilidade da sílica dada por:

$$SiO_{2corr} = Ks_{SiO_2} * pH_{corr} \quad (32)$$

onde o pH_{corr} é a correção do pHs, que é o pH do concentrado, para SiO_2 e é determinado através de gráficos encontrados em manuais de membranas.

Se $SiO_{2c} > SiO_{2corr}$, ocorrerá incrustação de sílica e um ajuste é requerido.

1.5.9. Diagnóstico de incrustações através dos parâmetros normalizados

Como comentado nos ítens anteriores, existem um conjunto de ferramentas que estão disponíveis para estimar o potencial de incrustação da água de alimentação. Para uma

operação produtiva de um sistema de osmose inversa é essencial uma indicação antecipada da incrustação e uma boa identificação do seu tipo. Para isso, é necessário uma normalização dos dados operacionais que, em combinação das ferramentas apropriadas, permite aos usuários e aos operadores de sistemas de osmose inversa melhorar a eficiência da operação e da limpeza de sua planta.

Segundo Huiting *et al.* (2001), apesar de todas as ferramentas e pré-tratamentos, a incrustação ocorrerá. Para poder executar um projeto de limpeza bem feito (ao invés de tentativas e erros), é essencial saber o tipo de incrustação, pois certos tipos são irreversíveis se detectados em um estágio avançado (biofouling e scalling de sílica). Isto é importante para identificar a incrustação o mais breve possível. Os diferentes tipos de incrustações têm efeitos diferentes nos dados normalizados (CTM, NDP e PSN) como ilustrado na Tabela 4.

Tabela 4: Identificação da incrustação pelo impacto no desempenho.

Incrustação	CMT	NDP	PSN	Localização
Scaling	↓	↑	⇑	Último arranjo
Óxidos metálicos	⇓	⇑	⇑	Primeiro arranjo
Colóides/sílica	↓	↑	↑	Primeiro / último
Orgânicos	↓	↑	--	Primeiro arranjo
biofouling (rápido)	⇓	⇑	--	Primeiro arranjo

↓ / ↑ Pequena a moderada mudança
⇓ / ⇑ Rápida / significante mudança

Os dados normalizados quando representados graficamente mostram não somente a condição instantânea do sistema de membrana, mas também o histórico da operação. A estabilidade de cada deterioração (pequena/moderada a rápida/significante) pode ser observada nos gráficos.

As incrustações ocorrem quando os parâmetros normalizados apresentarem as seguintes modificações em relação aos dados de partida da planta ou aos dados de referência escolhidos inicialmente (Huiting *et al.*, 2001; Amjad, 1992):

1. O gradiente médio de pressão efetivo da membrana aumentar 10-15%;
2. A passagem de sal normalizada (PS_n) ter um aumento significativo ao longo do tempo (em torno de 50%);
3. O coeficiente de transferência de massa (CTM) diminuir 10-15%.

A Tabela 5 mostra uma série de ações corretivas que devem ser tomadas em função do tipo de incrustação diagnosticada.

Tabela 5: Ações corretivas em função do tipo de incrustação (Huiting *et al*, 2001; Amjad, 1992).

Incrustação	Ferramentas para confirmação	Ações corretivas
Scaling / Óxidos metálicos	Monitor ScaleGuard. Análise de íons metálicos na solução de limpeza. Verificar o ISL do rejeito. Calcular a solubilidade máxima do $CaSO_4$, $BaSO_4$ e SiO_2 na análise do rejeito.	Aumentar a adição de ácido e anti-incrustante para o $CaCO_3$ e $CaSO_4$. Reduzir a recuperação. Limpeza com ácido cítrico ou solução baseada em EDTA para $CaCO_3$, $CaSO_4$ e $BaSO_4$. Limpeza de incrustantes a base de silicatos com soluções a base de fluoreto de amônia. Otimizar o pré-tratamento para remover metais.
Colóides / Sílica	Calcular o SDI e/ou MFI-UF da alimentação. Difração de raios-X. Análise da solução de limpeza e reciclo.	Otimizar o sistema de pré-tratamento para remoção de colóides, incluindo a filtração. Estabilização da carga. Limpeza com EDTA, STP ou detergentes do tipo BIZ com pH elevado. Baixar a recuperação do sistema. Limpeza de incrustantes a base de silicatos com soluções a base de fluoreto de amônia
Orgânicos	Teste destrutivo, isto é, análises de reflexão do IR (infra-vermelho).	Otimização do sistema pré-tratamento com processos de coagulação e filtração. Tratamento com carvão ativado ou resina. Limpeza com detergente a elevado pH ou com solução de isopropanol.
Biofouling	Monitor de biofilme, AOC e SOCR. Contagem de bactérias no permeado e no rejeito. Verificação de formação de lodo nos tubos e vasos.	Adição de bissulfeto de sódio. Cloração com ou sem filtração por carvão ativado. Limpeza com soluções a base de EDTA ou detergentes do tipo BIZ a pH elevado. Desenvolver programas de desinfecção com formaldeído, peróxido de hidrogênio, etc.

A ação corretiva da incrustação geralmente mais usada é a limpeza química do sistema. Entretanto, sem a otimização do pré-tratamento, a incrustação retornará. Em baixas freqüências de limpeza (menos de uma vez a cada trimestre) a limpeza corretiva é prática. Em

freqüências de limpeza maiores pode ser um custo útil fazer ações preventivas, tais como melhoramento do pré-tratamento (incrustação coloidal), seleção de um anti-incrustante mais apropriado (no caso do scaling) ou reduzir a recuperação (como por exemplo no caso de scaling de sílica).

Com a normalização apropriada em combinação com uma avaliação freqüente dos dados e um registro de todos os eventos e mudanças na operação, os operadores podem adquirir uma melhor compreensão dos efeitos climáticos, da eficiência das limpezas, etc.. Isto conduz a uma melhor consciência de operação e a regimes de limpeza otimizados em que é alcançada uma operação mais estável do sistema de osmose inversa.

As experiências com a osmose inversa, incluindo o desenvolvimento de estratégias de monitoração mais exatas e de novas ferramentas de estimação, conduz a um melhor critério de diagnóstico da incrustação e também de como reagir a ela sem a necessidade de parar a produção (Huiting *et al.*, 2001).

1.6. Programação orientada a objetos

A programação orientada a objetos (POO) é o paradigma de programação predominante dos dias atuais, tendo substituído as técnicas de programação baseadas em procedimentos, "estruturadas", que foram desenvolvidas nos anos 70.

A programação tradicional estruturada consiste em projetar um conjunto de funções (algoritmos) para resolver um problema. O passo seguinte convencional é encontrar a forma apropriada de armazenar os dados. Logo, primeiro decide-se como manipular os dados; depois, que estrutura impor aos dados a fim de tornar mais fácil o processamento. A POO inverte essa ordem e coloca as estruturas de dados em primeiro lugar, examinando depois os algoritmos que vão processar os dados.

A chave para ser mais produtivo em POO é tornar cada objeto responsável pela realização de um conjunto de tarefas relacionadas. Se um objeto depender de uma tarefa que não seja de sua responsabilidade, ele vai precisar ter acesso a um objeto cujas responsabilidades incluem essa tarefa.

Em particular, um objeto nunca deveria manipular diretamente os dados internos de outro objeto. Toda a comunicação deve ser através de 'mensagens', ou seja, chamadas a métodos. Ao projetar um objeto para lidar com todas as mensagens apropriadas e manipular os dados internamente, maximiza-se a reutilização, reduz-se a dependência de dados e minimiza-se o tempo de depuração (Horstmann e Cornell, 2001a). Evidentemente, não é

desejável que um objeto individual faça coisas demais de uma só vez. Segundo estes autores, tanto o projeto quanto a depuração são simplificados quando se formam objetos pequenos que realizam algumas poucas tarefas em vez de objetos enormes com dados internos extremamente complexos, com centenas de funções para manipular os dados.

Para isso, é necessário "organizar" as funções de um objeto em classes. Uma classe é geralmente descrita como o modelo ou a forma a partir da qual um objeto é criado. Isso leva ao modo padrão de pensar sobre as classes: como sendo as fôrmas com as quais se fazem biscoitos. Os objetos são os biscoitos. A "massa do biscoito", na forma de memória, também terá de ser providenciada. A linguagem Java, por exemplo, é ótima para esconder as etapas da "preparação da massa dos biscoitos". E só usar a palavra-chave *new* para obter memória e o sistema de coleta de lixo interno irá comer os biscoitos que ninguém mais quer. Quando se cria um objeto a partir de uma classe, diz-se que *foi criada uma instância* da classe. Quando se tem uma instrução como: `Normalizacao n = new Normalizacao();` internamente o operador será usado para criar uma nova *instância* da classe Normalizacao.

Tudo o que se escreve em Java está dentro de uma classe. Segundo Horstmann e Cornell (2001a), a biblioteca Java padrão fornece centenas de classes para diversos propósitos tais como projeto de interface de usuário e programação em rede. Apesar de tudo, é necessário que o programador crie suas próprias classes em Java para descrever os objetos dos domínios do problema de seus aplicativos e adaptar as classes que são fornecidas pela biblioteca padrão para atender a seus próprios objetivos.

As classes podem ser formadas a partir de outras classes. Uma classe que é formada a partir de outra classe a *estende*. Ao estender uma classe base, a nova classe inicialmente tem todas as propriedades e métodos de seu progenitor. Pode-se escolher então entre modificar ou simplesmente manter qualquer método da progenitora e também adicionar novos métodos que aplicam-se somente à classe filha. O conceito genérico de estender uma classe base é chamado de *herança*.

O *Encapsulamento* é outro conceito-chave ao se trabalhar com objetos. Formalmente, o encapsulamento não é nada mais que combinar dados e comportamento em um pacote e ocultar os dados e métodos do usuário do objeto. Os dados em um objeto são geralmente, chamados de *variáveis de instância ou campos*; e as funções e procedimentos de uma classe Java são chamados *métodos*. Um objeto específico que é uma instância de uma classe terá valores específicos em seus campos que definem seu *estado* atual (Horstmann e Cornell, 2001a).

Para se fazer o encapsulamento funcionar é necessário fazer programas que dificilmente tenham acesso direto às variáveis de instância (campos) de uma classe. Os programas precisam interagir com seus dados somente através dos métodos do objeto. O encapsulamento é a forma de se dar aos objetos seu comportamento de "caixa preta", característica-chave para a reutilização e a confiabilidade. Isso significa que um objeto pode mudar totalmente como ele armazena seus dados, mas enquanto continuar usando os mesmos métodos para processar os dados, nenhum outro objeto vai saber disso ou se importar com isso.

As relações mais comuns entre as classes são uso, inclusão ("tem-um") e herança ("é-um"). A relação uso é a mais óbvia e também a mais genérica. Por exemplo, uma classe Pedido usa uma classe Conta, já que os objetos Pedido precisam acessar os objetos Conta para verificar o crédito. Mas a classe Item não usa a classe Conta pois os objetos Item nunca precisam se preocupar com as contas dos clientes. Assim, uma classe usa outra classe se ela manipula objetos dessa classe (Horstmann e Cornell, 2001a). De um modo geral, uma classe A usa uma classe B se um método de A envia uma mensagem para um objeto da classe B, ou se um método de A cria, recebe ou retorna objetos da classe B.

A inclusão significa que objetos da classe A contêm objetos da classe B. Evidentemente, a inclusão é um caso especial de uso; se um objeto A contém um objeto B, então pelo menos um método da classe A irá usar esse objeto da classe B.

A relação de *herança* denota especialização. Por exemplo, uma classe PedidoExpresso será herdeira de uma classe Pedido. A classe especializada PedidoExpresso tem métodos especiais para lidar com prioridades e um método diferente para calcular as taxas de remessa, enquanto que os outros métodos, como adição de itens e cobrança serão simplesmente herdados da classe Pedido. Em geral, se a classe A estender a classe B, a classe A vai herdar métodos da classe B e ter recursos a mais.

1.7. Java como ferramenta de programação

Java é uma linguagem de programação orientada a objetos e foi produzido pela Sun Microsystems no início da década de 90. Ganhou uma grande popularidade em pouco tempo devido a sua ampla utilização no desenvolvimento de ambientes web. Esta linguagem possui uma ampla biblioteca de rotinas para lidar facilmente com ambiente em rede, banco de dados, e outros.

Além das várias vantagens dessa linguagem tais como orientação a objetos, robusta, simples e livre, Java possui uma biblioteca em tempo de execução que visa proporcionar independência de plataforma: é possível usar o mesmo código nos sistemas Windows 95/98/NT, Solaris, Unix, Macintosh etc. Isto certamente é necessário para a programação na Internet (Horstmann & Cornell, 2001a).

Outra vantagem de programação é que a linguagem Java tem uma sintaxe similar à C++. Esta linguagem é totalmente orientada a objetos - mais ainda do que o C++. Tudo, em Java, com exceção de alguns poucos tipos básicos como números, são objetos. (O projeto orientado a objetos substituiu técnicas estruturadas anteriores porque ele tem muitas vantagens ao lidar com projetos mais sofisticados).

Entretanto, ter outro dialeto de C++, mesmo que de certa forma melhorado, ainda não é suficiente. O ponto chave é este: é muito mais fácil elaborar código sem erros usando Java do que usando C++ (Horstmann & Cornell, 2001a).

Os projetistas da linguagem Java pensaram muito sobre o que faz o código C++ tão sujeito a erros e acrescentaram recursos que eliminam a possibilidade de se criar código com os tipos mais comuns de erros. Algumas estimativas afirmam que, por alto, a cada 50 linhas de código C++ há pelo menos um erro. Para diminuir a existência de erros, os desenvolvedores da linguagem Java (Horstmann & Cornell, 2001a):

- Eliminaram a alocação e liberação manuais de memória. A memória em Java tem coleta de lixo automática. Não é necessário se preocupar com falta de memória.
- Introduziram arrays verdadeiros e eliminaram a aritmética de ponteiros. Não é necessário se preocupar com a sobrescrita de uma área de memória-chave devido a um erro simples de aritmética ao se trabalhar com um ponteiro.
- Eliminaram a possibilidade de se confundir uma atribuição com um teste de igual em uma condição. Não se pode nem mesmo compilar if (ntries = 3) ....
- Eliminaram a herança múltipla, substituindo-a por uma nova noção de interface retirada do Objective C. As interfaces oferecem o que se quer da herança múltipla, sem a complexidade que surge no controle de sua hierarquia.

1.8. Java Database Connectivity (JDBC)

O kit Java Database Connectivity permite aos programadores Java conectar-se a bancos de dados, fazer consultas ou atualizações usando a linguagem de consulta padrão do mercado, o SQL.

O JDBC consiste de duas camadas. A camada superior é a API JDBC. Essa API comunica-se com a API do gerenciador de driver JDBC, enviando-lhe as várias declarações SQL (Figura 1.3). O gerenciador de driver comunica-se com os vários drivers de terceiros que estão no momento conectados ao bando de dados e retorna a informação da consulta ou executa a ação especificada pela query (Horstmann & Cornell, 2001b).

Os Drivers JDBC são classificados em 4 tipos:

- Um driver tipo 1 traduz JDBC em ODBC e confia no driver ODBC para se comunicar com o banco de dados. A Sun inclui um driver assim, a ponte JDBC/ODBC com o JDK. A ponte é útil para testes, mas não é recomendada para uso industrial.
- Um driver tipo 2 é um driver, escrito uma parte em Java e a outra em código nativo, que se comunica com a API cliente de um banco de dados. Quando se usa tal driver, é preciso instalar algum código específico de plataforma junto com a biblioteca Java.

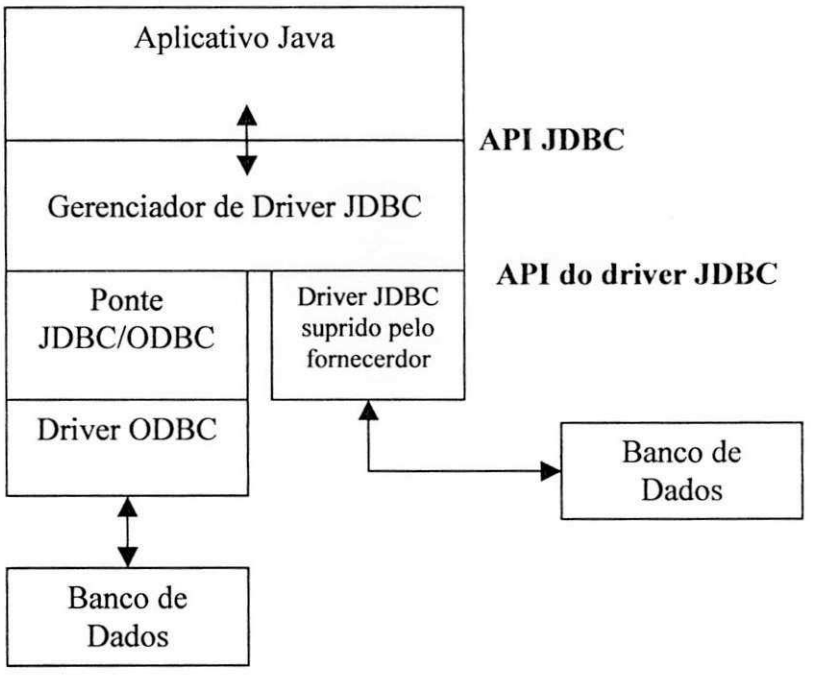


Figura 1.3: JDBC e a comunicação ao banco de dados

- Um driver tipo 3 é uma biblioteca cliente de puro Java que usa um protocolo independente de banco de dados para comunicar requisições de banco de dados a um componente servidor, que então traduz as requisições em protocolos de banco de dados específicos. A biblioteca cliente é independente do banco de dados, simplificando o desenvolvimento.
- Um driver tipo 4 é uma biblioteca em puro Java que traduz requisições JDBC diretamente ao protocolo específico do banco de dados.

A maioria dos fabricantes de banco de dados fornece um driver tipo 3 ou 4 com seus bancos de dados (Horstman & Cornell, 2000).

1.9. Servlets e Java Server Pages

Um servlet é uma classe escrita em Java cujos objetos têm a finalidade de gerar documentos codificados em HTML. Esta característica dos servlets implica em que um web designer precisa conhecer Java para poder construir as páginas de uma aplicação. Esta limitação é superada pela tecnologia de JSP.

As páginas Java Server Pages (JSP) é uma tecnologia desenvolvida pela Sun Microsystems como resposta a aparição das páginas ASP (Active server Pages) por parte da Microsoft. Uma página escrita em JSP (arquivos com extensão .jsp) é uma página escrita em HTML e que contém pequenos fragmentos de código Java e/ou tags especiais (definidos na especificação JSP). Com estes tags, o web designer não necessita escrever uma única linha de código Java. Ao contrário do que acontece com a tecnologia ASP, o analista/programador pode criar (em Java) os seus próprios tags customizados para a aplicação em desenvolvimento.

A tecnologia JSP está relacionada com outra tecnologia, os servlets, pois uma página JSP é automaticamente transformada em servlet e o servlet executa no servidor para gerar a resposta (Figura 1.4). Este fato simplifica a geração de conteúdo dinâmico porque os Web Designers podem manipular as páginas com mais facilidade do que os servlets (Jacques, 2002).

Do ponto de vista operacional, a principal finalidade das tecnologias de servlets e JSP é permitir a criação dinâmica de conteúdos. A dinâmica, em um cenário típico, funciona do seguinte modo (Figura 1.4): Quando uma requisição é mapeada a uma página JSP, o container verifica se o servlet correspondente à página é mais antigo que a página (ou se não existe).

Se o servlet não existe ou é mais antigo, a página JSP será compilada para gerar novo servlet. Em seguida, a requisição é repassada ao servlet. Se o servlet está atualizado, a requisição é redirecionada para ele. Deste ponto em diante, o comportamento equivale ao ciclo de vida do servlet, mas os métodos são diferentes. Se o servlet ainda não estiver na memória, ele é instanciado, carregado e seu método jspInit() é chamado. Para cada requisição, seu método jspService(req, res) é chamado. Ele é resultado da compilação do corpo da página JSP. No fim da vida, o método jspDestroy() é chamado (Argonavis, 2003).

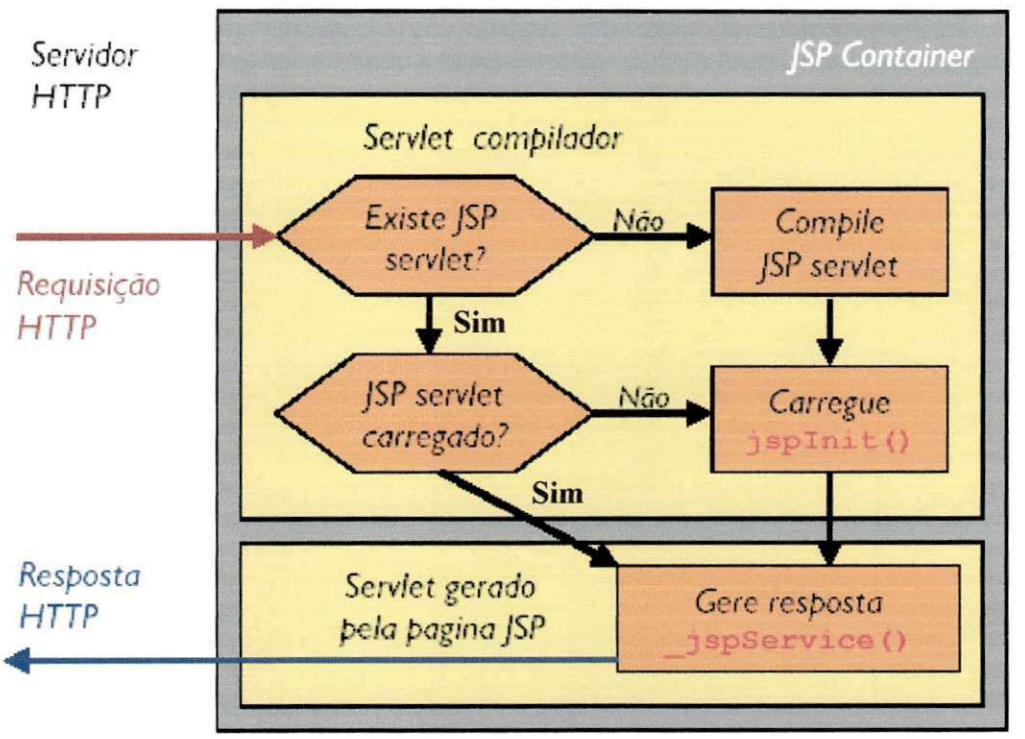


Figura 1.4: Ciclo de vida de uma página JSP

Como no caso dos servlets, para se criar páginas JSP é necessário instalar junto com o servidor WEB um "motor adicional" que permita a execução desse tipo de páginas. Como comentado no ítem anterior, um produto gratuito que segue a linha marcada pelo Java utilizado para exibir as páginas HTML dinâmicas é o servidor Tomcat do projeto Jakarta (http://jakarta.apache.org).

O enlace entre uma página JSP e seu correspondente servlet se realiza por meio de duas classes contidas dentro do pacote javax.servlet.jsp (uma classe dentro de um pacote destinado à manipulação de servlets): JSPPage e HttpJspPage que são as classes que permite criar a interface da JSP compilada, ou seja, do servlet (WMLClub, 2003).

1.10. O software Tomcat

O software Tomcat, desenvolvido pela Fundação Apache, permite a execução de aplicações para web. Sua principal característica técnica é estar centrada na linguagem de programação Java, mais especificamente nas tecnologias de Servlets e de Java Server Pages (JSP). Esta abordagem rivaliza, por exemplo, com a usada pela Microsoft com o ASP (baseada na linguagem Visual Basic).

A Fundação Apache, mais conhecida pelo seu servidor web de mesmo nome, permite, como no caso do servidor Apache, que o Tomcat seja usado livremente, seja para fins comerciais ou não.

O Tomcat está escrito em Java e, por isso, necessita que a versão Java 2 Standard Edition (J2SE) esteja instalada no mesmo computador onde ele será executado. No entanto, não basta ter a versão runtime de Java instalada, pois o Tomcat necessita compilar (e não apenas executar) programas escritos em Java. O projeto Jakarta da Fundação Apache, do qual o subprojeto Tomcat é o representante mais ilustre, tem como objetivo o desenvolvimento de soluções código aberto baseadas na plataforma Java.

O desenvolvimento de uma típica aplicação web a ser executada pelo Tomcat implica no domínio das seguintes linguagens:

a) Java: Todos os algoritmos da aplicação devem ser escritos em Java.

b) HTML: A interface da apresentação, isto é, aquilo que o usuário vê em sua tela, é construída na forma de páginas escritas em HTML e visualizada através do browser do usuário. Esta tarefa normalmente é delegada ao web designer.

c) XML: Qualquer aspecto relacionado à configuração da aplicação deve ser expresso por meio da linguagem XML em um arquivo chamado web.xml. Os dados de configuração podem ser usados tanto pelo Tomcat como pela aplicação.

O servidor Tomcat tem a habilidade de converter automaticamente qualquer página JSP em um servlet equivalente. Em outras palavras, o Tomcat é capaz de criar código fonte Java a partir de um documento HTML.

Do ponto de vista técnico, Tomcat é a implementação referência das especificações das tecnologias de servlets e JSP criadas pela Sun. A versão 4.0.x do Tomcat implementa as especificações Servlet 2.3 e JSP 1.2 (que são as mais recentes). Várias empresas, como Borland, IBM, BEA, etc., também oferecem suas implementações das duas especificações da Sun.

Em termos práticos, o Tomcat pode ser usado isoladamente, assumindo o papel de um servidor web, ou em conjunto com outro servidor (como o Apache). Neste caso, o Apache atende a requisições de páginas estáticas enquanto que o Tomcat atende a requisições de páginas dinâmicas.

Outra forma de usar o Tomcat é como parte da versão J2EE (Enterprise Edition) de Java para a criação de servidores de aplicação. Este é o caso, por exemplo, do servidor de aplicação JBoss.

CAPÍTULO II

Metodologia

2.1. Sistema de dessalinização piloto (SDP)

O módulo de gerência com monitoração remota foi implementado com a linguagem de programação JAVA, da Sun Microsystems™, e de acordo com a prototipagem evolutiva, ou seja, a cada avanço do projeto foi feito uma demonstração e uso por parte do cliente para que o mesmo possa sugerir melhorias e novos requisitos (Almeida, 2002).

Para introduzir as equações do processo no módulo de gerência é necessário entender o funcionamento do Sistema de Dessalinização Piloto (SDP) e, principalmente, conhecer as variáveis de controle desse sistema. Além disso, é importante o conhecimento das equações do processo de dessalinização via osmose inversa para incrementá-las ao programa, realizando desse modo uma normalização do sistema a partir dos dados enviados pelo computador embarcado durante a sua operação.

Conforme mostra a Figura 2.1, o SDP é composto dos seguintes componentes: uma bomba de alta pressão de 1,5 CV monofásica, uma bomba dosadora de anti-incrustante, três

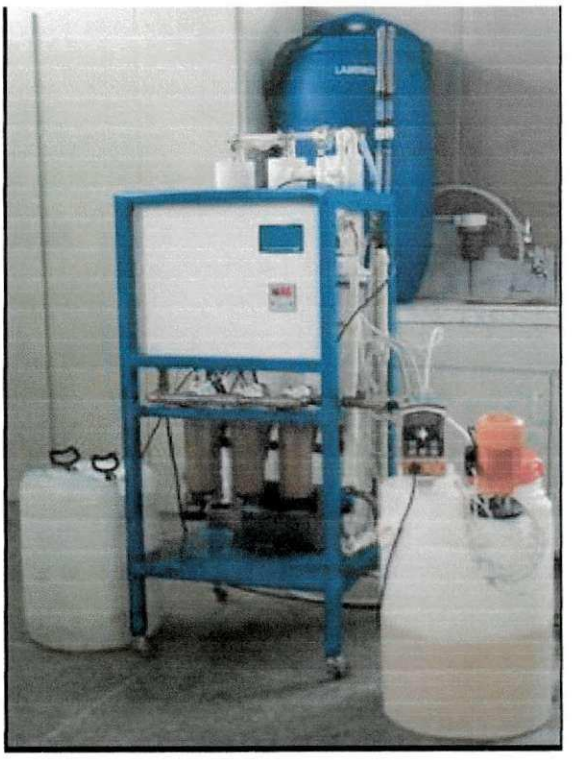


Figura 2.1: Sistema de Dessalinização Piloto (SDP)

elementos de filtros de acetato de celulose de 5 μm, três elementos de membranas do tipo BW30-4040, três vasos de alta pressão de 1 metro, dois rotâmetros, dois manômetros, um quadro elétrico de comando, quatro sensores de pressão, dois sensores de vazão, um sensor de temperatura e de pH, um computador embarcado e acessórios (uma estrutura metálica, quatro válvulas de esfera de controle de vazão, um tanque de 40 litros para solução anti-incrustante, etc.).

2.2. O sistema de monitoração de dessalinizadores

O desenvolvimento do sistema de monitoração consiste, principalmente, em monitorar as pressões de entrada (p3) e saída (p4) das membranas, as vazões do permeado (v2) e do concentrado (v1), a temperatura (T) e o pH da solução que alimenta o sistema de dessalinização, pois estas variáveis exercem maior influência no seu desempenho. As pressões de entrada (p1) e saída (p2) dos filtros também foram monitoradas devido à importância dos mesmos no processo.

As condições de alarme adotadas para o monitoramento são as seguintes:

1. Quando a pressão de entrada (p1) e de saída (p2) dos filtros apresentar uma diferença entre 15% e 20% da diferença de operação inicial, é gerado um alarme amarelo. Se o aumento do ΔP dos filtros for maior ou igual a 20%, ocorre um alarme vermelho;
2. Quando a pressão de entrada (p3) e de saída (p4) das membranas apresentar uma diferença entre 10% e 15% do valor de operação inicial, ocorre um alarme amarelo. Se o aumento do ΔP das membranas for maior ou igual a 15%, é gerado um alarme vermelho;
3. Quando a vazão do permeado (v2) apresentar uma diferença entre 10 e 20% do valor de operação inicial, ocorre um alarme amarelo. Se esta diferença for maior ou igual a 20%, é gerado um alarme vermelho;
4. Quando a temperatura da alimentação das membranas atingir um valor entre 40°C e 45°C, é gerado um alarme amarelo. Se a temperatura for maior ou igual a 45°C, ocorre um alarme vermelho.

O sistema de monitoração remota de dessalinizadores pode ser explicado como mostra a Figura 2.2. O dessalinizador possui um conjunto de sensores que enviam os valores das variáveis de medidas através do computador embarcado.

O sensor é um hardware conectado ao computador embarcado através da porta paralela. O computador embarcado armazena e analisa as medidas de pressão, vazão, temperatura e pH, verificando se os valores estão na condição de alarme e, em seguida, as

envia para a máquina servidor através da linha telefônica. Na máquina servidor é verificado novamente a ocorrência de alarme e, em seguida, os valores são armazenados no banco de dados. Através de páginas web, o servidor disponibiliza todos os dados armazenados no banco de dados na Internet. Dessa forma, o gerente pode monitorar os dessalinizadores de qualquer parte do mundo.

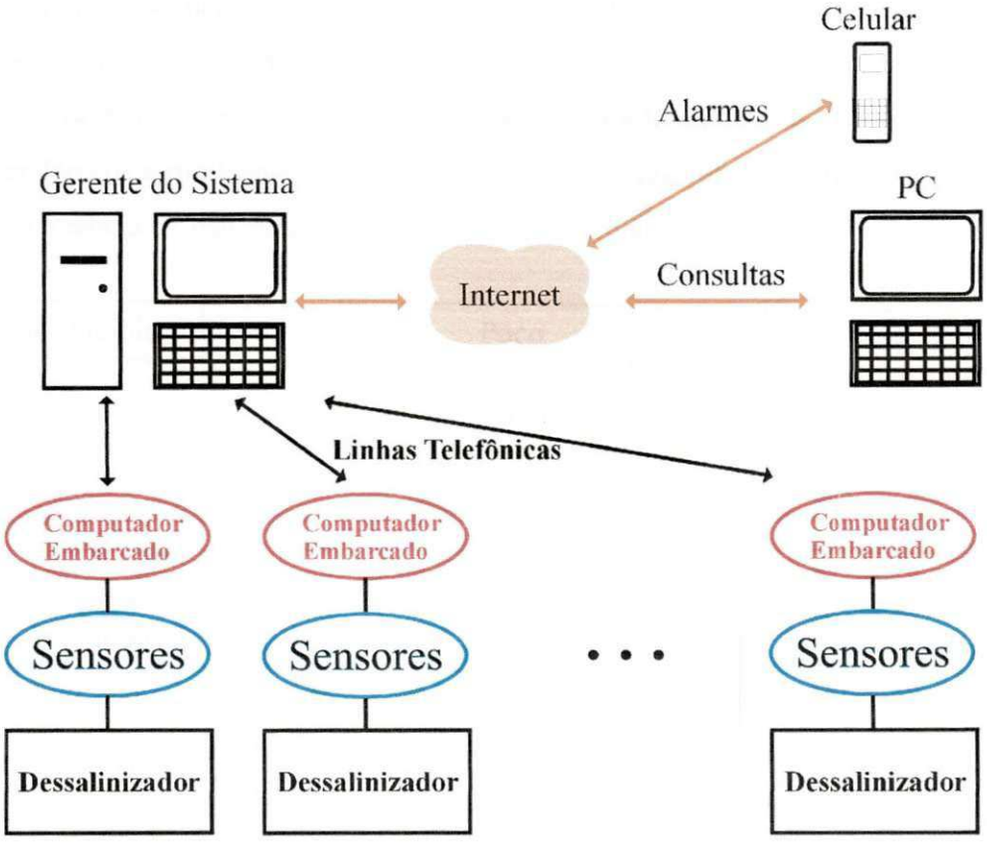


Figura 2.2: Sistema de monitoração de dessalinizadores

2.3. O módulo de gerência

Esta parte do projeto refere-se à implementação das equações do processo de dessalinização no módulo de gerência, ou seja, fazer com que o servidor seja capaz de realizar a normalização do SDP. No entanto, para entender como estas equações foram adicionadas ao software servidor é necessário saber como todo o módulo de gerência foi implementado. O sensor, o computador embarcado e a conexão com servidor são questões tratadas em outros projetos.

O padrão SQL (Structured Query Language) define precisamente uma interface SQL para a definição de tabelas, para as operações sobre as mesmas (seleção, projeção, junção e outras) e para a definição de regras de integridade de bancos de dados. A interface SQL é implementada em todos os sistemas de bancos de dados relacionais existentes (Date, 1998).

Uma das partes do SQL é o DDL (Linguagem de Definição de Dados), que fornece comandos para definições de esquemas de relação, criação ou remoção de tabelas, criação de índices e modificação de esquemas.

O SGBD (Sistema de Gestão de Banco de Dados) utilizado foi o SGBD Interbase 6.0 da Borland que é software livre e o sistema operacional utilizado foi o Linux Mandrake 8.1.

Para se realizar a normalização do sistema de dessalinização foi necessário acessar as tabelas dessalinizador, variáveis e limites de operação do banco de dados. Para a criação dessas tabelas, temos os DDL a seguir:

- Dessalinizador:

```
CREATE TABLE dessalinizador(
        idDessalinizador INT IDENTITY(1,1),
        idPoco INT,
        orgao VARCHAR(50),
        localidade VARCHAR(50),
        solicitante VARCHAR(50),
        responsavel VARCHAR(50),
        operador VARCHAR(50),
        endereco VARCHAR(100),
        cep VARCHAR(10),
        fones VARCHAR(50),
        fax VARCHAR(50),
        celular VARCHAR(30),
        numHab VARCHAR(10),
        numHabBeneficiados VARCHAR(10),
        empresa VARCHAR(20),
        capacidade VARCHAR(10),
        recuperacao VARCHAR(10),
        tipoMembrana VARCHAR(20),
        areaMembrana FLOAT,
        fabricante VARCHAR(20),
        numMembranas VARCHAR(10),
        numVasosPressao VARCHAR(10),
        numFiltros VARCHAR(10),
        mesh VARCHAR(10),
        alimentacao VARCHAR(10),
        permeado VARCHAR(10),
        concentrado VARCHAR(10),
```

```
    baPotencia VARCHAR(10),
    baMarca VARCHAR(20),
    baFase VARCHAR(10), /*Monofásica ou Trifásica*/
    b1Potencia VARCHAR(10),
    b1NumEstagios VARCHAR(10),
    b1Marca VARCHAR(20),
    b1Fase VARCHAR(10), /*Monofásica ou Trifásica*/
    b2Potencia VARCHAR(10),
    b2NumEstagios VARCHAR(10),
    b2Marca VARCHAR(20),
    b2Fase VARCHAR(10), /*Monofásica ou Trifásica*/

    /*Pré-Tratamento*/
    pFiltro VARCHAR(3), /*Sim ou Não*/
    pFiltroMulti VARCHAR(3), /*Sim ou Não*/
    pAcido VARCHAR(3), /*Sim ou Não*/
    pBasico VARCHAR(3), /*Sim ou Não*/
    acido VARCHAR(10), /*H2SO4 ou HCL*/
    base VARCHAR(10), /*NaCH ou Outros*/

    PRIMARY KEY (idDessalinizador),
    FOREIGN KEY(idPoco) REFERENCES poco(idPoco)
        );
Create generator genCodDessalinizador;
        SET TERM !! ;
        CREATE TRIGGER createIdDessalinizador FOR dessalinizador
        ACTIVE BEFORE INSERT POSITION 0
        AS BEGIN
        NEW.idDessalinizador = GEN_ID(genCodDessalinizador, 1);
        END !!
SET TERM ; !!
```

A tabela "dessalinizador" possui uma grande coleção de dados. Alguns dados são referentes à comunidade como, por exemplo: órgão responsável, nome da localidade, nome do operador, endereço, número de habitantes beneficiados, etc; e outros são referentes ao próprio equipamento como, por exemplo: número de membranas e filtros, área da membrana, potência, marca e outros.

- Variáveis:

```
CREATE TABLE variaveis(
    idVariaveis INT IDENTITY(1,1),
    idDessalinizador INT NOT NULL,
    data DATETIME NOT NULL ,
    p1menosp2Ultimo FLOAT,
```

```
    p3menosp4Ultimo FLOAT,
    p1Ultimo FLOAT,
    p2Ultimo FLOAT,
    p3Ultimo FLOAT,
    p4Ultimo FLOAT,
    v1Ultimo FLOAT,
    v2Ultimo FLOAT,
    temperatura FLOAT,
    ph FLOAT,
    tdsA FLOAT,

    alarmeRemoto VARCHAR(3),
    alarmeCentral VARCHAR(3),
    checado VARCHAR(3),
    PRIMARY KEY (idVariaveis),

    FOREIGN KEY(idDessalinizador)
    REFERENCES
    dessalinizador(idDessalinizador)
);

Create generator genCodVariaveis;
    SET TERM !! ;
    CREATE TRIGGER createIdVariaveis FOR variaveis
        ACTIVE BEFORE INSERT POSITION 0
        AS BEGIN
            NEW.idVariaveis = GEN_ID(genCodVariaveis, 1);
        END !!
    SET TERM ; !!
```

A tabela "variáveis" armazena os valores último de cada variável em relação a uma determinada data. Esses valores são enviados pelo computador remoto.

- Limites de operação:

```
CREATE TABLE limites(
    idLimites int NOT NULL,
    idDessalinizador int NOT NULL,
    data date,

    p3Simulado VARCHAR(10),
    p4Simulado VARCHAR(10),
    q1Simulado VARCHAR(10),
    q2Simulado VARCHAR(10),
    tSimulado VARCHAR(10),
    phSimulado VARCHAR(10),
    tdsASimulado VARCHAR(10),
```

```
p1Operacao FLOAT,
p2Operacao FLOAT,
p3Operacao FLOAT,
p4Operacao FLOAT,
p5Operacao FLOAT,
q1Operacao FLOAT,
q2Operacao FLOAT,
tOperacao FLOAT,
phOperacao FLOAT,
tdsAOperacao FLOAT,

p1Menosp2 FLOAT,
p3Menosp4 FLOAT,

        PRIMARY KEY (idLimites),
        FOREIGN KEY(idDessalinizador) REFERENCES
        dessalinizador(idDessalinizador) ON DELETE CASCADE );

Create generator genCodLimites;
        SET TERM !! ;
        CREATE TRIGGER createIdLimites FOR limites
                ACTIVE BEFORE INSERT POSITION 0
                AS BEGIN
                        NEW.idLimites = GEN_ID(genCodLimites, 1);
                END !!
        SET TERM ; !!
```

A tabela de "limites de operação" armazena os valores simulados (referência ou de projeto) das variáveis de medidas cadastrados pelo usuário. Esta tabela também é responsável por armazenar os valores do limite de operação do dessalinizador ao longo de seu funcionamento. As medidas de operação são obtidas quando o dessalinizador se estabiliza logo após a sua partida inicial ou após a realização de uma limpeza química do mesmo.

Além de realizar a normalização do sistema de dessalinização, o módulo de gerência é capaz de calcular o Índice de Saturação de Langelier (ISL) através das Equações 22 a 27. Este fator mede o risco de formação de precipitados de carbonato de cálcio em águas salobras com STD até 10.000 mg/L. Para isso, é necessário acessar no banco de dados os valores do STD (sólidos totais dissolvidos), da concentração do cálcio como $CaCO_3$, da temperatura, da alcalinidade como $CaCO_3$ e do pH da solução de alimentação. Estes valores são cadastrados pelo usuário no banco de dados através da tabela Análise físico-química, cujo DDL é o seguinte:

```
CREATE TABLE analise(
        idAnalise INT NOT NULL,
```

```
        idPoco INT,
        laudo VARCHAR(10),
        orgao VARCHAR(20),
        localidade VARCHAR(20),
        procedencia VARCHAR(20),
        dataColeta DATE,
        respColeta VARCHAR(20),
        dataEntregaAmostra DATE,
        recipiente VARCHAR(20),
        dataAnalise DATE,
        condutividade VARCHAR(10),
        potencial VARCHAR(10),
        turbidez VARCHAR(10),
        cor VARCHAR(20),
        odor VARCHAR(20),
        sabor VARCHAR(20),
        durezaCalcio VARCHAR(10),
        durezaMagnesio VARCHAR(10),
        durezaTotal VARCHAR(10),
        sodio VARCHAR(10),
        potassio VARCHAR(10),
        estroncio VARCHAR(10),
        bario VARCHAR(10),
        ferro VARCHAR(10),
        manganes VARCHAR(10),
        alcalinidadeHidroxidos VARCHAR(10),
        alcalinidadeCarbonatos VARCHAR(10),
        alcalinidadeBicarbonatos VARCHAR(10),
        alcalinidadeTotal VARCHAR(10),
        sulfato VARCHAR(10),
        cloreto VARCHAR(10),
        nitrato VARCHAR(10),
        nitrito VARCHAR(10),
        silica  VARCHAR(10),
        totalSolidos VARCHAR(10),
        realizadorAnalise VARCHAR(50),

        PRIMARY KEY (idAnalise),
        FOREIGN KEY(idPoco) REFERENCES    poco(idPoco)
);

Create generator genCodAnalise;
SET TERM !! ;
CREATE TRIGGER createIdAnalise FOR analise
        ACTIVE BEFORE INSERT POSITION 0
        AS BEGIN
                NEW.idAnalise = GEN_ID(genCodAnalise, 1);
        END !!
SET TERM ; !!
```

Nesta tabela são armazenadas informações sobre as análises físico-químicas e organolépticas da água do poço como, por exemplo, a condutividade elétrica, potencial hidrogeniônico, turbidez, cor, odor, sabor e outros. Todos as medidas precisam estar dentro do máximo permissível ou recomendável pela Legislação Brasileira (PORTARIA 36/90 MS).

2.3.2. O bean Normalizacao.java

A classe Normalizacao.java é responsável por realizar os cálculos dos parâmetros normalizados a partir dos dados encapsulados nos beans variaveis, dessalinizador e limites. Para isso, foram inseridas as equações encarregadas de realizar a normalização de um sistema de osmose inversa apresentadas no Capítulo I (Equações 3, 10 e 13 a 21).

Na documentação dessa classe mostrada no Anexo I, pode-se observar os dois contrutores e os métodos da mesma, bem como as suas variáveis de instância declaradas como *private* (privado) para garantir que estas não sejam visíveis para as outras classes, pois uma subclasse não pode acessar os membros de dados privados de sua superclasse. Ou seja, a palavra-chave *private* garante que ninguém nem nada poderá acessar os campos de instância sem ser pelos métodos da classe.Logo, a declaração das variáveis de instância foi feita no inicio da classe da seguinte maneira:

```
...
public class Normalizacao implements Serializable{
// campos
private Variaveis variaveis;
private Limites limites;
private Dessalinizador dessalinizador;
private static final float TREF = 25; // Temperatura de referência
...
```

Além das variáveis de instância, são declarados como *public*, no inicio da classe, algumas constantes do tipo String que contém um diagnóstico para cada tipo de incrustação. Estas constantes são retornadas no método da classe Normalizacao.java que trata do diagnóstico do sistema de dessalinização:

```
...
// constantes de mensagens de diagnosticos
public static final String MENSAGEM_OK = "<br>\nNo momento não está havendo potencial de incrustação.";

public static final String SCALING = "Diagnóstico:<br>\nPotencial de incrustação ocasionada por scaling localizada nos últimos elementos de membranas.<br><br>\nFerramentas que podem ser usadas para confirmação do" +
```

" diagnóstico:
\n- Usar um monitor ScaleGuard;
\n - Verificar o ISL do rejeito;
\n- Calcular a solubilidade máxima do sulfato de cálcio, sulfato de bário" +
" e sílica na análise do rejeito.

\nAções corretivas:
\n-Aumentar a adição de ácido e anti-incrustante para o carbonato de cálcio e o sulfato de cálcio;" +
"
\n- Reduzir a recuperação;
\n- Limpeza com ácido cítrico ou solução baseada em EDTA para carbonato de cálcio, sulfato de cálcio e sulfato de bário;" +
"
\n- Limpeza de incrustantes a base de silicatos com soluções a base de fluoreto de amônia.

\n";

public static final String OXIDOS_METALICOS = "Diagnóstico:
\nPotencial de incrustação ocasionada por óxidos metálicos localizada nos primeiros elementos de membranas.

\nFerramenta que pode ser usada" +
" para confirmação do diagnóstico:
\n- Análise de íons metálicos na solução de limpeza.

\nAção corretiva:
\n- Otimizar o pré-tratamento para remover metais.

\n";

public static final String COLOIDES_SILICA = "Diagnóstico:
\nPotencial de incrustação ocasionada por colóides/sílica localizada na entrada e saída do equipamento.

\nFerramentas que podem ser usadas para" +
" confirmação do diagnóstico:
\n- Calcular o SDI e/ou MFI-UF da alimentação;
\n- Difração de raios-X;
\n- Análise da solução de limpeza e reciclo." +
"

\nAções corretivas:
\n- Otimizar o sistema de pré-tratamento para remoção de colóides, incluindo a filtração;
\n- Estabilização da carga;" +
"
\n- Limpeza com EDTA, STP ou detergentes do tipo BIZ com pH elevado;
\n- Baixar a recuperação do sistema;
\n- Limpeza de incrustantes a base de" +
" silicatos com soluções a base de fluoreto de amônia.

\n";

public static final String ORGANICOS = "Diagnóstico:
\nPotencial de incrustação ocasionada por orgânicos localizada nos primeiros elementos de membranas.

\nFerramenta que pode ser usada para confirmação" +
" do diagnóstico:
\n- Teste destrutivo, isto é, análises de reflexão do IR.

\nAções corretivas:
\n- Otimização do sistema de pré-tratamento com processos" +
" de coagulação e filtração;
\n- Tratamento com carvão ativado ou resina;
\n- Limpeza com detergente a pH elevado ou com solução de isopropanol.

\n";

public static final String BIOFOULING = "Diagnóstico:
\nPotencial de incrustação ocasionada por biofouling localizada nos primeiros elementos de membranas.

\nFerramentas que podem ser usadas para" +
" confirmação do diagnóstico:
\n- Monitor de biofilme, AOC e SOCR;
\n- Contagem de bactérias no permeado e no rejeito;
\n- Verificação da formação" +
" de lodo nos tubos e vasos.

\nAções corretivas:
\n- Adição de bissulfeto de sódio;
\n- Cloração com ou sem filtração por carvão ativado;" +
"
\n- Limpeza com soluções a base de EDTA ou detergentes do tipo BIZ a pH elevado;
\n- Desenvolver programas de desinfecção com formaldeído," +
" peróxido de hidrogênio, etc.

\n";
...

Após a declaração das constantes são declarados os valores normalizados, que são os parâmetros que se quer obter com a classe Normalizacao.java calculados nos vários métodos da mesma.. Estes valores também são declarados como *private*:

```
...
// Valores normalizados
private float diferencialPressaoN; // Diferencial de pressão normalizada
private float diferencialPressaoNS; // Diferencial de pressão normalizada simulada
private float vazaoN; // Vazão normalizada
private float passagemSalN; // Passagem de sal normalizada
private float concentracaoPermeadoN; // Concentração do permeado normalizada
private float transferenciaMassaN; // Transferência de massa normalizada
...
```

Como se pode observar na documentação da classe Normalizacao.java mostrada no Anexo I, esta apresenta dois construtores, sendo o primeiro padrão e o segundo contendo como parâmetros objetos das classes Variaveis.java, Limites.java e Dessalinizador.java. O construtor padrão é um construtor sem parâmetros que inicializa todas as variáveis de instância com seus valores padrão. Assim, todas as variáveis objeto das classes citadas anteriomente apontam para nulo. O segundo construtor é usado para inicializar os objetos das classes Variaveis.java, Limites.java e Dessalinizador.java, dando às variáveis de instância o seu estado inicial.

Após os construtores, temos os vários métodos que são responsáveis de realizar as ações da classe Normalizacao.java. Como se pode observar na documentação no Anexo I, a maioria dos métodos são *get* (acessadores) que têm a função de consultar o estado do objeto e o informar. Estes métodos geralmente retornam valores do tipo *float* e são os que contêm as equações e realizam os cálculos propriamente dito. Nos ítens 2.3.2.2 a 2.3.2.5 deste capítulo são detalhadas as etapas usadas por estes métodos para calcular os parâmetros normalizados a partir das variáveis de medidas. Há também os métodos *get* que retornam um objeto das classes Limites, Dessalinizador e Variaveis. Além dos métodos *get*, existem os métodos *set* (modificadores) que servem para modificar o estado dos objetos, ou seja, alteram os campos de instância. Estes métodos não têm retorno e isto é indicado pela palavra-chave *void* na sua assinatura. A classe Normalizacao.java contém um método *formataValor* que recebe um valor do tipo *float* como parâmetro e retorna uma *String*. Este método é encarregado de formatar o número visualizado pelo usuário no browser, transformando um valor do tipo *float* em uma *String* que representa um "número" com três casas decimais separadas do inteiro por vírgula como, por exemplo, 2,315. Finalmente, esta classe contém um método *equals* que retorna um *boolean* e tem a função de verificar se dois objetos da classe são iguais. Todos os métodos são *public* e, por isso, podem ser chamados a partir de outras classes, ou seja, estão visíveis para os usuários da classe Normalizacao.java.

2.3.2.1.A equação da concentração do permeado

Além das equações mostradas no Capítulo I (Equações 3, 10 e 13 a 21) foi introduzido no bean Normalizacao.java um conjunto de equações que obtêm a concentração do permeado a partir da concentração da alimentação e da recuperação do sistema. Isto foi necessário devido ao fato de que o computador embarcado não apresenta no momento memória suficiente para coletar e enviar mais do que oito variáveis. Logo, não foi possível incluir a condutividade elétrica do permeado como uma varável a ser enviada remotamente.

Para se obter estas equações foram realizadas simulações realizadas nos programas ROSA (Reverse Osmosis System Analysis) versão 4.30 for Windows fornecido pela Dow Química, fabricante das membranas Filmtec BW30-4040, e o ROPRO 7.0 fornecido pelo fabricante de membranas Koch Membrane Systems Inc.. Conforme mostram as Tabelas 6 e 7, para cada recuperação do sistema, utilizou-se águas de poços tubulares com concentrações diferentes (600 a 8500 mg/L), obtendo-se após a simulação a concentração do permeado correspondente. As análises fisico-químicas dessas águas encontram-se no Anexo II.

Tabela 6: Concentração do permeado (C_P) em função da concentração da alimentação (Ca) e da recuperação do sistema (r) utilizando o simulador ROSA 4.30.

Concentração da alimentação (mg/L)	**Recuperação do sistema (r)**				
	0,25	**0,35**	**0,45**	**0,55**	**0,65**
611,56	5,0	6,0	7,0	7,0	9,0
1052,4	11,0	12,0	15,0	15,0	18,0
2281,5	29,0	31,0	38,0	35,0	41,0
3500,0	41,0	44,0	53,0	50,0	59,0
5283,6	54,0	59,0	71,0	67,0	79,0
7472,6	85,0	93,0	112,0	106,0	125,0
8491,8	95,0	104,0	125,0	118,0	139,0

Os sistemas de dessalinização utilizados nas simulações dependeram da recuperação e apresentaram as seguintes características:

- Para $0,20 \leq r < 0,40$: Sistema constituído por três vasos de alta pressão em série com um elemento de membrana do tipo BW30-4040 (ROSA 4.30) ou TFC HR-4040 (ROPRO 7.0) cada. A vazão do permeado obtida com esse sistema foi fixada em 0,6 m^3/h.

Tabela 7: Concentração do permeado (C_P) em função da concentração da alimentação (Ca) e da recuperação do sistema (r) utilizando o simulador ROPRO 7.0.

Concentração da alimentação (mg/L)	Recuperação do sistema (r)				
	0,25	**0,35**	**0,45**	**0,55**	**0,65**
611,56	22,4	18,4	18,2	13,4	13,6
1052,4	40,2	33,1	33,0	24,4	25,0
2281,5	96,3	80,4	81,5	61,7	65,5
3500,0	155,8	129,8	133,3	100,8	110,6
5283,6	229,2	193,8	203,0	157,2	178,8
7472,6	423,2	360,0	382,3	299,8	350,7
8491,8	487,1	415,5	443,6	349,3	411,3

- Para $0,40 \leq r < 0,50$: Sistema constituído por um vaso de alta pressão com três elementos de membranas do tipo BW30-4040 (ROSA 4.30) ou TFC HR-4040 (ROPRO 7.0) em série com outro vaso também com três elementos, totalizando 6 elementos de membranas. A vazão do permeado obtida com esse sistema foi fixada em 1,0 m^3/h.
- Para $0,50 \leq r < 0,60$: Sistema constituído por dois vasos de alta pressão em série com três elementos de membranas do tipo BW30-4040 (ROSA 4.30) ou TFC HR-4040 (ROPRO 7.0) cada e em paralelo com um terceiro vaso também com três elementos, totalizando 9 elementos de membranas. A vazão do permeado obtida com esse sistema foi fixada em 2,0 m^3/h.
- Para $0,60 \leq r < 0,70$: Sistema constituído por um vaso de alta pressão com seis elementos de membranas do tipo BW30-4040 (ROSA 4.30) ou TFC HR-4040 (ROPRO 7.0) em série com outro vaso também com seis elementos, totalizando 12 elementos de membranas. A vazão do permeado obtida com esse sistema foi fixada em 2,5 m^3/h.

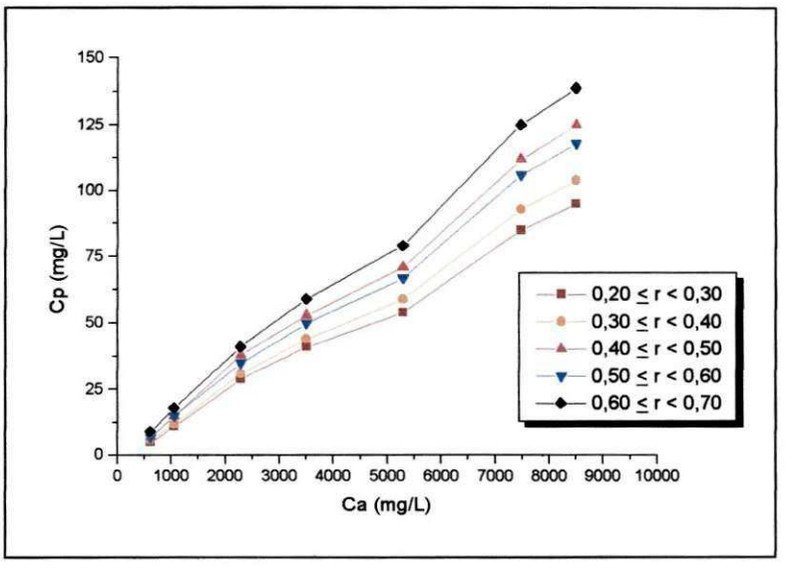
Com os dados das Tabelas 6 e 7, plotou-se os gráficos de C_P versus Ca, obtendo-se as curvas mostradas na Figura 2.4.

A partir dessas curvas, encontrou-se o conjunto de equações que obtêm a concentração do permeado a partir da concentração da alimentação e da recuperação do sistema para cada simulador utilizado:

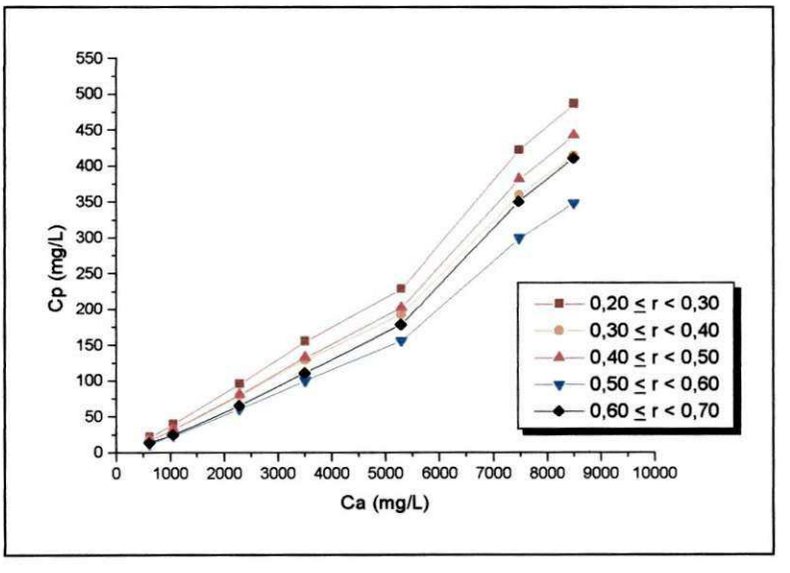
Simulador ROSA 4.30

- Para $0,20 \leq r < 0,30$ (Fator de regressão R = 0,993):

$$C_P = 0{,}0112\ Ca - 0{,}14 \qquad (33)$$



(a)



(b)

Figura 2.4: Concentração do permeado (C_P) em função da concentração da alimentação (Ca) e da recuperação do sistema (r) utilizando os simuladores (a) ROSA 4.30 e (b) ROPRO 7.0.

- Para $0{,}30 \leq r < 0{,}40$ (Fator de regressão R = 0,994):

$$C_P = 0{,}0122\ Ca - 0{,}34 \qquad (34)$$

- Para $0{,}40 \leq r < 0{,}50$ (Fator de regressão R = 0,994):

$$C_P = 0{,}0147\ Ca - 0{,}07 \qquad (35)$$

- Para $0{,}50 \leq r < 0{,}60$ (Fator de regressão R = 0,994):

$$C_P = 0{,}115 + 0{,}0138\ Ca \quad (36)$$

- Para $0{,}60 \leq r < 0{,}70$ (Fator de regressão R = 0,995):

$$C_P = 0{,}5 + 0{,}0163\ Ca \quad (37)$$

Simulador ROPRO 7.0

- Para $0{,}20 \leq r < 0{,}30$ (Fator de regressão R = 0,995):

$$C_P = 8{,}82 + 0{,}027\ Ca + 3{,}53x10^{-6}\ Ca^2 \quad (38)$$

- Para $0{,}30 \leq r < 0{,}40$ (Fator de regressão R = 0,996):

$$C_P = 7{,}37 + 0{,}022\ Ca + 3{,}15x10^{-6}\ Ca^2 \quad (39)$$

- Para $0{,}40 \leq r < 0{,}50$ (Fator de regressão R = 0,996):

$$C_P = 7{,}46 + 0{,}021\ Ca + 3{,}62x10^{-6}\ Ca^2 \quad (40)$$

- Para $0{,}50 \leq r < 0{,}60$ (Fator de regressão R = 0,996):

$$C_P = 5{,}78 + 0{,}015\ Ca + 3{,}05x10^{-6}\ Ca^2 \quad (41)$$

- Para $0{,}60 \leq r < 0{,}70$ (Fator de regressão R = 0,996):

$$C_P = 6{,}12 + 0{,}014\ Ca + 4{,}01x10^{-6}Ca^2 \quad (42)$$

Para se determinar o conjunto de equações que melhor representa o cálculo do concentrado do permeado a partir da concentração da alimentação e da recuperação do sistema, comparou-se a concentração do permeado do Sistema de Dessalinização Piloto (SDP) utilizado no projeto com os resultados obtidos utilizando as equações encontradas com os simuladores ROSA 4.30 e ROPRO 7.0. Para isso, o SDP foi alimentado com a água de um poço localizado na cidade de Riacho de Santo Antônio – PB cujas concentrações de sais estão indicadas na Tabela 8.

Tabela 8: Comparação da concentração do permeado do SDP com os resultados obtidos utilizando as equações encontradas com os simuladores ROSA 4.30 e ROPRO 7.0.

P3 (kgf/cm^2)	Ca (mg/L)	r	Cp SDP (mg/L)	Cp Rosa (mg/L)	Cp Ropro (mg/L)
14	9705	0,211	334,0	108,6	603,3
16	8996	0,277	289,0	100,6	537,4
18	8729	0,344	169,0	106,2	439,4
20	8540	0,455	276,0	125,5	450,8

Esta tabela mostra também a recuperação do sistema em função da concentração da alimentação e da pressão aplicada na entrada da membrana. Logo, de acordo com as recuperações obtidas com o SDP, utilizou-se as Equações 33, 34 e 35 encontradas com o

simulador ROSA 4.30 e as Equações 38, 39 e 40 obtidas com o simulador ROPRO 7.0 para se calcular a concentração do permeado em cada caso. A Figura 2.5 apresenta as curvas da concentração do permeado para os três casos apresentados na Tabela 8.

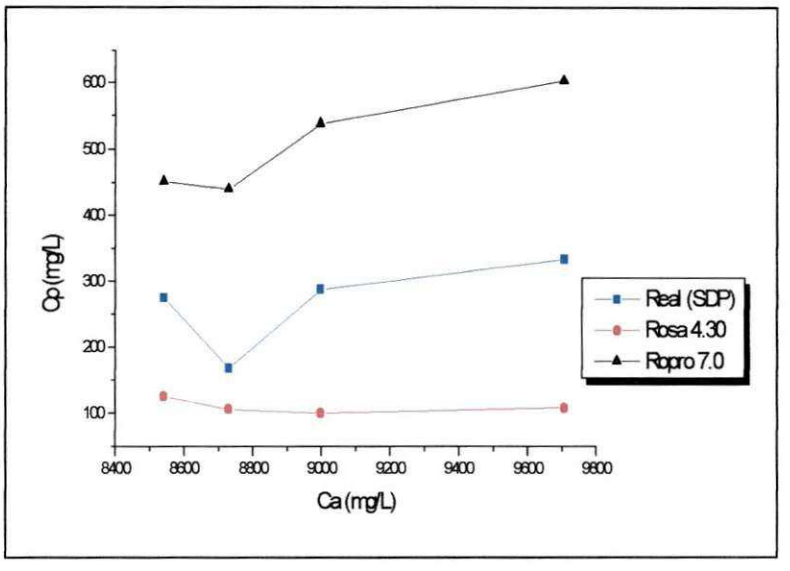


Figura 2.5: Comparação da concentração do permeado do SDP com os resultados obtidos utilizando as equações encontradas com os simuladores ROSA 4.30 e ROPRO 7.0.

Analisando-se esta figura, observa-se que os valores da concentração do permeado obtidos com as equações originadas pelas simulações com o ROSA 4.30 e com o ROPRO 7.0 estão distantes dos valores reais obtidos do SDP. No entanto, pode-se observar também que os valores reais representam aproximadamente a média aritmética dos valores obtidos com os modelos dos simuladores. Com isso, fez-se a média aritmética dos termos das equações obtidas com o ROSA 4.30 e com o ROPRO 7.0 para cada faixa de recuperação, encontrando os seguintes modelos:

- Para $0{,}20 \leq r < 0{,}30$: $C_P = 4{,}34 + 0{,}0191\ Ca + 1{,}765 \times 10^{-6}\ Ca^2$ (43)
- Para $0{,}30 \leq r < 0{,}40$: $C_P = 3{,}51 + 0{,}0171\ Ca + 1{,}575 \times 10^{-6}\ Ca^2$ (44)
- Para $0{,}40 \leq r < 0{,}50$: $C_P = 3{,}69 + 0{,}0178\ Ca + 1{,}81 \times 10^{-6}\ Ca^2$ (45)
- Para $0{,}50 \leq r < 0{,}60$: $C_P = 2{,}95 + 0{,}0144\ Ca + 1{,}525 \times 10^{-6}\ Ca^2$ (46)
- Para $0{,}60 \leq r < 0{,}70$: $C_P = 3{,}31 + 0{,}0151\ Ca + 2{,}005 \times 10^{-6} Ca^2$ (47)

A Tabela 9 apresenta os resultados obtidos a partir das Equações 43, 44 e 45, comparando-os com a concentração do permeado do Sistema de Dessalinização Piloto (SDP) utilizado no projeto e com os resultados obtidos utilizando as equações encontradas com os

simuladores ROSA 4.30 e ROPRO 7.0. A Figura 2.6 apresenta as curvas da concentração do permeado para os quatro casos apresentados nesta tabela.

Tabela 9: Comparação da concentração do permeado do SDP com os resultados obtidos utilizando as equações do modelo médio e as dos simuladores ROSA 4.30 e ROPRO 7.0.

P3 (kgf/cm^2)	Ca (mg/L)	r	Cp SDP (mg/L)	Cp Rosa (mg/L)	Cp Ropro (mg/L)	Cp Médio (mg/L)
14	9705	0,211	334,0	108,6	603,3	355,9
16	8996	0,277	289,0	100,6	537,4	319,0
18	8729	0,344	169,0	106,2	439,4	272,8
20	8540	0,455	276,0	125,5	450,8	287,7

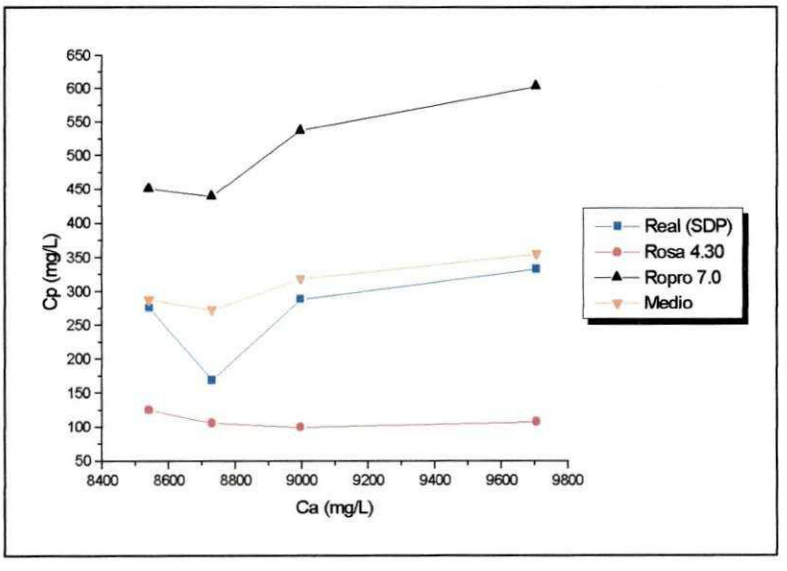


Figura 2.6: Comparação da concentração do permeado do SDP com os resultados obtidos utilizando as equações do modelo médio e as dos simuladores ROSA 4.30 e ROPRO 7.0.

Analisando-se esta figura, observa-se que os valores da concentração do permeado obtidos com o modelo médio (Equações 43, 44 e 45) estão mais próximos dos valores reais obtidos do SDP. Isto significa que o erro ao se utilizar as equações obtidas com o modelo médio é menor do que quando se usa as equações obtidas com os simuladores ROSA 4.30 e ROPRO 7.0. Com isso, implementou-se as Equações 43 a 47 na classe Normalizacao.java para se calcular a concentração do permeado.

O método da classe Normalizacao.java que trata do cálculo da concentração do permeado(C_P) foi implementado da seguinte maneira:

```
...
public float getConcentracaoP(){
        float r = getRecuperacao(); // Recuperação do sistema
        float tdsA = variaveis.getTdsA(); // Total de sólidos da alimentação
        float retorno = 100;
        if (r >= 0.20 && r < 0.30){
                retorno = (float) (4.34 + 0.0191*tdsA + 0.000001765*tdsA*tdsA);
        } else if (r >= 0.30 && r < 0.40){
                retorno = (float) (3.51 + 0.0171*tdsA + 0.000001575*tdsA*tdsA);
        } else if (r >= 0.40 && r < 0.50){
                retorno = (float) (3.69 + 0.0178*tdsA + 0.00000181*tdsA*tdsA);
        } else if (r >= 0.50 && r < 0.60){
                retorno = (float) (2.95 + 0.0144*tdsA + 0.000001525*tdsA*tdsA);
        } else if (r >= 0.60 && r < 0.70){
                retorno = (float) (3.31 + 0.0151*tdsA + 0.000002005*tdsA*tdsA);
        }
        return retorno;
}
...
```

Como se pode observar no método acima, se a recuperação do sistema não estiver em nenhuma das faixas estudadas, o valor da concentração do permeado retornado é 100 mg/L.

A concentração do concentrado foi obtida a partir de um balanço de massa (Equação 10), utilizando a concentração da alimentação (Ca), a recuperação do sistema (r) e a concentração do permeado (C_P) calculada da maneira descrita anteriormente.

2.3.2.2. Cálculo da vazão do permeado normalizada (Q_n)

A Figura 2.7 apresenta um diagrama que representa o algoritmo adotado para se determinar a vazão do permeado normalizada (Q_n) na classe Normalizacao.java. As variáveis que se encontram representadas no diagrama dentro de círculos são as cadastradas no banco de dados e que alimentam as equações durante os cálculos. As variáveis em tempo de operação são obtidas através da classe Variaveis.java usando os métodos *get* dessa classe, ao passo que as simuladas ou de referência são obtidas através da classe Limites.java. Cada etapa descrita nesse algorítmo corresponde a um método *get* da classe Normalizacao.java que retorna o valor do parâmetro desejado. Logo, de acordo com a Figura 2.7, adotou-se o seguinte algoritmo:

1. Cálculo da razão de recuperação do sistema (r) através da Equação 3. Variáveis de entrada: vazão do permeado (Q_p ou v2Ultimo) e a vazão do concentrado (Q_c ou v1Ultimo);

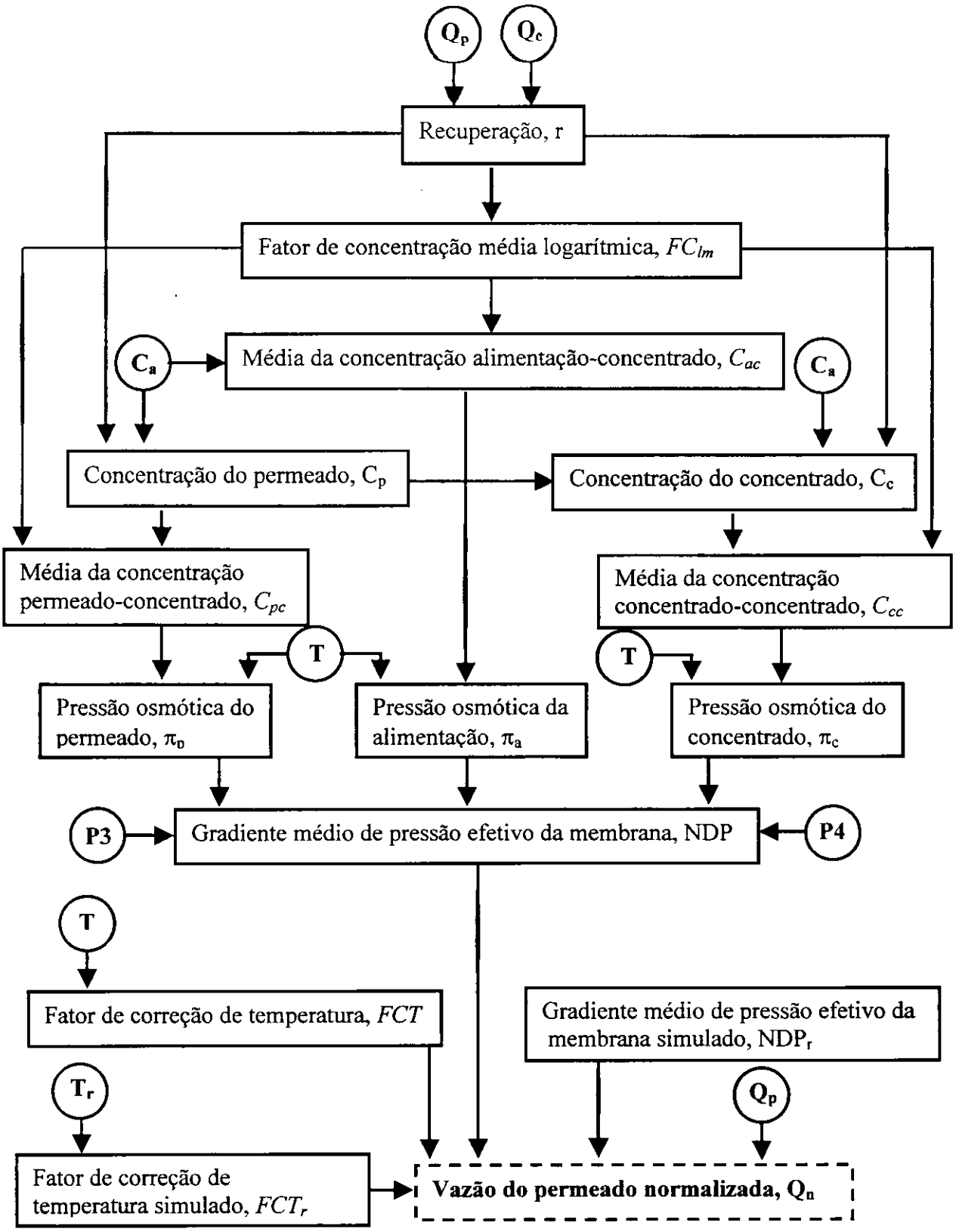


Figura 2.7: Diagrama que representa o algoritmo adotado para se determinar a vazão do permeado normalizada (Q_n).

2. Obtenção do fator de concentração média logarítmica (FC_{lm}) utilizando a Equação 17 e tendo como parâmetro de entrada a razão de recuperação do sistema (r) calculada anteriormente;
3. Cálculo da média da concentração alimentação-concentrado (C_{ac}) tendo como variável de entrada a concentração da alimentação (Ca ou tdsA) e como parâmetro de entrada o fator de concentração média logarítmica (FC_{lm});
4. Cálculo da concentração do permeado (C_p) através das Equações 43 a 47 tendo como variável de entrada a concentração da alimentação (Ca ou tdsA) e como parâmetro de entrada a razão de recuperação do sistema (r);
5. Determinação da concentração do concentrado (C_c) através da Equação 10 tendo como variável de entrada a concentração da alimentação (Ca ou tdsA) e como parâmetros de entrada a razão de recuperação do sistema (r) e a concentração do permeado (C_p);
6. Cálculo da média da concentração permeado-concentrado (C_{pc}) tendo como parâmetros de entrada a concentração do permeado (C_p) e o fator de concentração média logarítmica (FC_{lm});
7. Cálculo da média da concentração concentrado-concentrado (C_{cc}) tendo como parâmetros de entrada a concentração do concentrado (C_c) e o fator de concentração média logarítmica (FC_{lm});
8. Determinação da pressão osmótica da alimentação (π_a) através da Equação 15 tendo como variável de entrada a temperatura (T ou t) e como parâmetro de entrada a média da concentração alimentação-concentrado (C_{ac});
9. Determinação da pressão osmótica do permeado (π_p) através da Equação 15 tendo como variável de entrada a temperatura (T ou t) e como parâmetro de entrada a média da concentração permeado-concentrado (C_{pc});
10. Cálculo da pressão osmótica do concentrado (π_c) através da Equação 15 tendo como variável de entrada a temperatura (T ou t) e como parâmetro de entrada a média da concentração concentrado-concentrado (C_{cc});
11. Obtenção do gradiente médio de pressão efetivo da membrana (NDP) através da Equação 14 tendo como variáveis de entrada a pressão de entrada da membrana (P3 ou p3ultimo) e a pressão de saída da membrana (P4 ou p4ultimo) e como parâmetros de entrada as pressões osmóticas da alimentação, permeado e do concentrado. Nesse caso, a pressão do permeado (P_p) foi considerada constante e igual a 1,0 kgf/cm^2.

12. Cálculo do fator de correção de temperatura (*FCT*) através da Equação 18 tendo como variável de entrada a temperatura (T ou t);
13. Cálculo do fator de correção de temperatura simulado (FCT_r) através da Equação 18 tendo como variável de entrada a temperatura simulada ou de referência (T_r ou tSimulado);
14. Cálculo do gradiente médio de pressão efetivo da membrana simulado (NDP_r) através da Equação 14 tendo como variáveis de entrada a pressão de entrada da membrana simulada ($P3_r$ ou p3Simulado) e a pressão de saída da membrana simulada ($P4_r$ ou p4Simulado) e como parâmetros de entrada as pressões osmóticas simuladas da alimentação, permeado e do concentrado. Nesse caso, a pressão do permeado simulada (P_{pr}) também foi considerada constante e igual a 1,0 kgf/cm^2. O procedimento adotado para se calcular o gradiente médio de pressão efetivo da membrana simulado (NDP_r) foi o mesmo usado no cálculo do NDP (etapas 1 a 11 descritas anteriormente), tendo como diferença que no caso do NDP_r, utilizou-se as variáveis de referência (simuladas) obtidas através de um objeto da classe Limites.java.
15. Finalmente, chega-se a vazão do permeado normalizada (Q_n) através da Equação 13 tendo como variável de entrada a vazão do permeado (Q_p ou v2Ultimo) e como parâmetros de entrada o gradiente médio de pressão efetivo da membrana simulado (NDP_r), o gradiente médio de pressão efetivo da membrana (NDP), o fator de correção de temperatura simulado (FCT_r) e o fator de correção de temperatura (*FCT*).

2.3.2.3. Cálculo dos sólidos totais dissolvidos do permeado normalizado (C_{Pn})

A Figura 2.8 apresenta um diagrama que representa o algorítimo adotado para se determinar os sólidos totais dissolvidos do permeado normalizado (C_{Pn}). Como se pode observar, este algorítmo é mais simples, pois todos os parâmetros usados nesse cálculo já foram determinados anteriormente durante o cálculo da vazão do permeado normalizada. Logo, o C_{Pn} é determinado em apenas um método da classe Normalizacao.java que invoca os parâmetros usados através dos métodos *get* que os determinam sem a necessidade de se criar objetos para chamá-los por se tratar de métodos da própria classe.

Nesse caso, para o cálculo do C_{Pn}, foi usada a Equação 19 tendo como parâmetros de entrada a concentração do permeado (C_p), o gradiente médio de pressão efetivo da membrana simulado (NDP_r), o gradiente médio de pressão efetivo da membrana (NDP), a pressão osmótica do permeado simulado (π_{pr}), a pressão osmótica do permeado (π_p), a média da

concentração alimentação-concentrado simulado (C_{acr}) e a média da concentração alimentação-concentrado (C_{ac}).

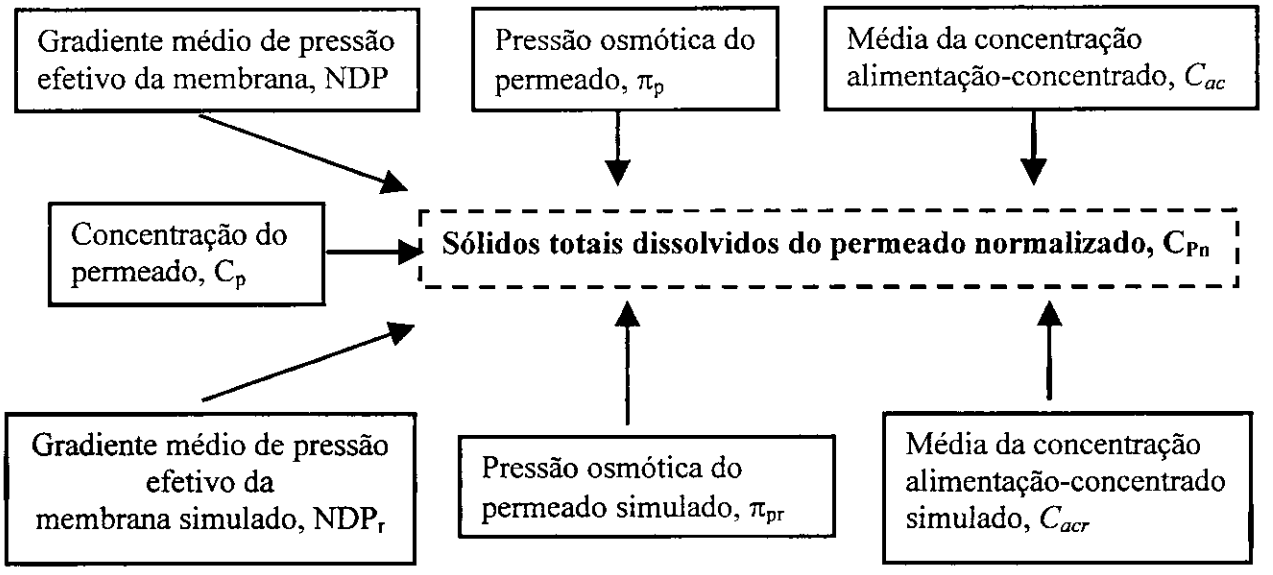


Figura 2.8: Diagrama que representa o algoritmo adotado para se determinar os sólidos totais dissolvidos do permeado normalizado, C_{Pn}.

2.3.2.4. Cálculo da passagem de sal normalizada (PS_n)

Como se pode observar na Figura 2.9, o algorítimo adotado para se determinar a passagem de sal normalizada (PS_n) também é simples, sendo necessário apenas um método *get* da classe Normalizacao.java.

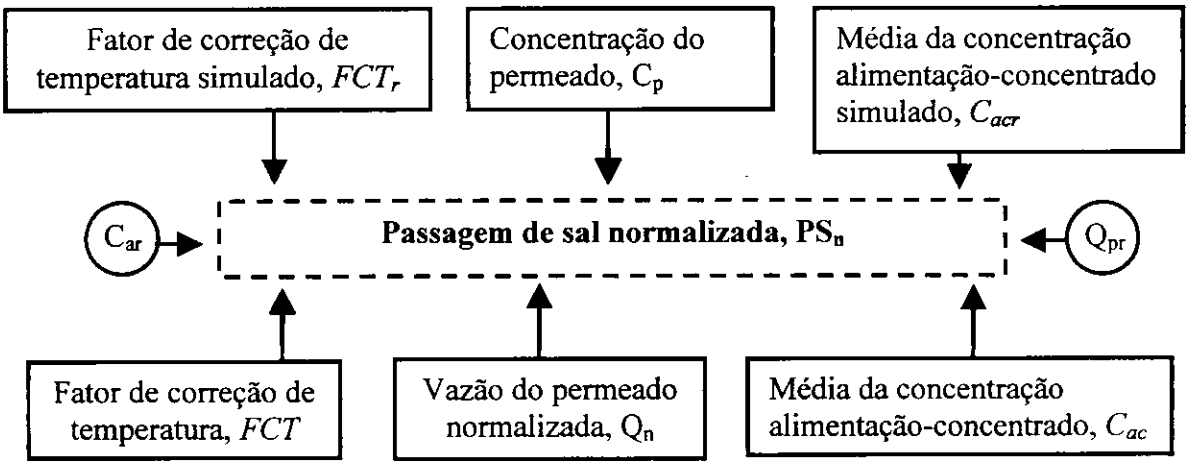


Figura 2.9: Diagrama que representa o algoritmo adotado para se determinar a passagem de sal normalizada (PS_n).

Para o cálculo do PS_n, foi usada a Equação 20 tendo como parâmetros de entrada, já calculados conforme descrito no ítem 2.3.3.2, a concentração do permeado (C_p), o fator de correção de temperatura simulado (FCT_r), o fator de correção de temperatura (FCT), a vazão do permeado normalizada (Q_n), a média da concentração alimentação-concentrado simulado (C_{acr}) e a média da concentração alimentação-concentrado (C_{ac}). As varáveis de entrada usadas nesse cálculo foram a concentração da alimentação simulada ou de referência (C_{ar} ou tdsASimulado) e a vazão do permeado simulada ou de referência (Q_{pr} ou qpSimulado).

2.3.2.5. Cálculo do coeficiente de transferência de massa (CTM)

Para se determinar o coeficiente de transferência de massa (CTM), cujo algorítimo está representado na Figura 2.10, utilizou-se a Equação 21, sendo necessário apenas um método *get* da classe Normalizacao.java.

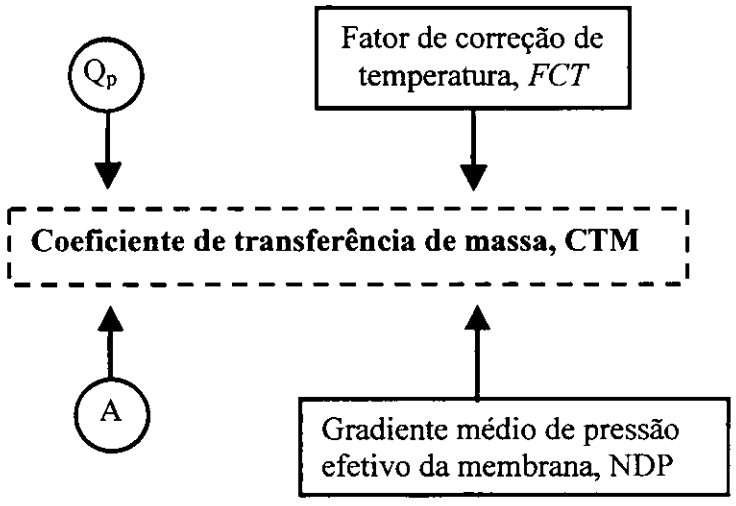


Figura 2.10: Diagrama que representa o algoritmo adotado para se determinar o coeficiente de transferência de massa (CTM).

Os parâmetros de entrada usados foram o fator de correção de temperatura (FCT) e o gradiente médio de pressão efetivo da membrana (NDP). As varáveis de entrada usadas nesse cálculo foram a área total das membranas (A ou areaMembrana) e a vazão do permeado (Q_p ou v2Ultimo). A área da membrana foi obtida através de um objeto da classe Dessalinizador.java usando um método *get* dessa classe, enquanto que a vazão do permeado é obtida através de um objeto da classe Variaveis.java.

2.3.2.6. O diagnóstico do sistema

O diagnóstico do sitema de dessalinização piloto foi realizado baseando-se nas variações dos dados normalizados (CTM, NDP e PSN), pois, segundo Huiting *et al.* (2001), os diferentes tipos de incrustações têm efeitos diferentes nestes parâmetros.

Como foi comentado no item 1.5.9, as incrustações ocorrem quando os parâmetros normalizados apresentarem as seguintes modificações em relação aos dados de partida da planta (Huiting *et al.*, 2001; Amjad, 1992):

1. O gradiente médio de pressão efetivo da membrana aumentar 10 -15%;
2. A passagem de sal normalizada (PS_n) ter um aumento significativo ao longo do tempo (em torno de 50%);
3. O coeficiente de transferência de massa (CTM) diminuir 10 -15%.

Logo, como se pode verificar no código abaixo, em valores absolutos, a variação mínima adotada foi de 10% para o NDP e o CTM, enquanto que a variação máxima adotada foi de 15% nos tipos de incrustações diagnosticados por uma mudança pequena a moderada nesses parâmetros normalizados (Tabela 4 do item 1.5.9). Nos casos em que essa mudança foi rápida / significante, adotou-se como limite superior qualquer valor maior do que 15%. No caso do PSN, adotou-se como limite qualquer valor maior ou igual a 50% quando o diagnóstico for representado por uma mudança rápida / significante (Scalling e óxidos metálicos). Para uma mudança pequena a moderada, adotou-se um intervalo entre 30 e 50%, como ocorre no caso de uma incrustação por colóides/sílica. No caso de incrustações por orgânicos e material biológico ("biofouling"), a passagem de sal normalizada não serve como parâmetro para se obter um diagnóstico.

```
...
/**
* Retorna o diagnóstico de incrustação, informando a sua localização no equipamento e o seu tipo.
*/
public String getDiagnostico() {

        String diagnosticoFinal = "";
        float varCtm = getVariacaoCTM();
        float varDpn = getVariacaoDPN();
        float varPsn = getVariacaoPSN();

        if ( (varCtm <= -10) && (varCtm > -15) && (varDpn >= 10) && (varDpn < 15) && (varPsn >= 50) ){
                diagnosticoFinal += SCALING;
```

```
            }
        if ( (varCtm <= -15) && (varDpn >= 15) && (varPsn >= 50) ){
            diagnosticoFinal += OXIDOS_METALICOS ;
            }
        if ( (varCtm <= -10) && (varCtm > -15) && (varDpn >= 10) && (varDpn < 15) &&
(varPsn > 30) && (varPsn <= 50) ){
            diagnosticoFinal += COLOIDES_SILICA;
            }
        if ( (varCtm <= -10) && (varCtm > -20) && (varDpn >= 10) && (varDpn < 20)){
            diagnosticoFinal += ORGANICOS;
            }
        if ( (varCtm < -15) && (varDpn > 10) ){
            diagnosticoFinal += BIOFOULING;
            }
        if (diagnosticoFinal.equals("")) {
            return (MENSAGEM_OK);
            }
        return diagnosticoFinal;
        }
...
```

Como se pode observar no código acima, para cada caso é retornada uma constante do tipo String que contém um diagnóstico para cada tipo de incrustação, informando a sua localização no equipamento, bem como as ações corretivas que devem ser tomadas para acabar ou diminuir a incrustação no sistema de mebranas, conforme descrito na Tabela 5 do item 1.5.9. Estas constantes foram declaradas no início da classe Normalizacao.java como *static final* e, por isso, podem ser usadas em qualquer lugar desta classe sem precisar de serem instanciadas.

2.3.3. A classe NormalizacaoDAO.java

Esta classe trata das tarefas referentes à normalização, sendo responsável por realizar uma consulta ao banco de dados e acessar os dados que "alimentam" as equações implementadas em Normalizacao.java e, a partir deste bean, gerar uma coleção de normalizações na faixa de tempo requerida que é, por sua vez, entregue a classe Sistema.java para que ela direcione uma resposta (as normalizações) a interface gráfica (JSP).

Para se consultar um banco de dados, é necessário antes se fazer uma conexão ao mesmo. Para isso, existe no sistema uma classe ConnectionDB.java que utiliza o kit Java Database Connectivity (JDBC), que é um pacote que permite aos programadores Java conectar-se a bancos de dados, fazer consultas ou atualizações usando a linguagem de

consulta padrão do mercado, o SQL. Logo, a classe NormalizacaoDAO.java usa um objeto do tipo ConnectionDB.java para se conectar ao banco de dados.

Em Java, uma conexão ao banco de dados pode ser feita através de um código como o apresentado abaixo:

```
...
/*Registrando um driver jdbc
  com.inet.tds.TdsDriver é o nome do driver para o SQLServer
  interbase.interclient.Driver driver JDBC para o interbase
*/

//Class.forName("com.inet.tds.TdsDriver").newInstance();
Class.forName("interbase.interclient.Driver").newInstance();
...
String jdbcURL = endereço do banco de dados;
String jdbcUsername = sa;
String jdbcPassword = "";
Connection con = DriverManager.getConnection(jdbcURL, jdbcUsername, jdbcPassword);
        ...
```

A linha *Class.forName("com.inet.tds.TdsDriver").newInstance();* é usada se o sistema utiliza o banco de dados SQLServer e portanto utiliza o driver *com.inet.tds.TdsDriver*. Como o banco de dados do projeto utiliza o Interbase, o driver usado foi o *interbase.interclient. Driver*. Depois de estabelecida a conexão, o banco de dados já pode ser manipulado através da linguagem Java.

Para executar um comando SQL, é preciso criar um objeto Statement do pacote java.sql. O objeto Connection que foi obtido da chamada DriverManager.getConnection pode ser usado para criar objetos Statement (Almeida, 2002):

```
Statement stmt = con.createStatement();
String command = "UPDATE variaveis SET checado = 'Sim' WHERE idVariaveis = " +
                    idVariaveis[i] + " and checado = 'Não'";
stmt.executeUpdate(command);
```

Posteriormente, coloca-se o statement que se quer executar em uma String (veja command). Então, chama-se o método executeUpdate da classe Statement. O método executeUpdate retorna uma contagem de linhas que foram afetadas pelo comando SQL (Almeida, 2002). Os comandos podem ser ações como INSERT (cadastrar), UPDATE (alterar) ou DELETE (apagar) assim como comandos de definição de dados como CREATE TABLE e DROP TABLE. No entanto, para executar consultas SELECT é utilizado o método executeQuery (consulta) (Horstman e Cornell, 2001).

Como pode ser observado na documentação apresentada no Anexo III, o construtor da classe NormalizacaoDAO.java recebe como parâmetros um objeto Connection que contém uma conexão ao banco de dados e uma referência ao objeto Sistema.

A classe NormalizacaoDAO.java contém um método que retorna uma coleção de normalizações do dessalinizador especificado no parâmetro e que está na faixa de datas solicitada (entre as datas início e fim). Este método recebe como parâmetros o id do dessalinizador, a data inicial e final das normalizações, como também a quantidade de normalizações que vai estar na coleção. Se a quantidade for <= 0, todas as normalizações são listadas. Se o idDessalinizador for zero são listados todos os dessalinizadores. Para executar consultas SELECT, esta classe utiliza o método executeQuery (consulta). O método executeQuery retorna um objeto do tipo ResultSet. Esse objeto possui um conjunto de tuplas que foram afetadas pela consulta. Como por exemplo:

```
...
/*Executando uma consulta*/
ResultSet rsVar = conDB.executeQuery(
"SELECT DISTINCT v.idVariaveis, v.idDessalinizador, v.idLimites, v.data, l.data, " +
"v.p3Ultimo, v.p4Ultimo, v.v1Ultimo, v.v2Ultimo, v.temperatura, v.ph, v.tdsA, " +
"d.areaMembrana, l.p3Simulado, l.p4Simulado, l.qpSimulado, l.qcSimulado, " +
"l.tSimulado, l.tdsASimulado " +
"FROM variaveis v, dessalinizador d, limites l " +
"WHERE " + idD + "
AND v.idDessalinizador = d.idDessalinizador AND " +
" l.idDessalinizador = d.idDessalinizador " +
" AND v.data >= '" + dataInicio + "' AND v.data <= '" + dataFim + "' " +
" AND l.data >= '" + dataInicio.substring(0, dataInicio.indexOf(" ")) + "' AND l.data <= '" +
dataFim.substring(0, dataFim.indexOf(" ")) + "' " +
"ORDER BY v.data DESC");
...
```

Como se pode observar no código acima, para se fazer uma normalização, a classe NomalizacaoDAO.java acessa os dados contidos nas tabelas dessalinizador, variáveis e limites do banco de dados. Uma vez "capturados" esses dados, essa classe se encarrega de criar um objeto da classe Normalizacao. Para isso, a classe NormalizacaoDAO dispõe de um método *private* que recebe como parâmetro um objeto do tipo ResultSet. Esse método obtém o valor de cada atributo da tupla usando objetos das classes Variaveis.java, Limites.java e Dessalinizador.java que são usados como parâmetros para instanciar a classe Normalizacao, criando desse modo um objeto do tipo Normalizacao. Assim, para cada conjunto de dados, será gerada uma normalização, formando no final da consulta uma coleção de normalizações na faixa de datas solicitada pelo usuário. Como um exemplo, para a classe Variaveis.java, o valor de cada atributo da tupla é obtido da seguinte maneira:

```
...
Variaveis var = new Variaveis();
var.setIdVariaveis(rsVar.getInt("idVariaveis"));
var.setIdDessalinizador( rsVar.getInt("idDessalinizador"));

//variaveis
        var.setP3Ultimo(rsVar.getFloat("p3Ultimo"));
        var.setP4Ultimo(rsVar.getFloat("p4Ultimo"));
        var.setV1Ultimo(rsVar.getFloat("v1Ultimo"));
        var.setV2Ultimo(rsVar.getFloat("v2Ultimo"));
        var.setT(rsVar.getFloat("temperatura"));
        var.setTdsA(rsVar.getFloat("tdsA"));
...
```

2.3.4. Cálculo do Índice de Saturação de Langelier (ISL)

Além de realizar a normalização do sistema de dessalinização, o módulo de gerência é capaz de realizar o cálculo do ISL da água de alimentação do dessalinizador piloto utilizando as Equações 22 a 27 apresentadas no capítulo I. Para isso, é necessário acessar no banco de dados (BD) os valores do STD (sólidos totais dissolvidos), da concentração do cálcio como $CaCO_3$, da temperatura, da alcalinidade como $CaCO_3$ e do pH da solução de alimentação. Esses dados relacionados à análise físico-química da água de alimentação são cadastrados no BD pelo usuário. O cálculo do ISL é realizado na classe Analise.java cuja documentação encontra-se no Anexo IV. A Figura 2.11 apresenta um diagrama que representa o algoritmo adotado para se determinar o Índice de Saturação de Langelier (ISL). As variáveis que se encontram representadas no diagrama dentro de círculos são as cadastradas no banco de dados e que alimentam as equações durante os cálculos. Cada etapa descrita nesse algorítmo corresponde a um método *get* da classe Analise.java que retorna o valor do parâmetro desejado. Logo, de acordo com a Figura 2.11, adotou-se o seguinte algoritmo:

1. Cálculo do efeito da concentração (A) através da Equação 24. Variável de entrada: sólidos totais dissolvidos da alimentação (STD);
2. Obtenção do efeito da temperatura (B) utilizando a Equação 25, tendo como variável de entrada a temperatura da alimentação (T);
3. Cálculo do efeito da dureza em cálcio como carbonato de cálcio (C) utilizando a Equação 26, tendo como variável de entrada a concentração de Ca^{+2} da alimentação como $CaCO_3$;
4. Cálculo do efeito da alcalinidade como carbonato de cálcio (D) através da Equação 27, tendo como variável de entrada a concentração de íons bicarbonatos da alimentação;

5. Determinação do pH de saturação (pH_s) da alimentação através da Equação 23 tendo como parâmetros de entrada os efeitos da concentração (A), da temperatura (B), da dureza em cálcio como carbonato de cálcio (C) e da alcalinidade como carbonato de cálcio (D);
6. Cálculo do Índice de Saturação de Langelier (ISL) através da Equação 22a tendo como variável de entrada o pH da alimentação e como parâmetro de entrada o pH de saturação (pH_s).

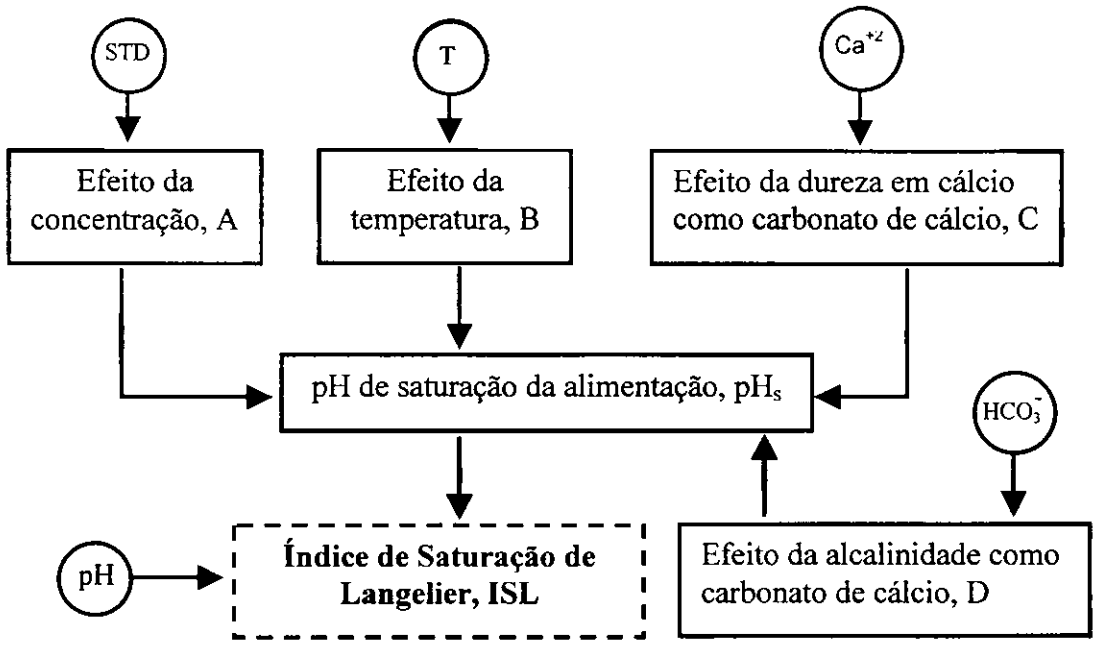


Figura 2.11: Diagrama que representa o algoritmo adotado para se determinar o Índice de Saturação de Langelier (ISL).

2.3.5. Interface gráfica (JSP)

O servidor utilizado para exibir as páginas HTML dinâmicas é o servidor tomcat do projeto Jakarta (http://jakarta.apache.org). Toda a interface do projeto foi desenvolvida em JSP.

Quando o cliente faz a solicitação de um arquivo JSP, um servlet especial verifica se a página é mais nova do que o servlet que a representa. Se for, automaticamente, o servlet é regenerado a partir da JSP e recompilado. Na próxima requisição, o servidor irá direcionar o request ao servlet gerado e este irá produzir uma resposta que será entregue ao servidor que irá passar a resposta http ao browser do cliente. A Figura 2.12 dá um exemplo de como este mecanismo funciona quando é solicitada a normalização de um dessalinizador.

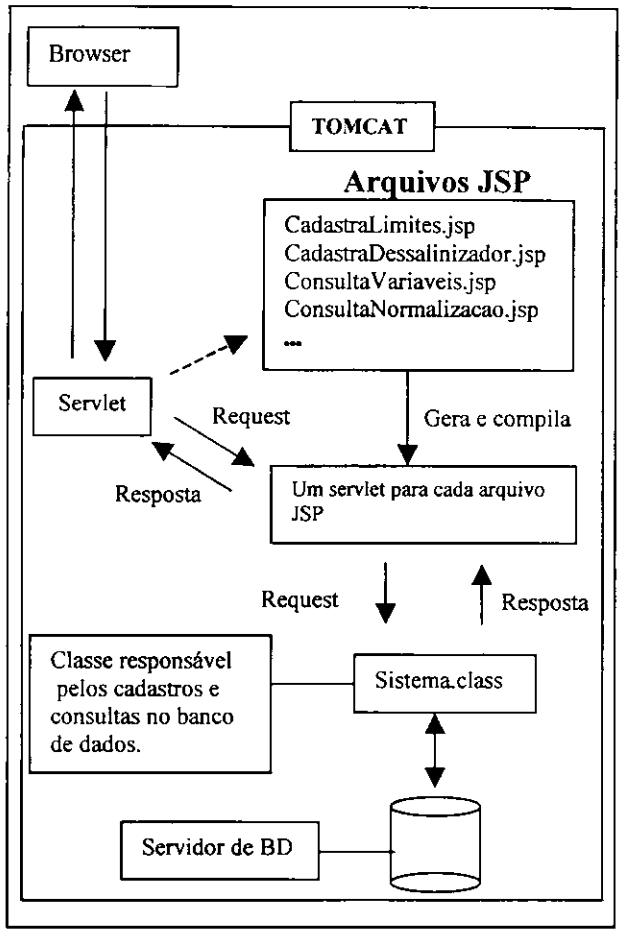


Figura 2.12: Servidor do projeto de monitoração remota de dessalinizadores.

Para realizar a normalização existe um conjunto de páginas JSP associadas. Esses arquivos são responsáveis por colher as informações do formulário html e enviar esses dados para a classe Sistema para que esta realize as ações adequadas a estes dados. Na realidade, a classe Sistema é apenas uma classe de interface entre o JSP e o banco de dados. Como foi comentado anteriormente, durante os cálculos dos parâmetros normalizados é necessário acessar as tabelas dessalinizador, variáveis e limites de operação do banco de dados. Nesse sentido, foi criado um "bean" Normalizacao.java que contém as equações do processo de osmose inversa que são alimentadas pelos dados encapsulados nos "beans" já existentes Dessalinizador.java, Variaveis.java e Limites.java. Para cada um desses "beans" existe uma classe de manipulação de dados associada (NormalizacaoDAO.java, DessalinizadorDAO. java, VariaveisDAO.java e LimitesDAO.java). Logo, a classe Sistema apenas chama essas classes quando o usuário solicita alguma ação no browser como, por exemplo, um cadastro das variáveis de medidas, uma consulta da normalização, etc.

Um procedimento análogo é seguido quando o cliente solicita o cálculo do ISL na determinação do potencial de incrustação por carbonato de cálcio. Nesse caso, a classe Sistema acessa os dados da tabela análise fisico-química do banco de dados através da classe de manipulação AnaliseDAO.java.

CAPÍTULO III

Resultados e Discussões

3.1. A interface gráfica

A primeira página a ser aberta quando é feita uma requisição ao servidor é o index.jsp (Figura 3.1). Este arquivo manipula uma página web responsável pela verificação de login e senha. Quando esses dados são fornecidos, o browser envia essas informações para o servidor tomcat. No servidor, o arquivo index.jsp faz uma requisição a classe Sistema para que ele verifique se o login e a senha são válidos (Almeida, 2002). A Figura 3.2 ilustra mais detalhadamente os passos que são efetuados na checagem de login. Se houver alguma tentativa de acesso a qualquer outra página do sistema sem antes passar pelo login, o sistema impede este acesso e redireciona o endereço para a tela de login (index.jsp). Isto é feito através do objeto "session" do servidor jsp que cria uma sessão para cada usuário que entra no sistema. Se alguém tenta fazer alguma consulta ao banco de dados sem antes fazer login, basta verificar se a sessão é nula, se este for o caso, então o sistema não deve permitir qualquer operação. O arquivo responsável por isso é o ChecaLogin.jsp (Figura 3.2). Segue abaixo o código jsp que verifica o login e cria uma sessão caso o login e senha sejam válidos (Almeida, 2002):

```
<%@ page import="sismodes.*, sismodes.beans.*"%>
<%
String login = request.getParameter("login");
String senha = request.getParameter("senha");

Sistema sistema = new Sistema();
Usuario usuario = sistema.loginValido(login, senha);
//Se o usuário for válido, entrão cria um atributo na sessão
if(usuario != null){
        sistema.setUsuario(usuario);
        session.setAttribute("sistema", sistema);
%>
<jsp:forward page="Principal.jsp"/>
<%}%>
<jsp:forward page="index.jsp?mensagem=Login+inexistente" />
```

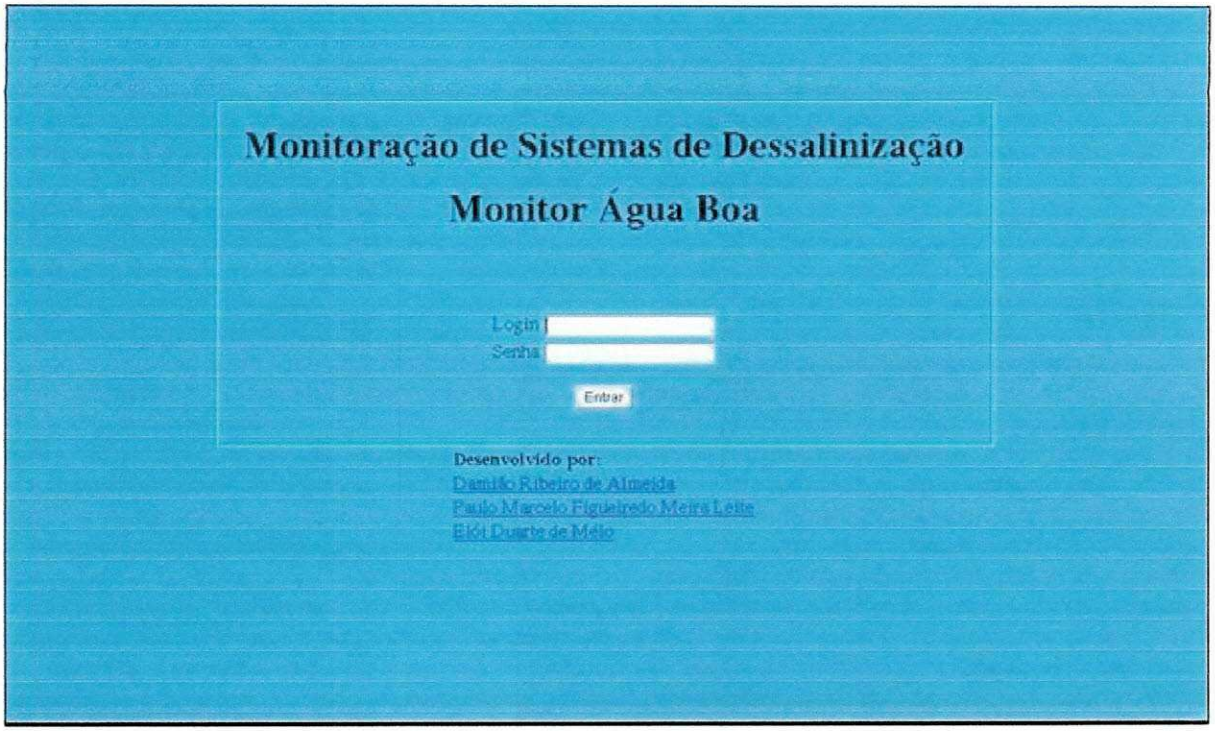


Figura 3.1: Primeira página do sistema.

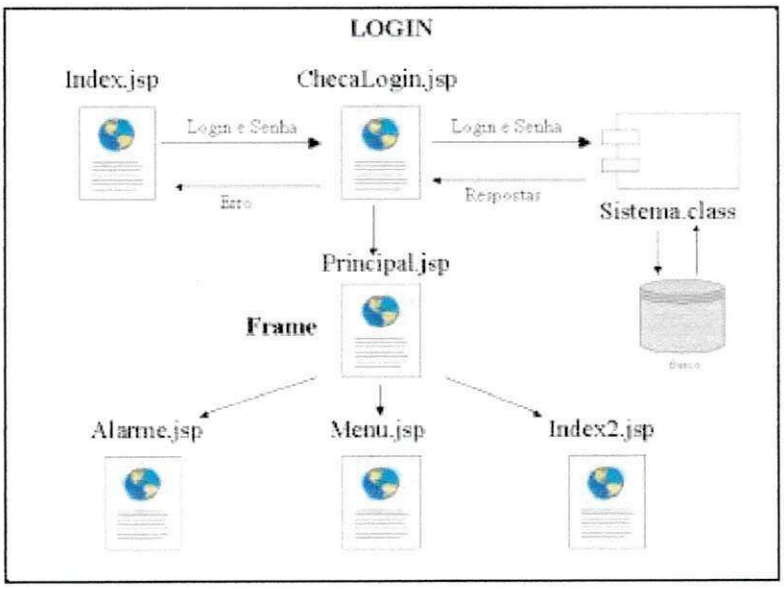


Figura 3.2: Efetuando login do sistema.

Ao efetuar o login no sistema, o servidor envia para o cliente um frame que é divido em três partes: alarme, menu e tela principal.

3.2. Cadastro dos dados

Na janela de menu apresentada na Figura 3.3 é mostrado as opções de cadastro, consulta, alteração e exclusão de município, poço, dessalinizador, análise fisico-química, variáveis e limites, além da consulta da normalização e verificação de alarmes. Existe também uma opção para se conectar imediatamente ao dessalinizador desejado para verificar as suas medidas nesse mesmo instante (consulta remota).

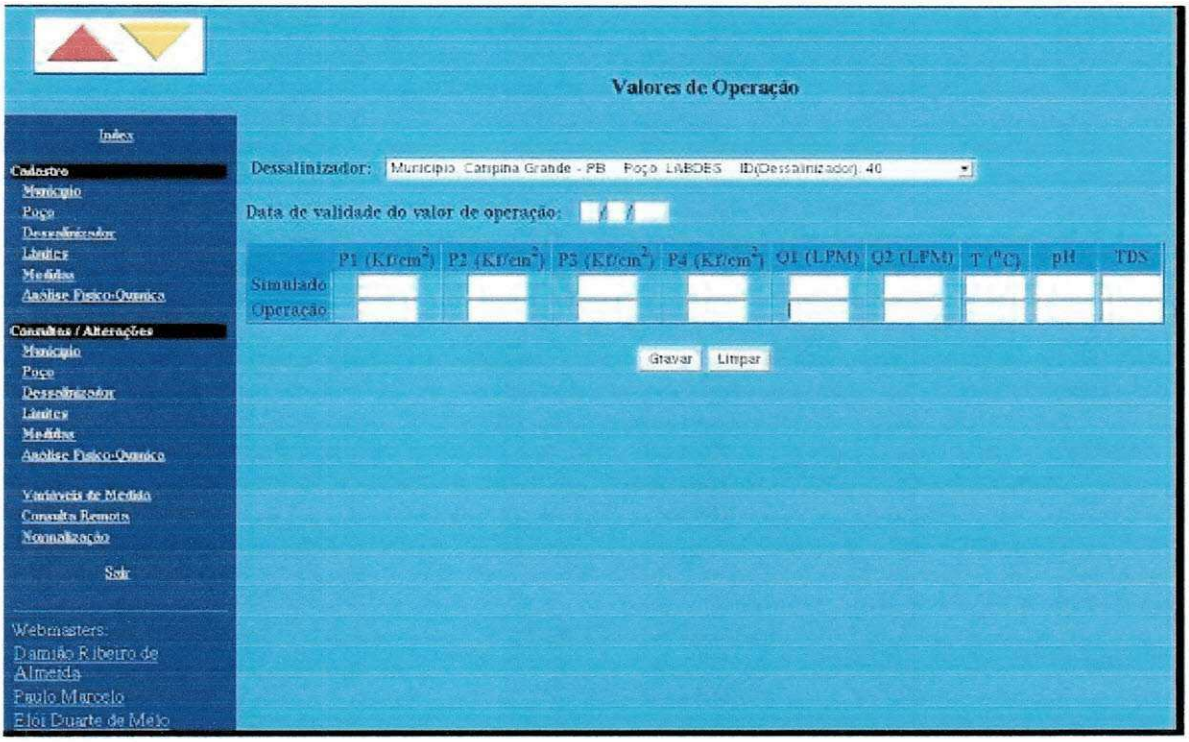


Figura 3.3: Organização da interface e cadastro de limites

Na janela principal temos um formulário de cadastro de limites com os campos: simulado e operação para as variáveis P1, P2, P3, P4, Q1, Q2, T, pH e TDS. Ao clicar no botão "gravar" o formulário será submetido ao arquivo CadastraLimites.jsp. Este arquivo é encarregado de encapsular os dados do formulário em um bean chamado Limites.java e entregar este bean a classe Sistema para que ela efetue o cadastro do limite no banco de dados. Os valores simulados são os limites cadastrados a partir do projeto do sistema ou na sua partida inicial, enquanto que os valores de operação são aqueles cadastrados após a sua estabilização e que são usados como referências para o acionamento dos alarmes. O cadastro de dessalinizador, medidas (variáveis) e análise físico-química são semelhantes ao processo de cadastro de limites.

Para se cadastrar os dados referentes ao dessalinizador como, por exemplo, número de membranas e filtros, área total do sistema de membranas, potência, marca e outros é necessário clicar em "Dessalinizador" no menu (Figura 3.4). Em seguida, seleciona-se o dessalinizador cujas informações deseja-se cadastrar e clica-se em "Pesquisar". Logo após, cadastra-se as informações correpondentes a cada campo e clica-se em "Gravar" abaixo do formulário. Para o cálculo do coeficiente de transferência de massa (CTM) durante a normalização é essencial o preenchimento do campo "Área total das membranas (m^2)".

Para se cadastrar as medidas (variáveis) é necessário antes indicar a quantidade desejada (Figura 3.5) e clicar em "Próximo". Em seguida, cadastra-se os valores de P1, P2, P3, P4, Q1, Q2, T, pH e TDS correspondente a cada data e clica-se em "gravar" (Figura 3.6). O cadastro das medidas desse modo só é necessário enquanto os dados do computador embarcado não estão sendo enviados ao servidor por algum motivo e se deseja atualizá-los para um melhor acompanhamento do sistema de dessalinização durante esse período. Esta forma de se cadastrar as medidas no banco de dados é a única possível quando se pretende monitorar um sistema de dessalinização que ainda não dispõe de um computador embarcado. No caso do sistema de dessalinização piloto (SDP) que vem sendo monitorado no LABDES, as variáveis P1, P2, pH e TDS são cadastradas pelo usuário durante a consulta, enquanto as demais são enviadas remoticamente.

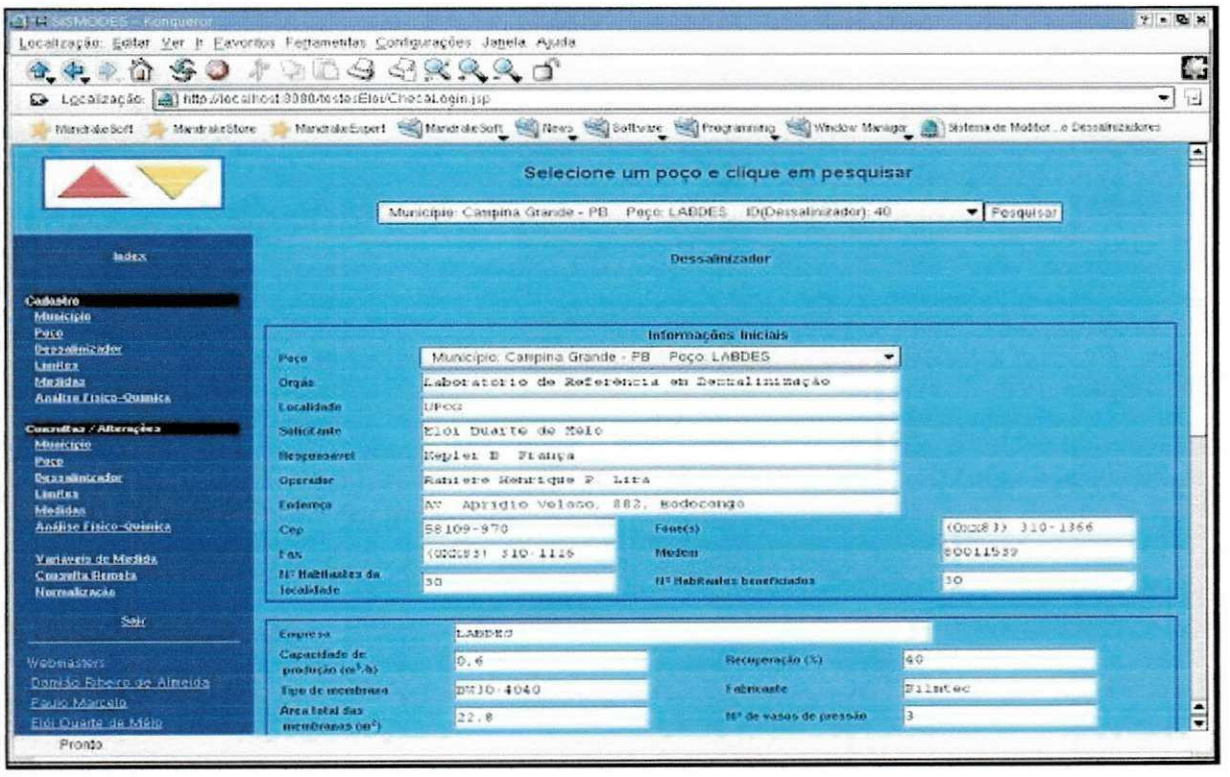


Figura 3.4a: Cadastro das informações referentes ao dessalinizador no banco de dados

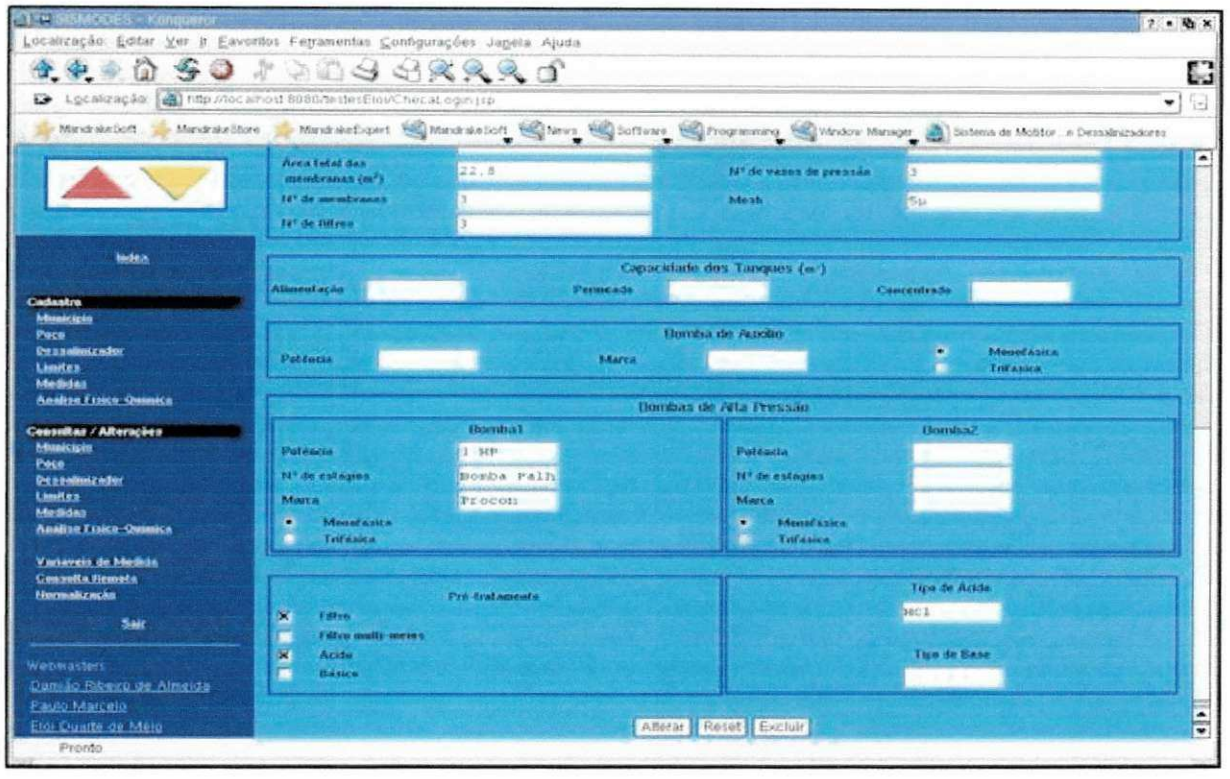


Figura 3.4b: Cadastro das informações referentes ao dessalinizador (continuação)

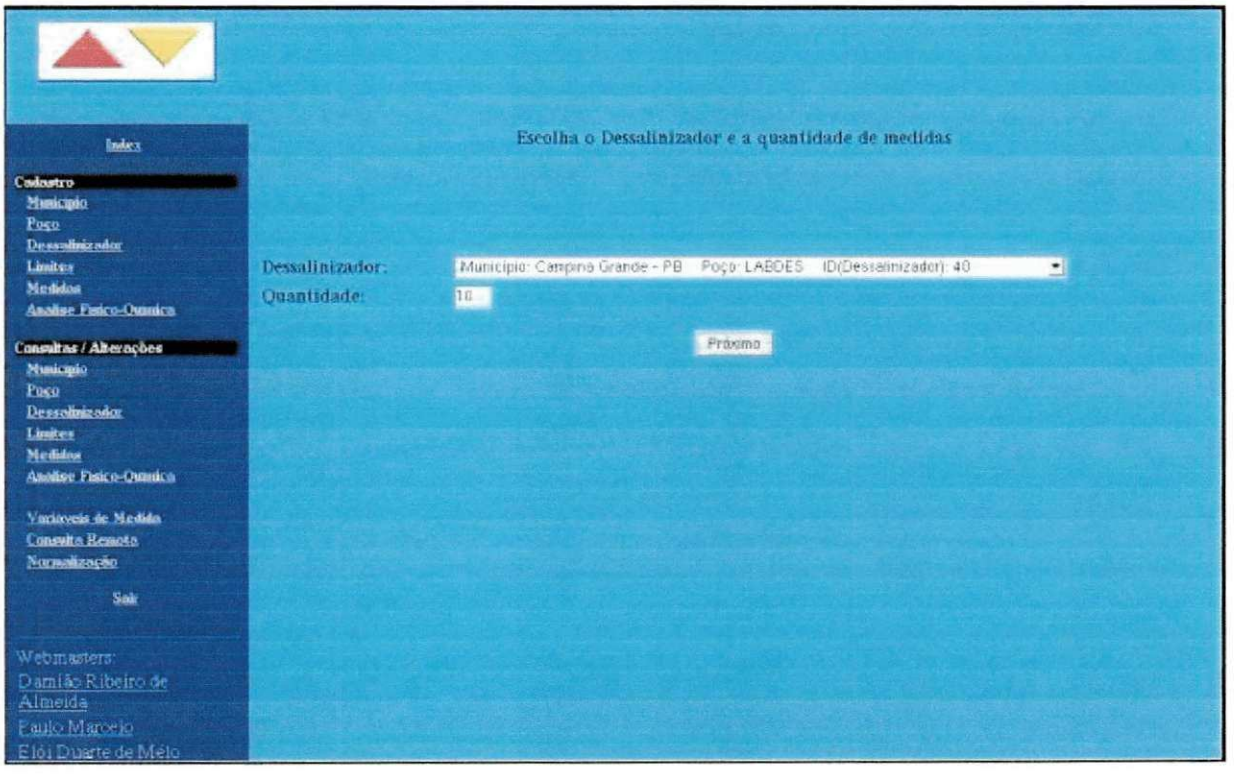


Figura 3.5: Quantidade de medidas a ser cadastradas no banco de dados

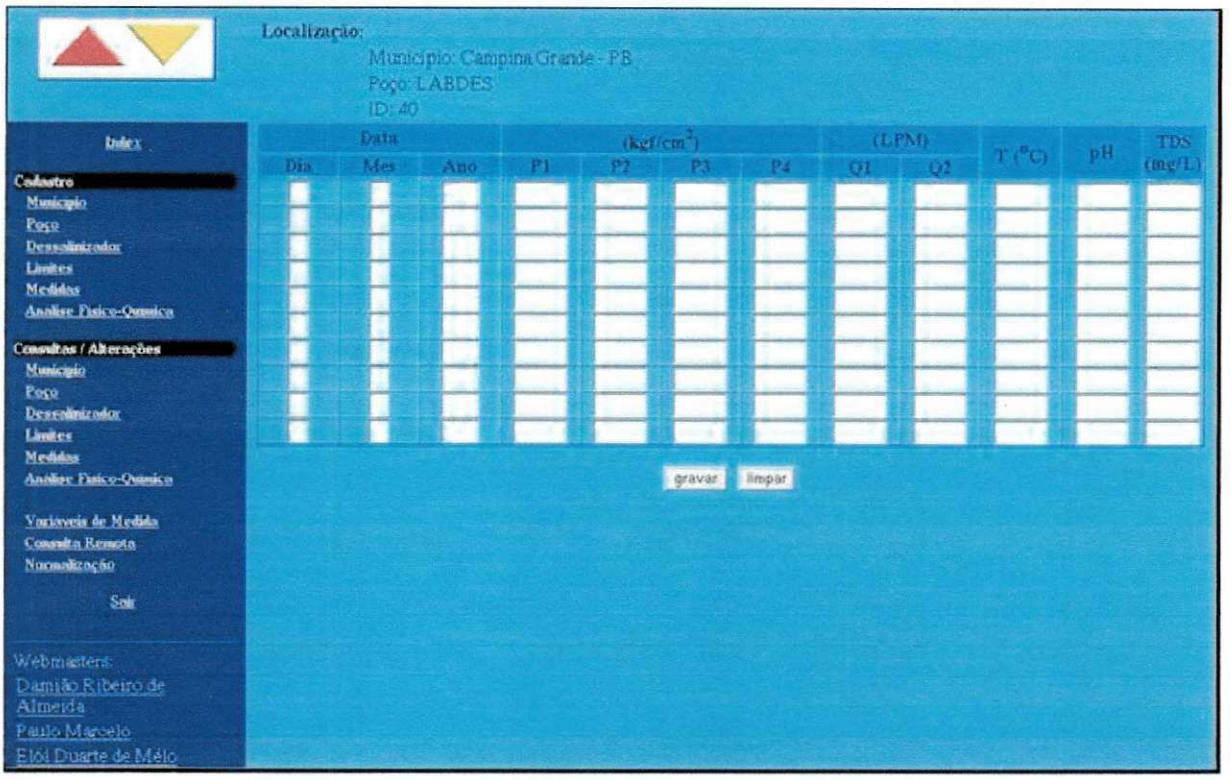


Figura 3.6: Cadastro de medidas no banco de dados

3.3. Consulta, alteração e exclusão dos dados

Como mencionado anteriormente, as opções de consulta, alteração e exclusão de município, poço, dessalinizador, análise fisico-química, variáveis e limites também são apresentadas na tela de menu. Por exemplo, na Figura 3.7 temos uma tela de um resultado da consulta da análise físico-química da água que alimenta o SDP. No canto superior da tela há um select que exibe todas as análises cadastradas e é onde se pode selecionar uma delas para poder observar os seus dados. Depois de escolhida a análise, os seus dados são exibidos no formulário logo abaixo. Junto com os dados cadastrados, na Figura 3.8, aparece na tela o resutado do Índice de Saturação de Langelier (ISL) correspondente a análise fisico-química da água de alimentação, sendo este calculado no bean Analise.java através das Equações 23 a 27 usando os dados cadastrados no banco de dados que são fornecidos pela classe Sistema.java através da classe de manipulação AnaliseDAO.java. No fim da página existem três botões alterar, reset e excluir, que são responsáveis, respectivamente, por cadastrar as alterações feitas para a análise, cancelar a alteração e excluir a análise. Para dessalinizador, o processo de consulta é semelhante.

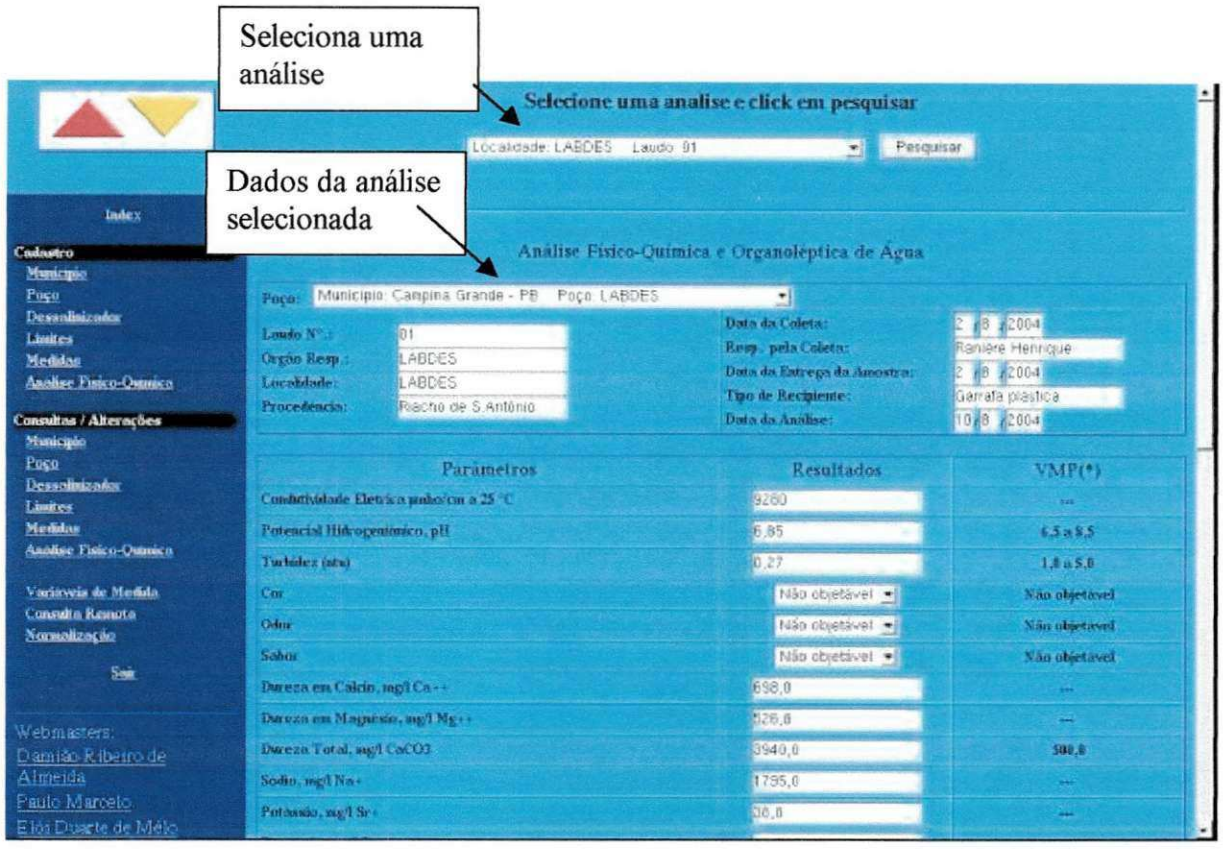


Figura 3.7: Consulta de uma análise físico-química.

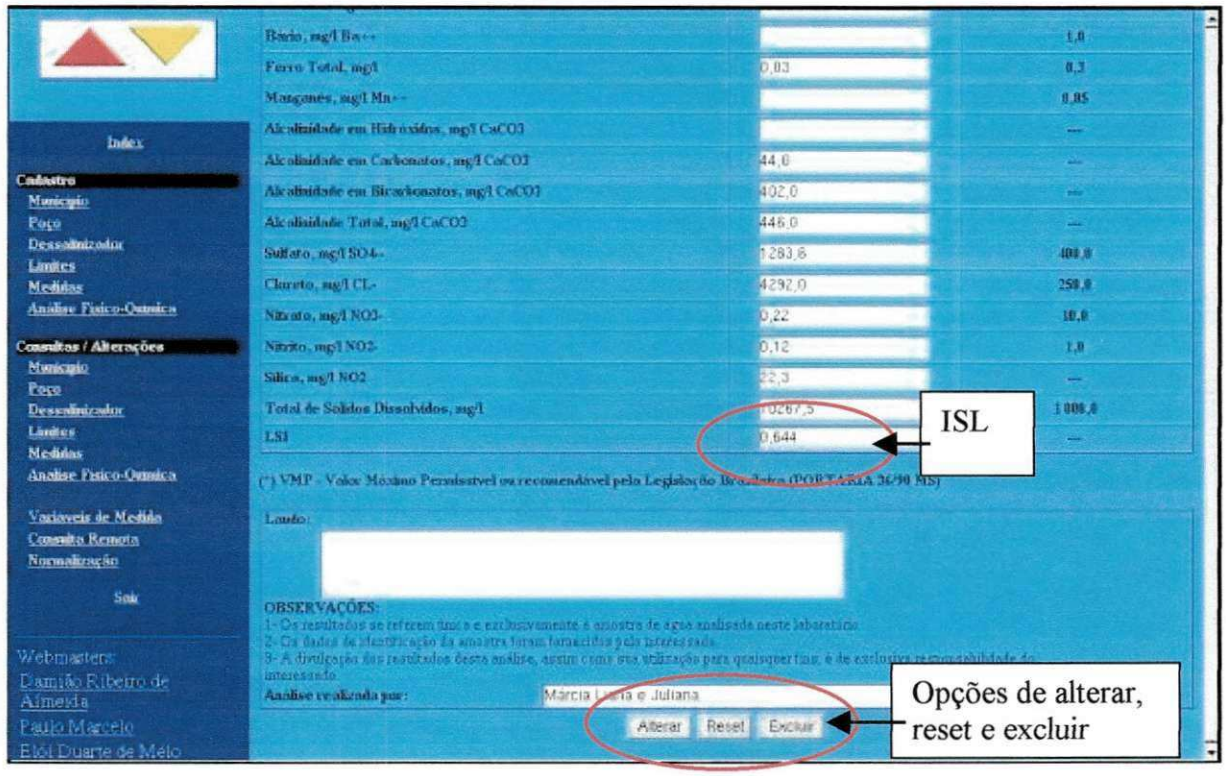


Figura 3.8: As opções de alterar, reset e excluir uma análise físico-química.

Para se fazer uma consulta de limites é necessário clicar em "Limites" na parte de consultas do menu (Figura 3.9).

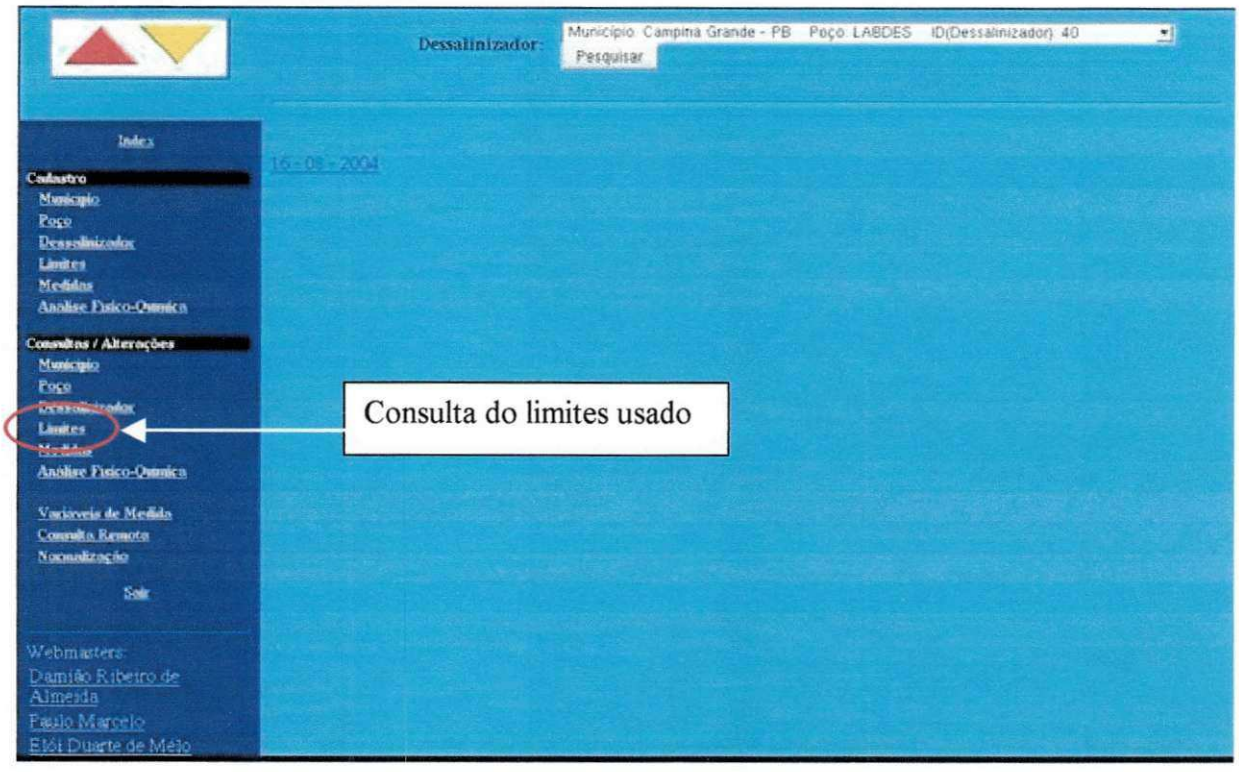


Figura 3.9: Consulta de limites

Em seguida, deve-se informar qual é o dessalinizador e a data cujos limites se deseja consultar. É interessante informar que esta data corresponde à data da partida do sistema ou a data de quando ocorre alguma modificação na operação do mesmo como, por exemplo, uma limpeza química. Ao clicar-se na data desejada, tem-se o resultado da consulta, que é a tabela mostrada na Figura 3.10. Esta tabela mostra os valores de todas as variáveis simuladas usadas como referência para se realizar a normalização para o dessalinizador desejado e na data solicitada. Os valores de operação são usados como referência na geração de alarmes.

Para se fazer uma consulta de variáveis é necessário clicar em "Medidas" no menu (Figura 3.11). Em seguida, deve-se informar qual é o dessalinizador e a faixa de data cujas variáveis se deseja consultar. Na Figura 3.11, o campo select apresenta uma lista de dessalinizadores cadastrados. Logo abaixo existem alguns espaços para se colocar a faixa de data de que se deseja efetuar a consulta. Dessa forma, a página irá mostrar as variáveis do dessalinizador selecionado na faixa de tempo descrita.

A validação dos campos dia, mês e ano, ou seja, assegurar que os valores nesses campos são realmente números e que esses números condizem com o calendário, incluindo

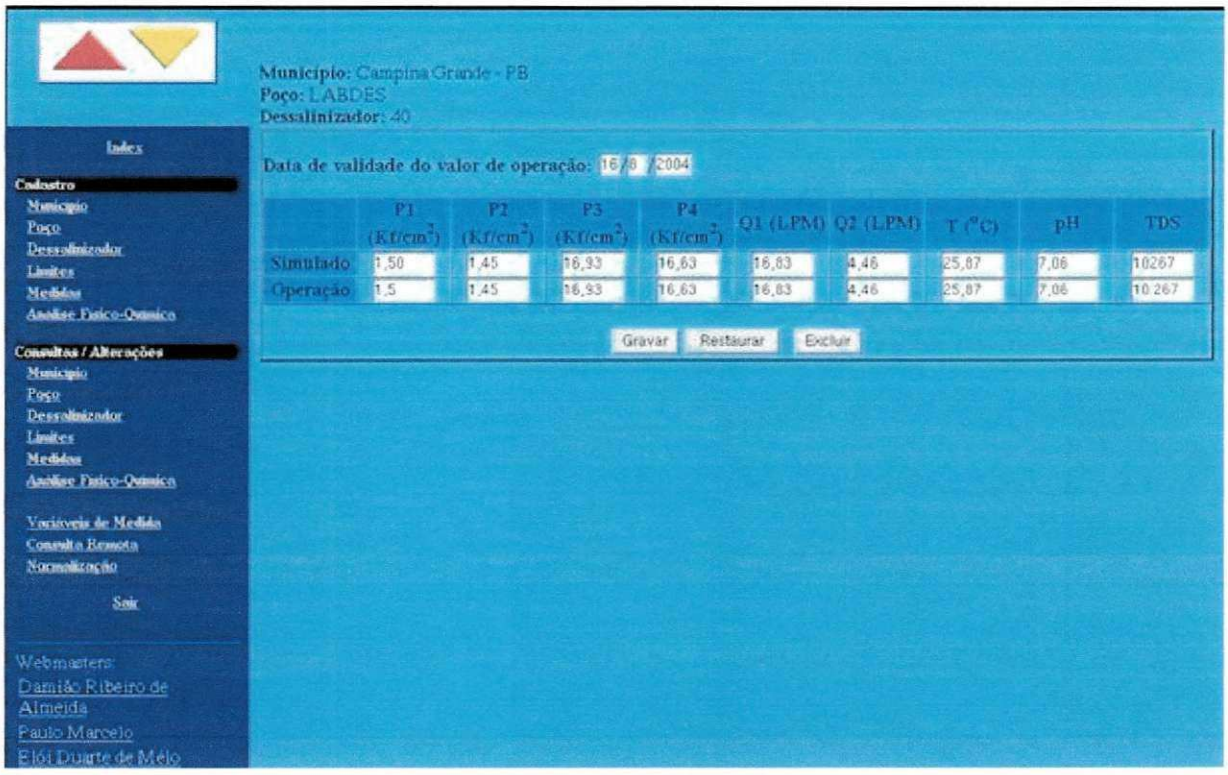


Figura 3.10: Tabela de limites

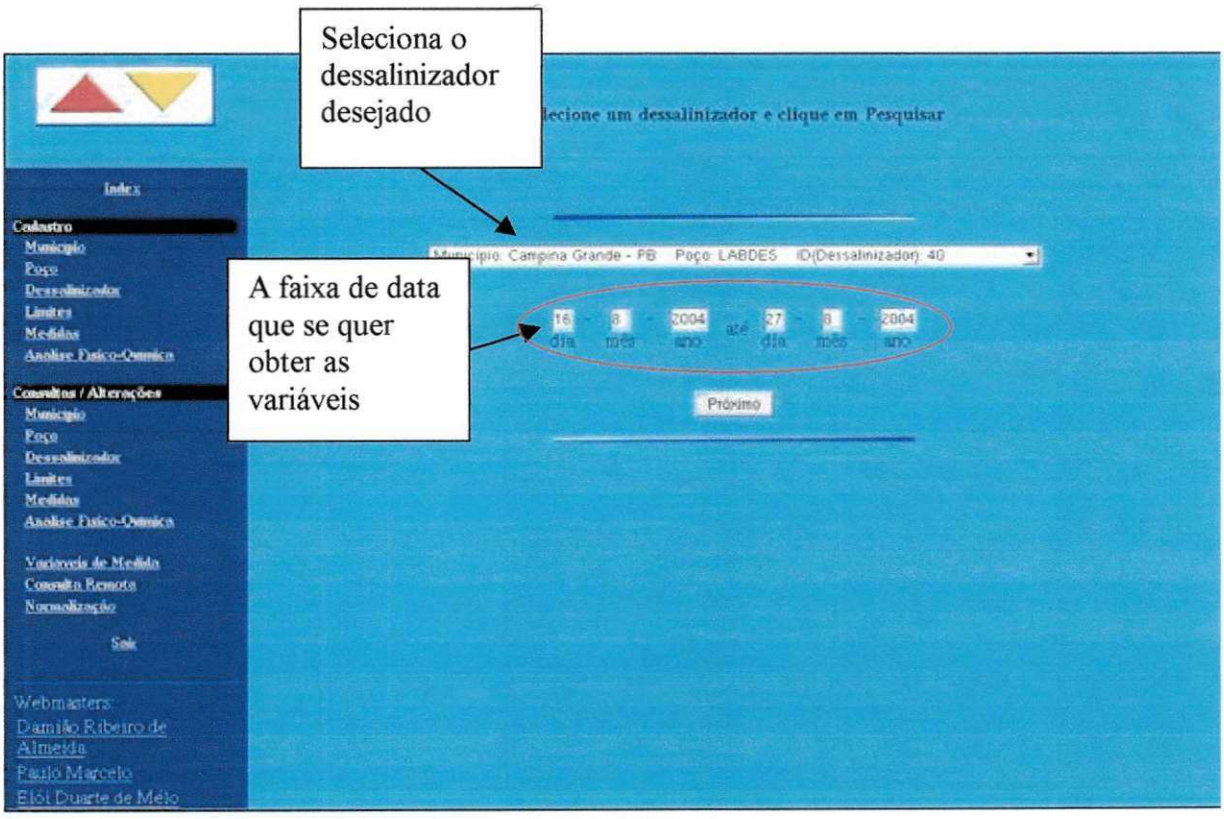


Figura 3.11: Consulta de variáveis

anos bissextos, é feito em javaScrit que é uma linguagem que permite colocar lógica em páginas HTML. Dessa forma o cliente recebe uma mensagem de erro, caso a data fornecida esteja incorreta, muito mais rápido do que processar essas informações no servidor (Almeida, 2002).

O resultado da consulta é a tabela mostrada na Figura 3.12. A tabela mostra os valores de todas as variáveis: P1, P2, P3, P4, Q1, Q2, T, pH e TDS para o dessalinizador desejado e na faixa de datas solicitada. Os valores destas variáveis podem ser tanto os cadastrados pelo usuário conforme descrito anteriormente (ítem 3.2) como os dados enviados pelo computador embarcado. Como ainda não está sendo possível receber os valores das variáveis P1, P2, pH e TDS do computador embarcado, estas variáveis são cadastradas automaticamente com valores zero no banco de dados. Logo, os valores destas variáveis devem ser alterados pelo usuário, durante a consulta, para os valores lidos nos instrumentos de medidas instalados no sistema de dessalinização piloto (SDP). Para isso, existe no fim da página um botão "Gravar" que é responsável por cadastrar as alterações feitas nas variáveis.

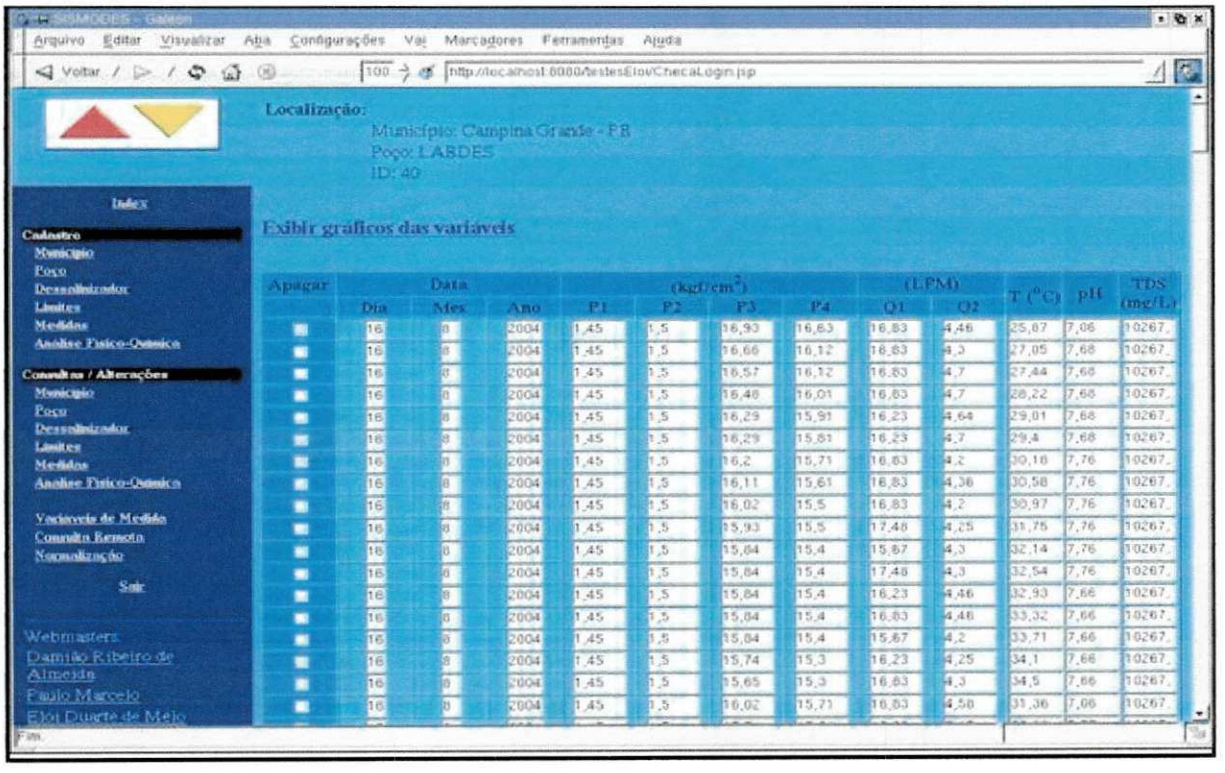


Figura 3.12a: Tabela de variáveis

Existe também um botão "Restaurar" que é responsável por cancelar a alteração, voltando aos dados iniciais. Existe também ao lado de cada conjunto de medidas para uma determinada data, um campo "Apagar", que serve para o usuário retirar do banco de dados

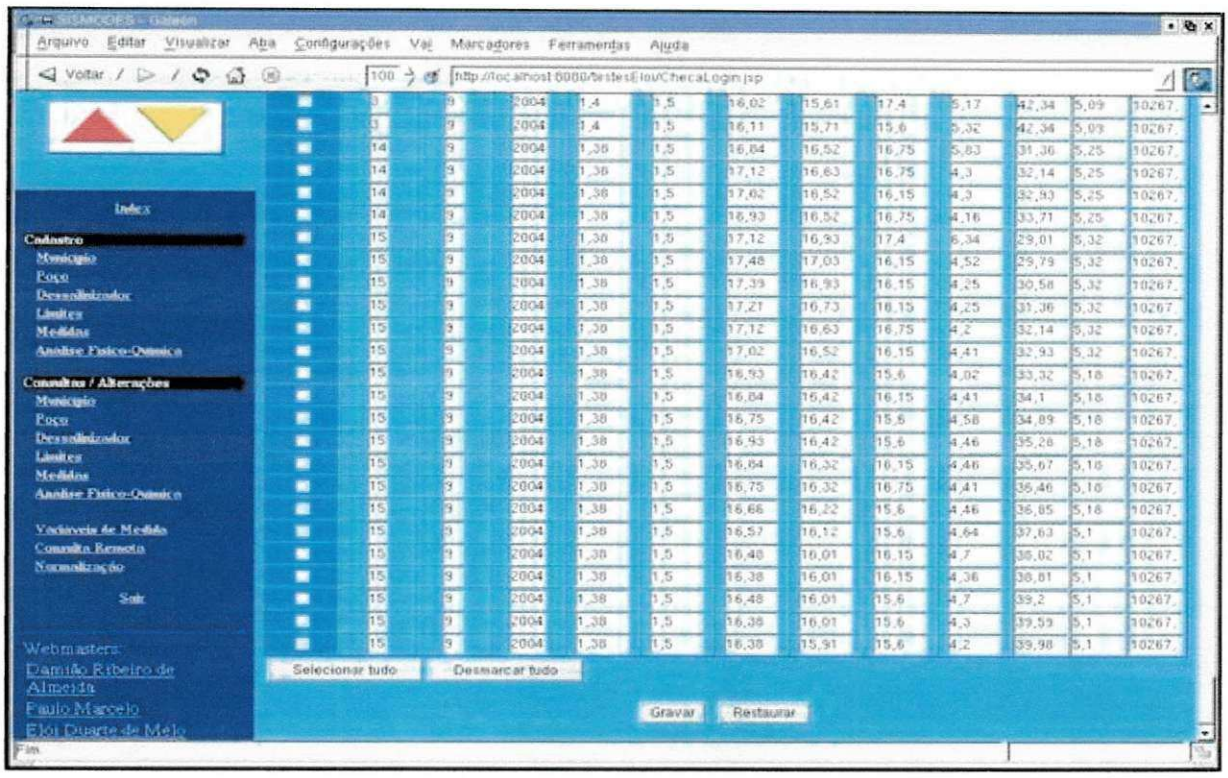


Figura 3.12a: Tabela de variáveis (continuação)

todas as variáveis recebidas remoticamente ou cadastradas na data selecionada. Para isso, é necessário acionar o botão "Gravar" após selecionar o campo "Apagar". Se o usuário preferir retirar todos os dados, existe o botão "Selecionar tudo" que seleciona todos os campos da opção "Apagar". Caso o usuário desista de retirar todos os dados por esta opção, ele pode acionar o botão "Desmarcar tudo".

Para exibir os gráficos das variáveis em função do tempo de operação do SDP, basta clicar em "Exibir gráficos das variáveis" na Figura 3.12. Ao fazer isso, será chamado o arquivo ExibeGraficoVariaveis.jsp. Este arquivo apresenta uma relação dos gráficos que deve ser escolhido pelo usuário (Figura 3.13).

Ao selecionar a opção desejada e clicar em "Pesquisar", será chamado o arquivo GraficoVariaveis.jsp que tem a finalidade de construir o gráfico desejado pelo usuário usando o pacote jspChart.jsp (Anexo V) que encontra-se disponível gratuitamente na internet (www.jspchart.com). Na Figura 3.14, temos o gráfico da pressão de entrada das membranas em função do tempo. Os gráficos das demais variáveis encontram-se representadas pelas Figuras 3.15 a 3.21 no Anexo VI.

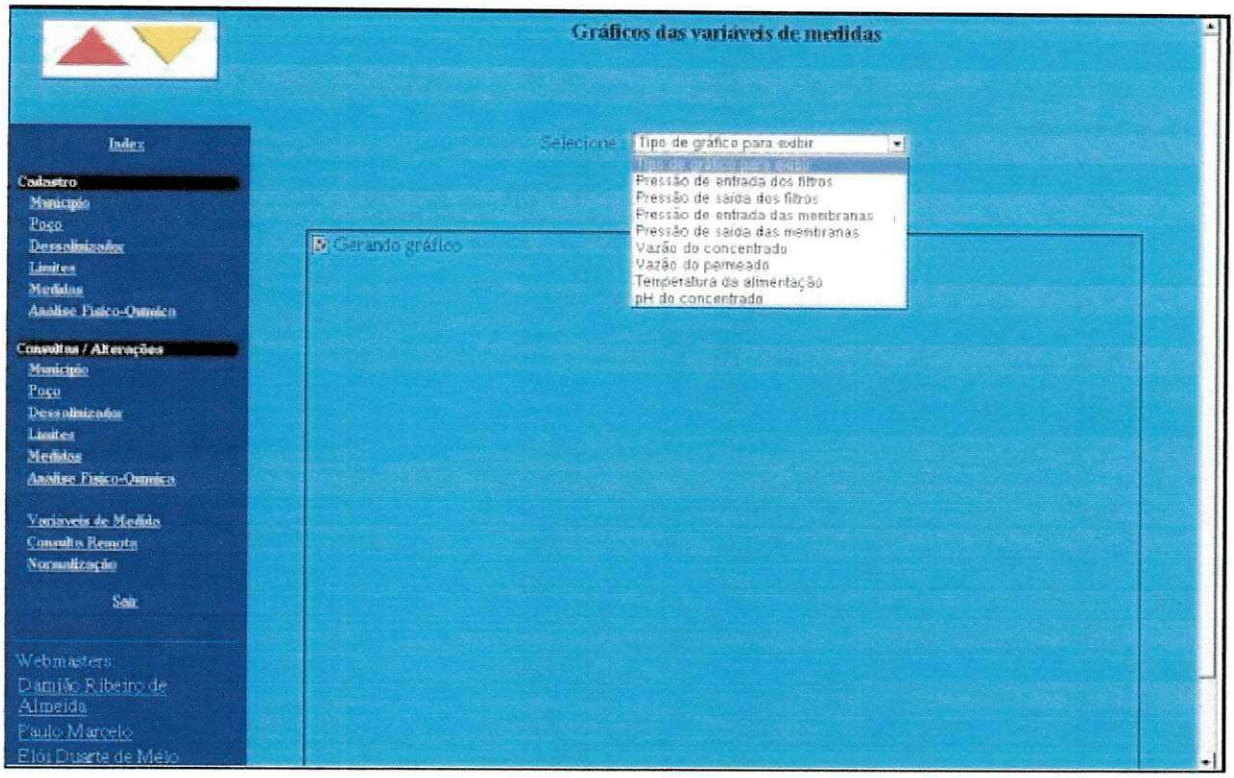


Figura 3.13: Seleção do tipo de gráfico das varáveis

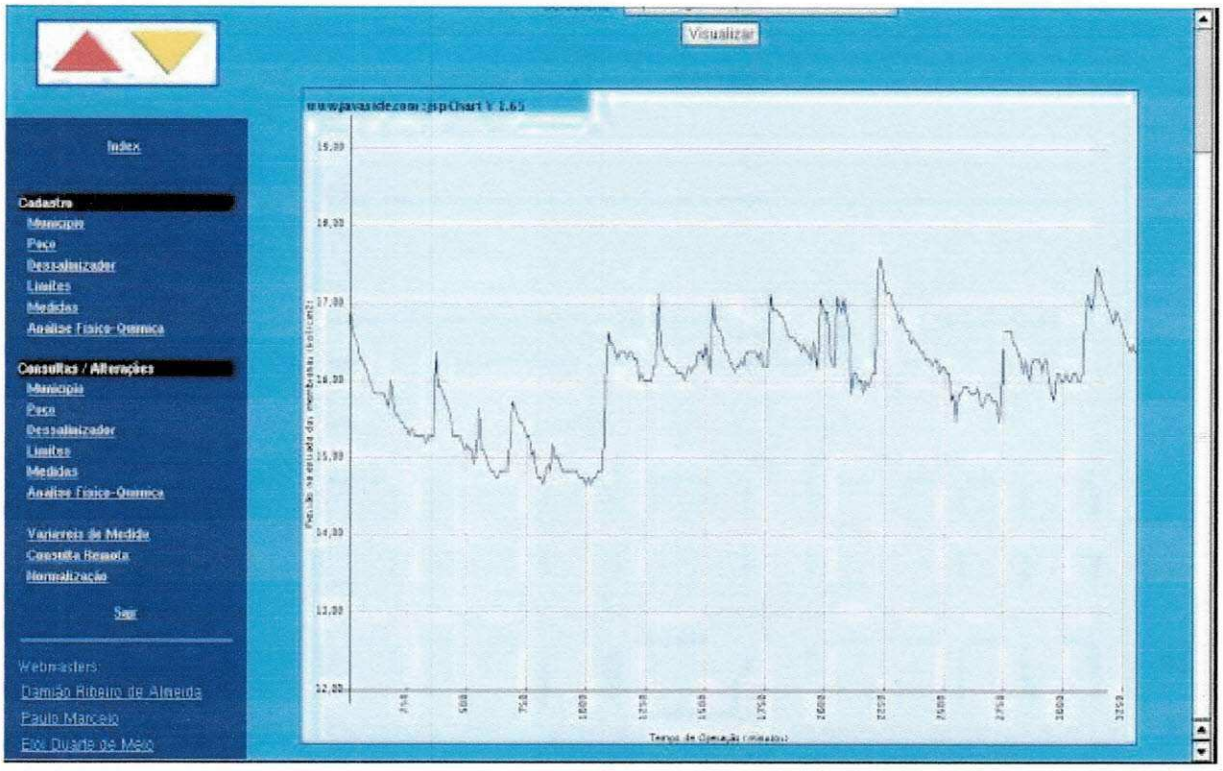


Figura 3.14: Gráfico da pressão de entrada da membrana em função do tempo

3.4. Normalização do sistema

Para se efetuar a normalização de um sistema de dessalinização é necessário antes se certificar que todos os dados correspondentes à faixa de data solicitada foram cadastrados e/ou enviados pelo computador embarcado. Deve-se também ter o cuidado de cadastrar os valores simulados dos limites de operação deste sistema na opção cadastrar limites do menu conforme descrito no item 3.2. Os valores simulados são as variáveis de medidas de referência que são comparados, durante a normalização, com os valores enviados pelo computador embarcado obtidos em tempo de operação. Estes valores de referência podem ser os valores de projeto do sistema de dessalinização ou os valores da partida do sistema, como também os valores obtidos após alguma modificação na operação do mesmo como, por exemplo, uma limpeza química. Além disso, deve-se cadastrar a área das membranas na opção cadastrar dessalinizador, pois este dado é usado no cálculo do coeficiente de transferência de massa (CTM) durante a normalização do sistema de dessalinização piloto.

Quando se é feita uma consulta de normalização é necessário que se informe que normalização se deseja consultar. Isto é feito ao clicar em "normalização" no menu. Ao fazer isto aparecerá o campo select que apresenta uma lista de dessalinizadores cadastrados, conforme pode ser vista na Figura 3.22. Logo abaixo existem alguns espaços para colocar a faixa da data de que se deseja efetuar a consulta. Ao clicar no botão "pesquisar" será chamado o arquivo ListaNormalizacao.jsp. Este arquivo é encarregado de gerar uma coleção de normalizações na faixa de data solicitada. Isto é feito usando um bean chamado Normalizacao.java que é entregue a classe Sistema através de uma classe de manipulação (NormalizacaoDAO.java).

Esta classe acessa no banco de dados as variáveis de medidas (através de Variaveis.java), os valores simulados (através de Limites.java) e a área das membranas (através de Dessalinizador.java). Estes valores são acessados do banco de dados e alimentam as equações contidas em Normalizacao.java, que em seguida envia uma resposta (os parâmetros normalizados) à classe Sistema.java, criando assim uma coleção de normalizações que é mostrada no browser através do arquivo TabelaNormalizacao.jsp solicitado por ListaNormalizacao.jsp.

O resultado da consulta é uma tabela de normalizações como é mostrado na Figura 3.23. A tabela mostra os valores dos parâmetros normalizados e a recuperação do sistema (r) na faixa de tempo solicitada.

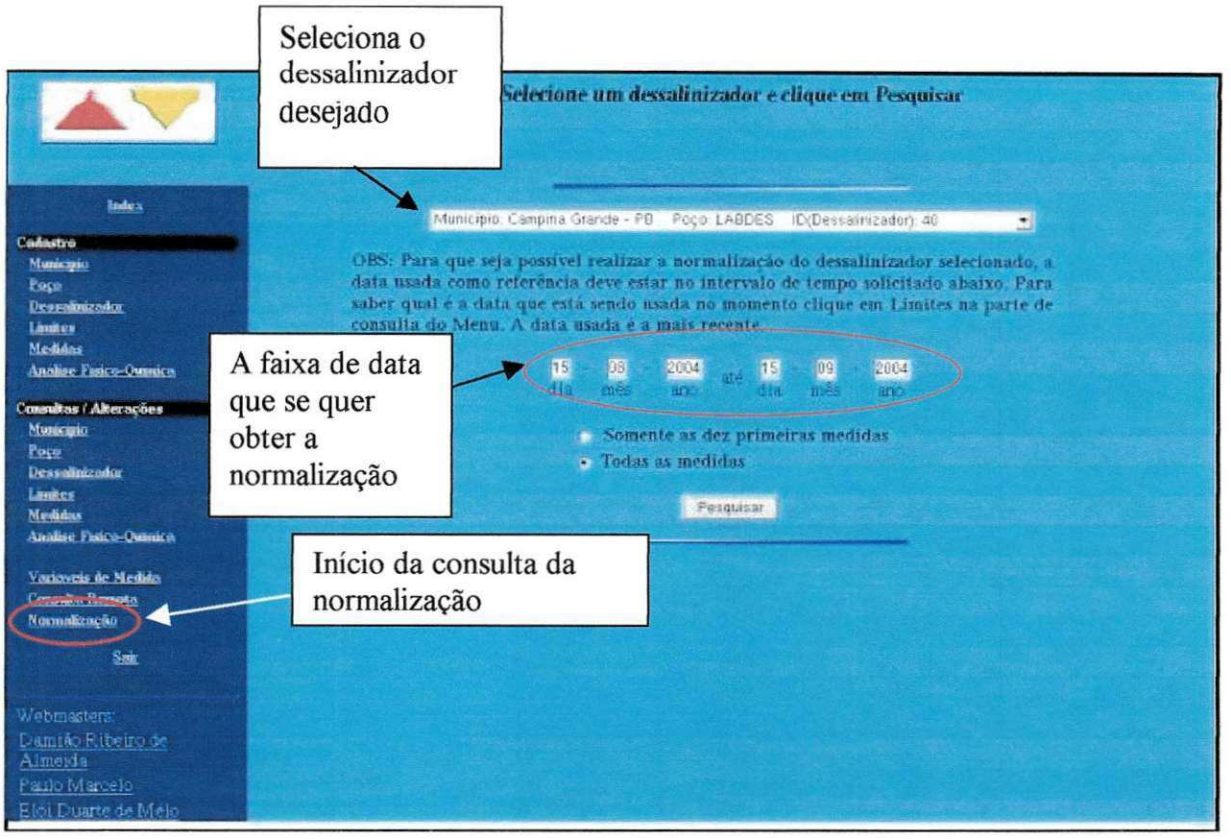


Figura 3.22: Consulta de normalização

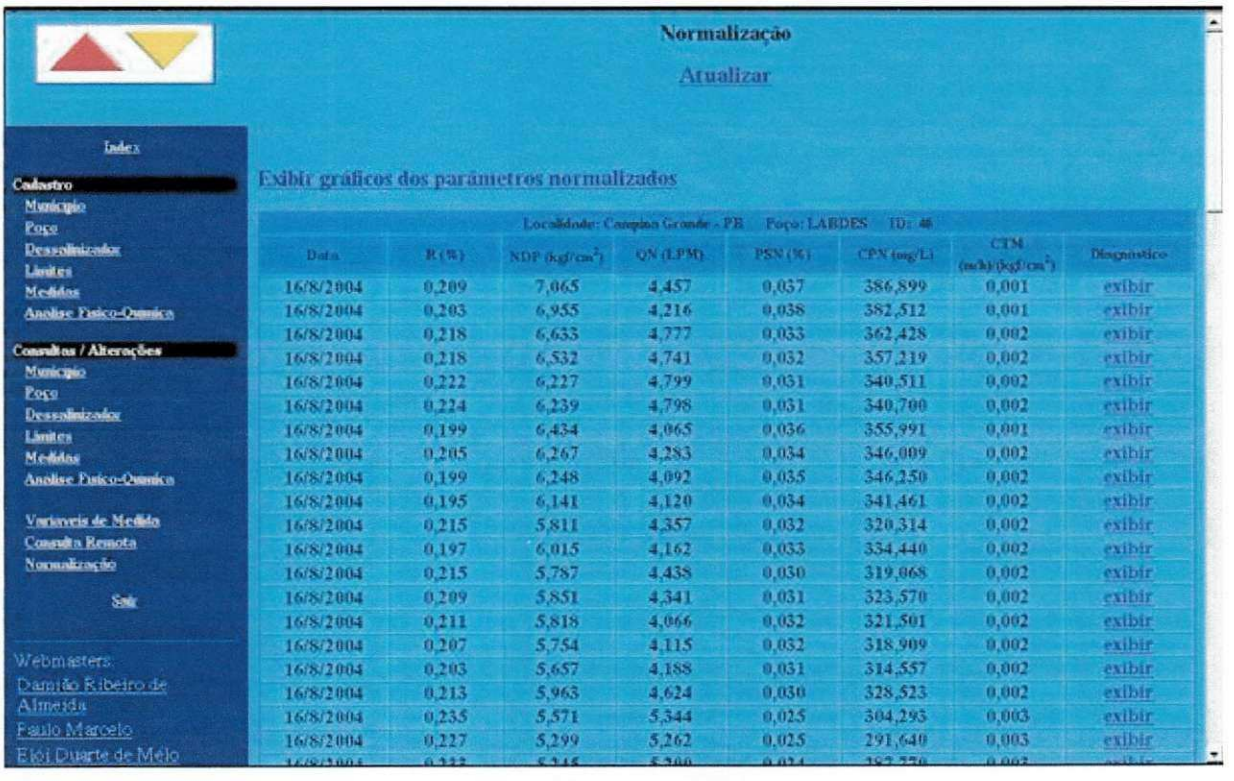


Figura 3.23: Tabela de normalizações

3.4.1. Gráficos dos parâmetros normalizados

Para exibir os gráficos dos parâmetros normalizados em função do tempo de operação do SDP, basta clicar em "Exibir gráficos dos parâmetros normalizados" na Figura 3.23. Ao fazer isso, será chamado o arquivo ExibeGrafico.jsp. Este arquivo apresenta uma relação dos gráficos que deve ser escolhido pelo usuário (Figura 3.24). Ao selecionar a opção desejada e clicar em "Pesquisar", será chamado o arquivo Grafico.jsp que tem a finalidade de construir o gráfico desejado pelo usuário usando o pacote jspChart.jsp. Nas Figuras 3.25 a 3.29, temos os gráficos dos parâmetros normalizados em função do tempo.

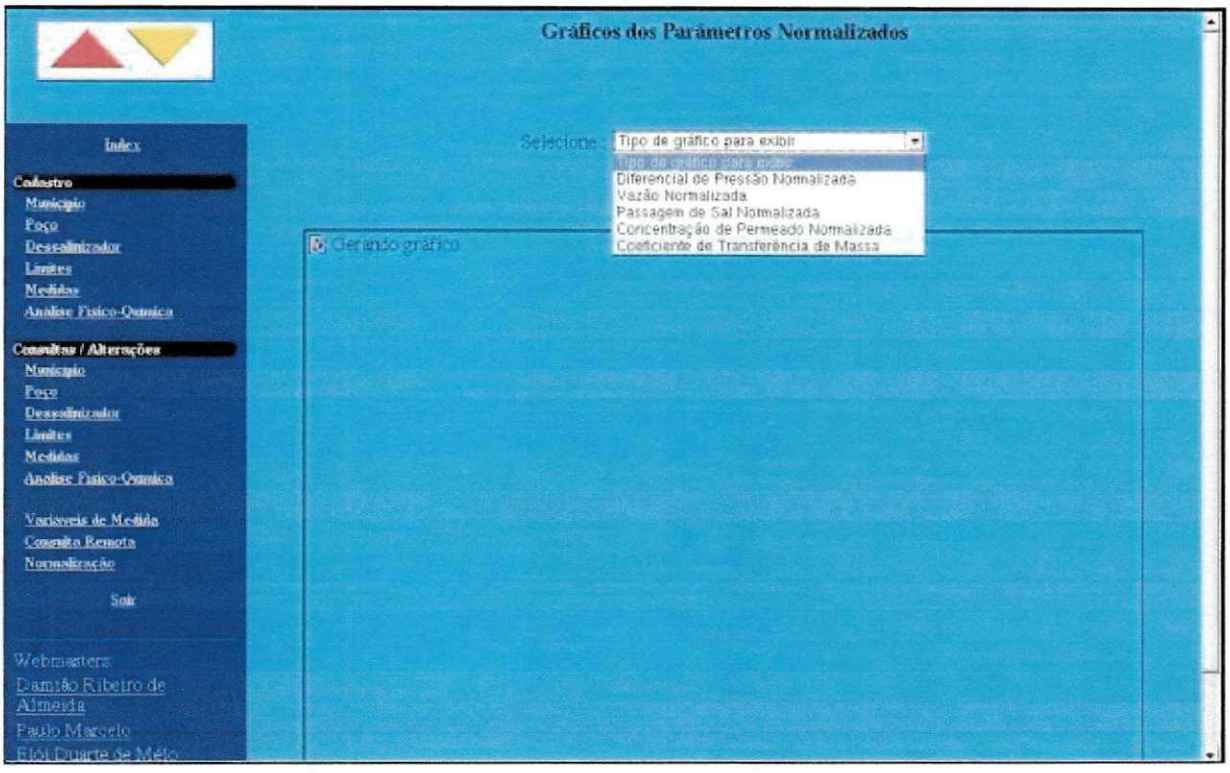


Figura 3.24: Seleção do tipo de gráfico dos parâmetros normalizados

Observando o comportamento do gradiente médio de pressão efetivo da membrana na Figura 3.25, verifica-se que este apresenta várias oscilações com uma leve tendência de crescimento a partir dos 1100 minutos (46 horas) de operação do SDP. O fato de haver estas oscilações deve-se ao aquecimento ocorrido durante o funcionamento do SDP, conforme mostra a Figura 3.20 do Anexo VI, tendo em vista que o processo era contínuo, ou seja, os efluentes permeado e concentrado eram retornados para o tanque de alimentação. Logo, com o aumento da temperatura ocorre uma diminuição do gradiente médio de pressão efetivo da membrana, conforme indica as Equações 14 e 15. Após o desligamento do dessalinizador

durante à noite ou ao meio dia, ocorria o esfriamento da solução de alimentação e, consequentemente, o aumento do NDP. A leve tendência de crescimento do NDP a partir dos 1100 minutos de operação indica um possivel inicio de formação de incrustacão nas membranas. No entanto, o aumento do NDP não foi o suficiente para chegar a proporcionar um diagnóstico de incrustação como será mostrado no ítem 3.4.2.

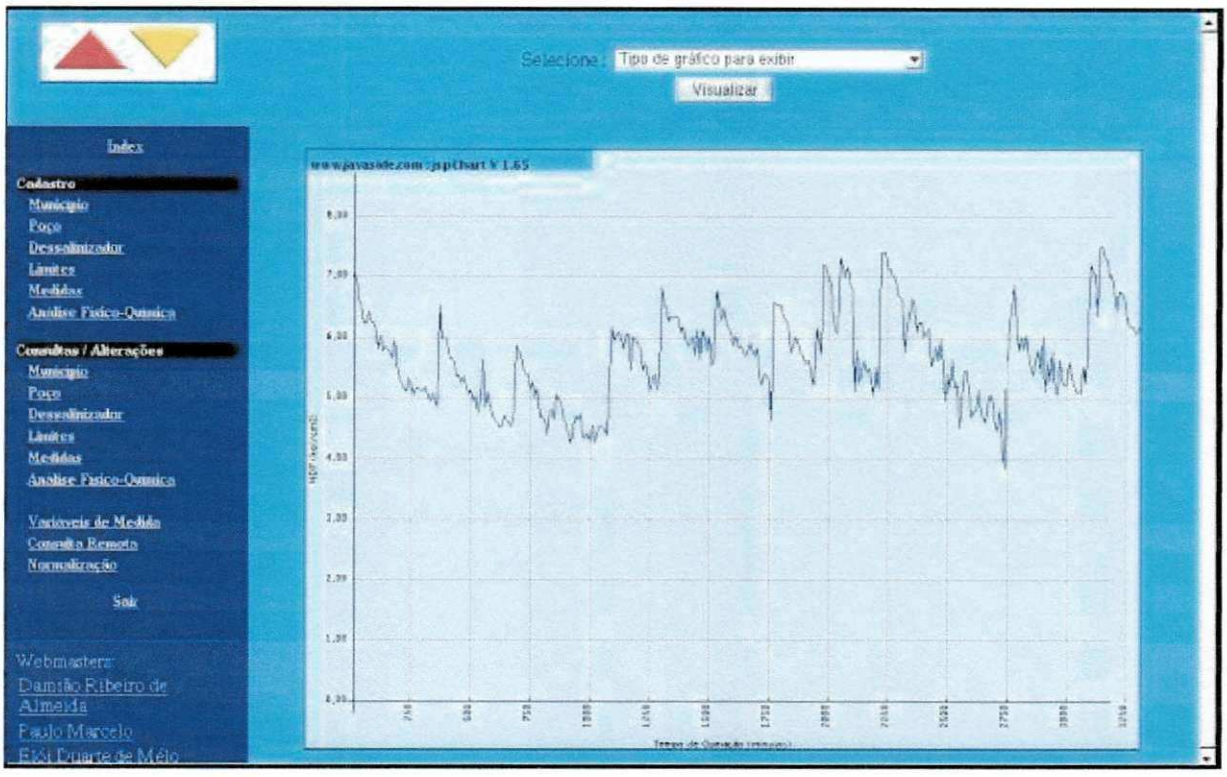


Figura 3.25: Gráfico do gradiente médio de pressão efetivo da membrana em função do tempo

A variação na temperatura da solução de alimentação também provocou oscilações no valor da vazão normalizada (Q_n), tendo em vista que com o aumento da temperatura, a membrana fica mais permeável e, com isso, a vazão do permeado aumenta. Os picos observados no valor de Q_n na Figura 3.26 correspondem justamente aos maiores aumentos de temperatura (Figura 3.20 do Anexo VI). Comparando-se o valor da vazão normalizada da partida do sistema (inicial) com os valores finais, observa-se que ocorreu uma queda desse parâmetro, indicando novamente uma pequena tendência a formação de incrustação nas membranas. Com a incrustação, há um aumento da resistência da membrana a passagem das moléculas da água e, consequentemente, ocorre uma diminuição na vazão do permeado.

A passagem de sal normalizada também é influenciada pela variação da temperatura, conforme mostra a Equação 20. De uma forma geral, observa-se na Figura 3.27 que há uma pequena tendência de queda da PSN com o tempo de operação do SDP, indo de encontro

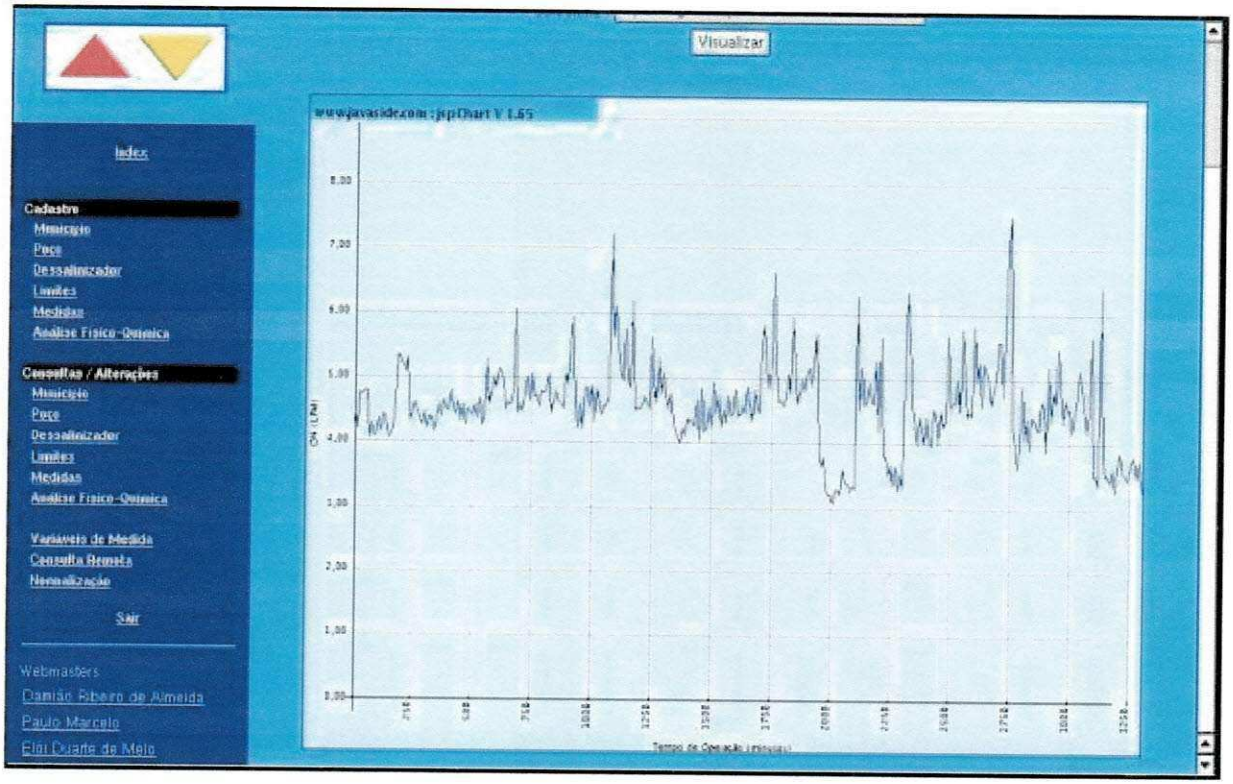


Figura 3.26: Gráfico da vazão normalizada em função do tempo

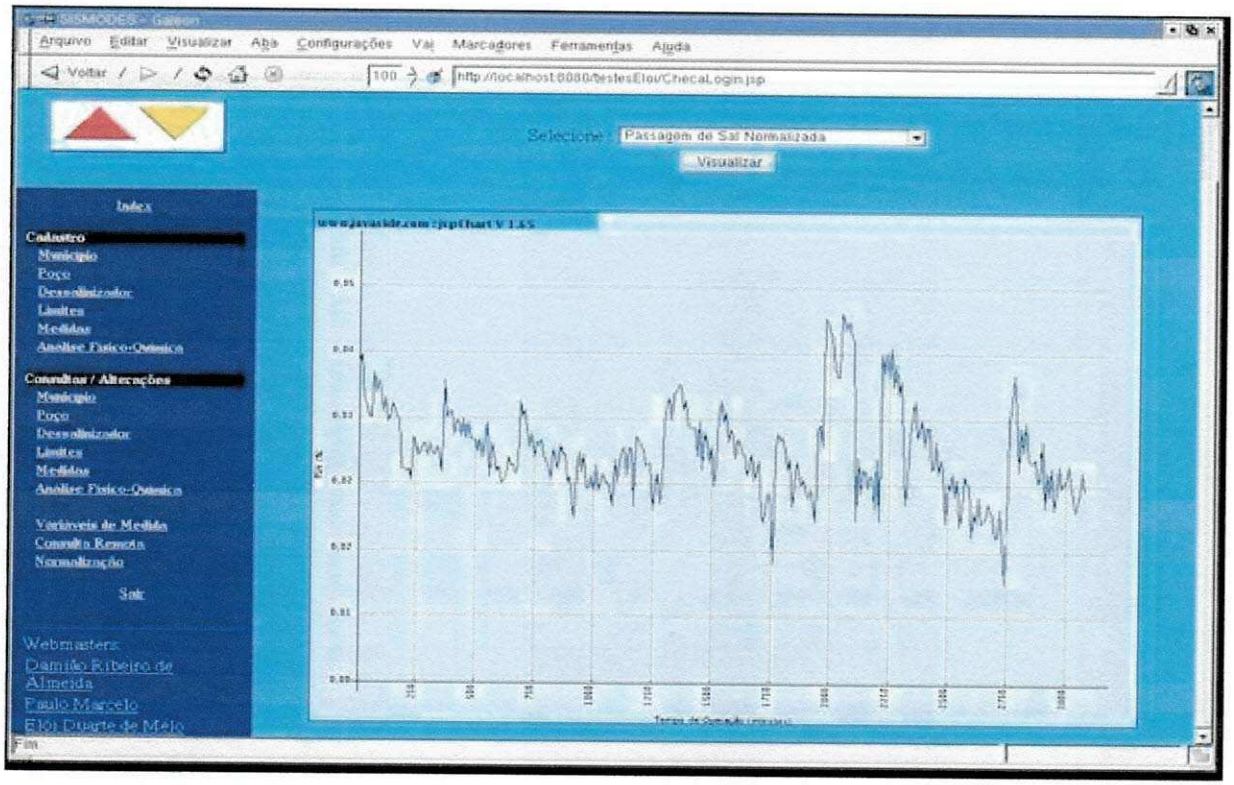


Figura 3.27: Gráfico da passagem de sal normalizada em função do tempo

aparentemente ao diagnóstico de incrustação inicial apresentado anteriormente. No entanto, segundo Huiting *et al.* (2001), no caso de incrustações por orgânicos e material biológico ("biofouling"), a passagem de sal normalizada não serve como parâmetro para se obter um diagnóstico. Portanto, pode-se supor que o princípio de incrustação pode estar sendo formado por um destes agentes. Segundo Huiting *et al.* (2001), os produtos químicos usados para prevenir o scaling tal como anti-incrustantes ou ácidos minerais apresentam potencial de biofouling. Como os efluentes do dessalinizador foram retornados ao processo e houve a adição de um anti-incrustante (Flocon 100) durante todo o período de funcionamento do SDP, pode ser que o aumento da concentração desse agente químico na solução de alimentação tenha iniciado um processo de incrustação por biofouling. No entanto, como veremos no ítem 3.4.2, a combinação da variação dos parâmetros normalizados não foi suficiente para se chegar a um diagnóstico preciso da formação de incrustação devido ao curto período de tempo de operação do SDP.

Como se pode observar na Figura 3.28, o aumento da temperatura provocou uma diminuição na concentração do permeado normalizada. De um modo geral, pode-se verificar também que a concentração de sais no permeado aumenta levemente em função do tempo (principalmente após os 1100 minutos). Esse aumento pode estar relacionado à formação inicial de incrustação.

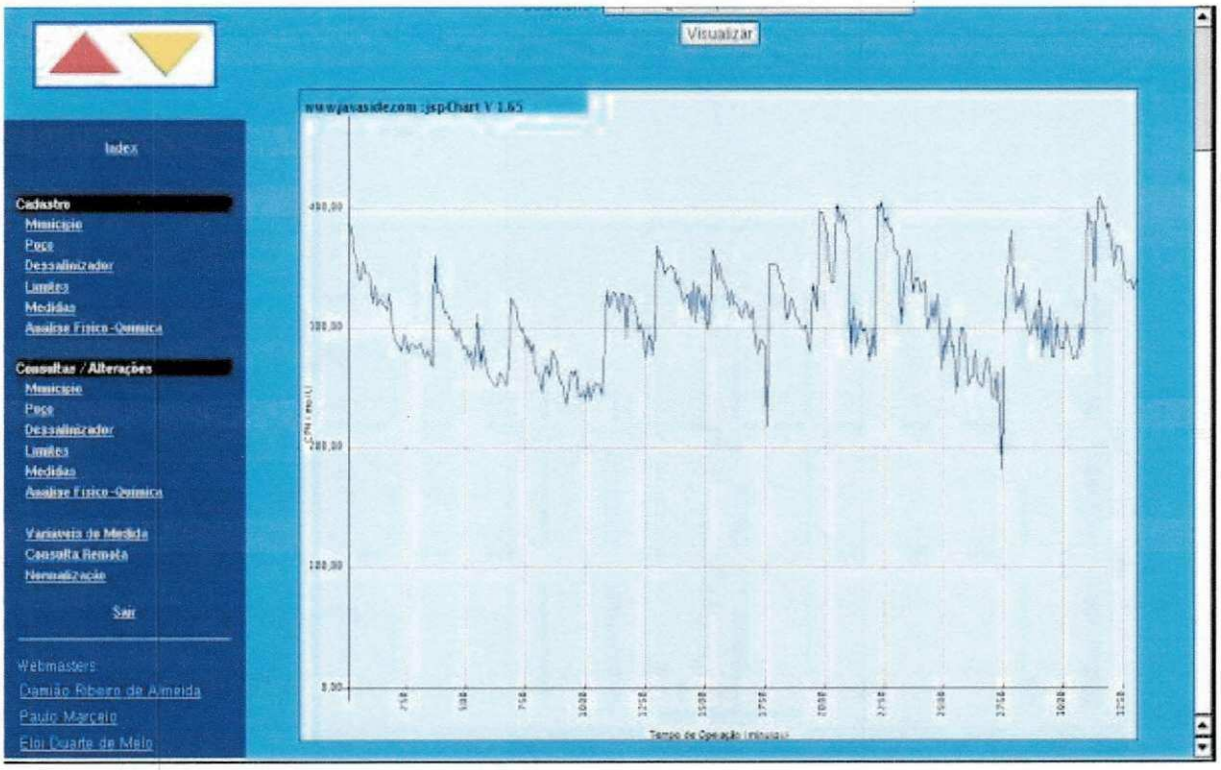


Figura 3.28: Gráfico da concentração do permeado normalizada em função do tempo

Outro fato é que a concentração do permeado normalizada apresenta valores bastante elevados (em torno de 300 mg/L). Isto se deve ao modelo proposto para a determinação da concentração do permeado (ítem 2.3.2.1) que não é muito adequado para a faixa de recuperação apresentada pelo sistema de dessalinização (19,5 a 23,5%). No entanto, para efeitos do diagnóstico de incrustação, o mais importante é a variação de um parâmetro e não o seu valor absoluto em si. De qualquer forma, este problema será imediatamente resolvido quando for instalado sensores de condutividade elétrica na alimentação do SDP e no efluente do permeado e/ou do concentrado.

O aumento da temperatura provocou também picos de crescimentos no coeficiente de transferência de massa, pois a vazão do permeado é maior para temperaturas mais elevadas. No entanto, de um modo geral, não se observou um "crescimento real" do CTM no decorrer do tempo de operação. Quando a incrustação ocorre há uma queda no CTM (van de Lisdonk *et al.*, 2000), fato este não observado durante o pequeno período de tempo em que o SDP operou.

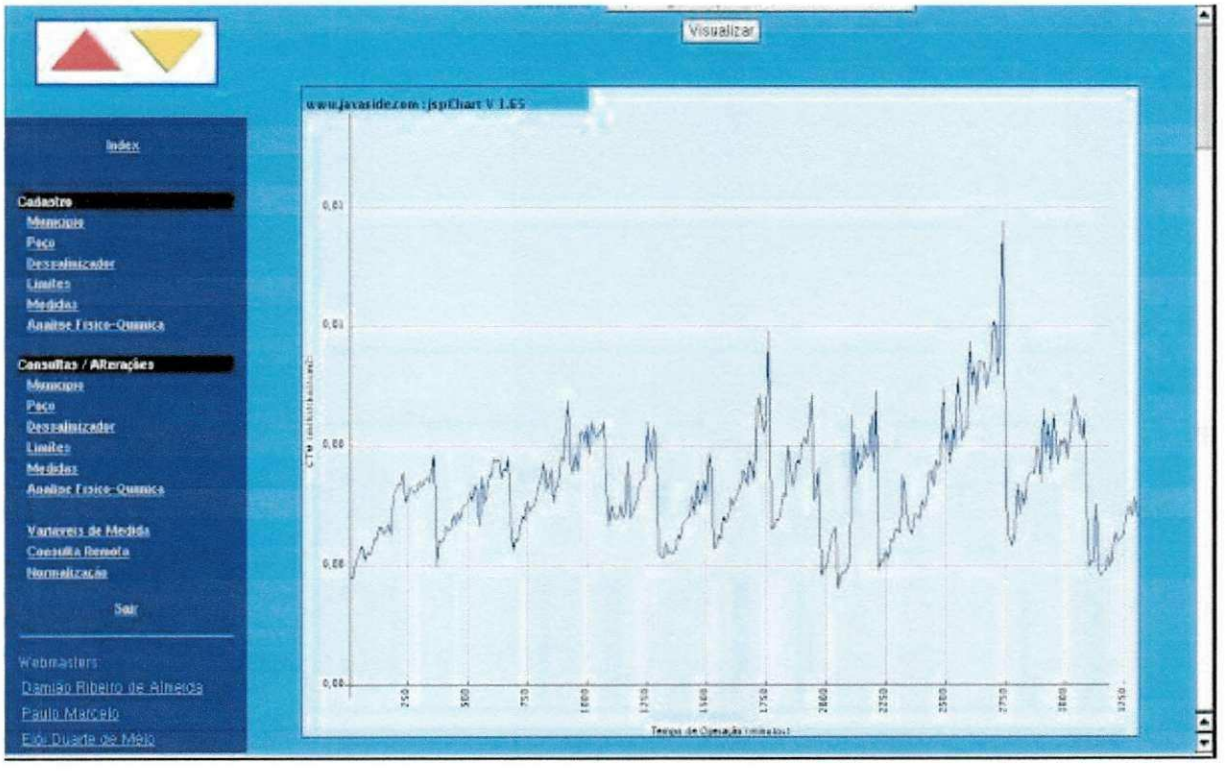


Figura 3.29: Gráfico do coeficiente de transferência de massa em função do tempo

3.4.2. Diagnóstico do SDP em função dos parâmetros normalizados

Para exibir um diagnóstico de incrustação em função dos parâmetros normalizados, basta clicar em "exibir" na coluna "Diagnóstico" na tabela de normalizações (Figura 3.23). Ao fazer isso, será chamado o arquivo Diagnostico.jsp. Este arquivo é responsável por imprimir um diagnóstico de acordo com as condições da classe Normalizacao.java (ítem 2.3.2.6) e também a variação de cada parâmetro em relação ao valor de partida na forma de uma tabela (Figura 3.30).

Conforme comentado anteriormente, a variação dos parâmetros normalizados em relação aos valores de partida do sistema não indicou um diagnóstico de incrustação preciso, tendo em vista que o tempo de operação do SDP no presente estudo foi curto. Como se pode observar na Figura 3.30, o diagnóstico obtido no final do monitoramento (3250 minutos) foi o seguinte: "No momento não está havendo potencial de incrustação".

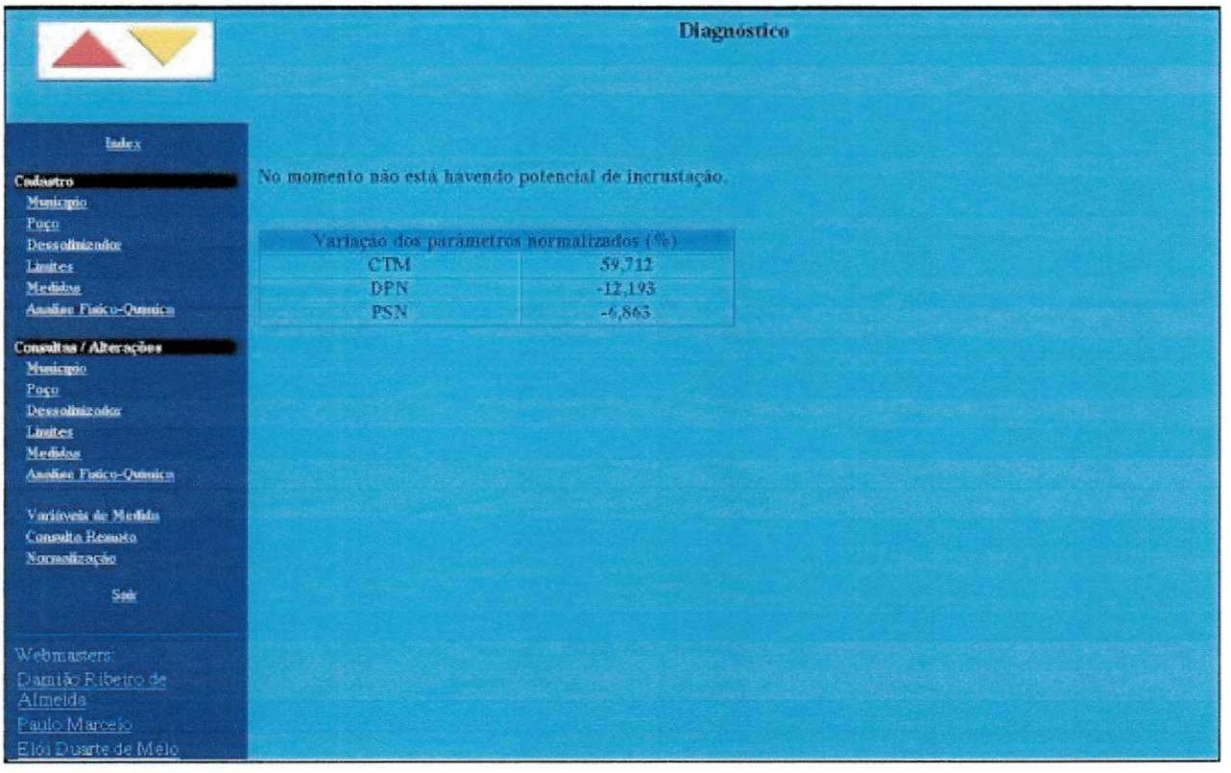


Figura 3.30: Diagnóstico do SDP com 3250 minutos (54 horas) de operação

Este diagnóstico foi dado pelo software porque, como se pode observar na figura acima, nehuma das condições do ítem 2.3.2.6 foi satisfeita. O coeficiente de transferência de massa (CTM) teve um aumento de 59,71% neste instante em relação ao valor da partida. Na realidade, o CTM deve apresentar um valor inferior ao da partida do sistema quando ocorre a

formação de incrustação. O fato do CTM estar com um valor acima neste instante deve-se ao aumento da temperatura. Verificou-se que quando a solução de alimentação esfria (durante a noite), o CTM apresenta um valor igual ao da partida do sistema durante todo o período de testes. No caso do gradiente médio de pressão efetivo da membrana (NDP) verifica-se que este teve uma diminuição de 12,19% neste instante em relação ao valor da partida. Ao contrário do CTM, o NDP deve apresentar um valor superior ao da partida do sistema quando ocorre a formação de incrustação. Contudo, observou-se que este parâmetro apresentou uma leve tendência de crescimento com o tempo de operação. Além disso, como comentado anteriormente, a temperatura também teve forte influência nesse parâmetro.

A Figura 3.31 comprova a importância de se realizar um monitoramento baseado na normalização do sistema e não apenas na simples variação das variáveis de suas medidas. Conforme se pode verificar, há a ocorrência de vários alarmes vermelhos, devido a variação da pressão, indicando claramente a presença de incrustação. No entanto, como foi observado com os parâmetros normalizados, estas variações não ocorrem devido a presença de incrustações, mas sim devido a temperatura. Ao longo de todo monitoramento, nenhum diagnóstico de incrustação foi indicado pela combinação dos parâmetros normalizados, comprovando desse modo a eficiência dessa metodologia para se monitorar verdadeiramente um sistema de dessalinização.

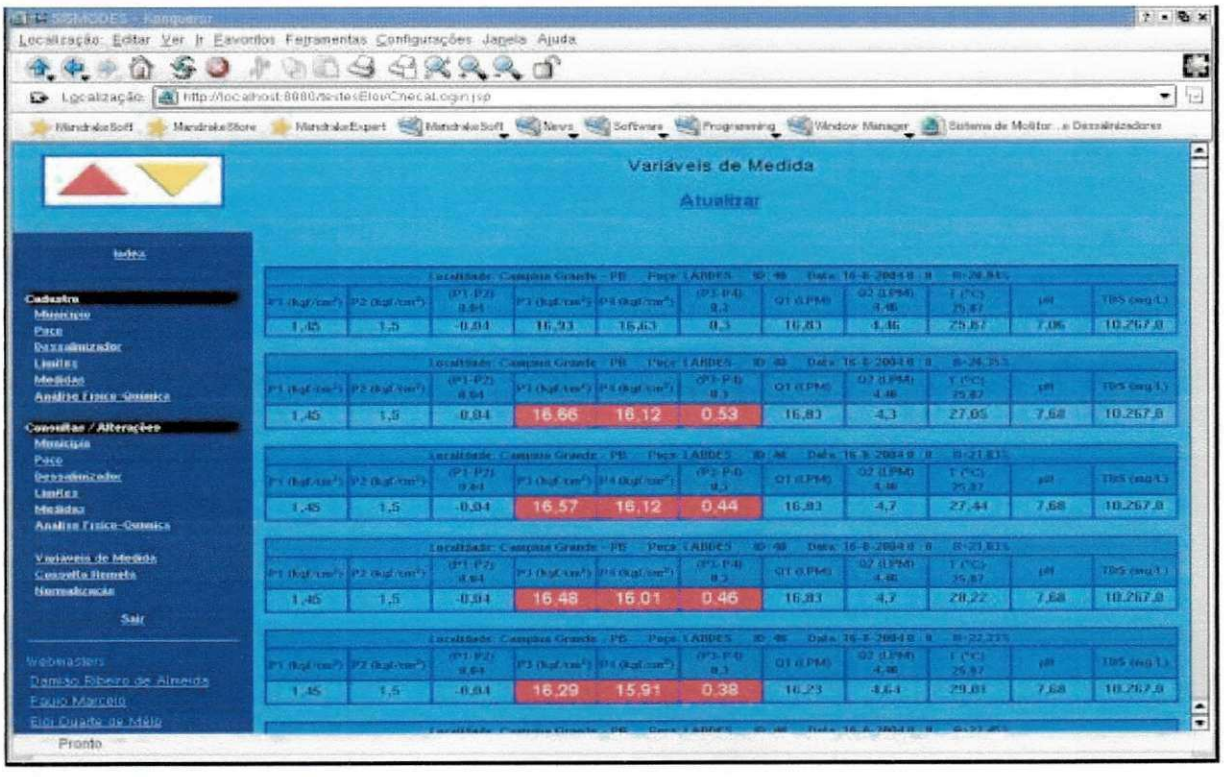


Figura 3.31: Diagnóstico obtido baseado simplesmente na variação das variáveis de medidas

Para se fazer uma demonstração, a Figura 3.32 apresenta um exemplo de um diagnóstico de incrustação dado pelo software para um sitema de dessalinização localizado na localidade de Pedra D'Água da cidade de Caturité – PB. Os dados das variáveis de medidas referentes a um monitoramento realizado de setembro de 1999 a outubro de 2000, que encontram-se no Anexo VII, foram todos digitados manualmente pelo usuário.

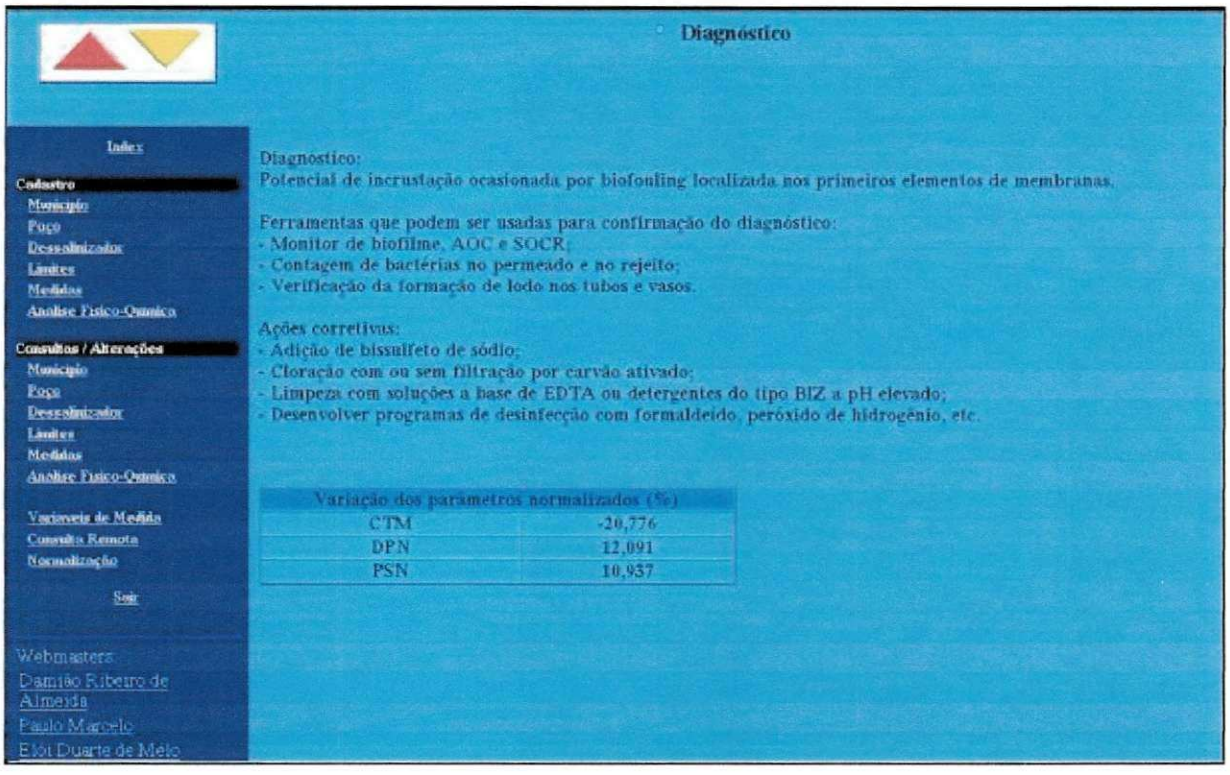


Figura 3.32: Diagnóstico obtido com os dados de um sistema instalado em Caturité – PB

Como se pode observar na figura acima, o software além de indicar o tipo e a localização da incrustação no sistema, indica uma série de ferramentas experimentais que podem ser usadas para se confirmar o diagnóstico, bem como aponta as ações corretivas que devem ser tomadas pelos operadores / técnicos para acabar ou inibir a incrutação.

CAPÍTULO IV

Conclusões

Em termos gerais, o protótipo desenvolvido possibilita o gerenciamento remoto do sistema de dessalinização piloto (SDP) com as opções de cadastro, consulta, alteração e exclusão de município, poço, dessalinizador, análise fisico-química, variáveis e limites, além de consultar e desligar alarmes acionados. Existe também uma opção para se conectar imediatamente ao dessalinizador desejado para verificar as suas medidas nesse mesmo instante (consulta remota).

Além de realizar um gerenciamento remoto do sistema de dessalinização piloto (SDP), o software desenvolvido faz uma normalização do mesmo numa faixa de tempo de operação desejada. Para tanto, é necessário antes se certificar que todos os dados correspondentes à faixa de tempo solicitada foram cadastrados pelo usuário (P1, P2, pH e TDS) e enviados pelo computador embarcado (P3, P4, Q1, Q2 e T). Deve-se também ter o cuidado de cadastrar os valores simulados dos limites de operação deste sistema na opção cadastrar limites do menu, bem como a área das membranas na opção cadastrar dessalinizador.

Além disso, SISMODES permite a exibição de gráficos que mostram o comportamento das variáveis de medidas, bem como dos parâmetros normalizados, em função do tempo de operação do SDP. Com isso, o usuário tem mais condições de perceber as alterações ocorridas nesses parâmetros durante o processo de dessalinização. Para fazer este recurso, o software utiliza o pacote free jspChart.jsp.

Analisando os gráficos dos parâmetros normalizados, por exemplo, percebe-se que estes apresentam várias oscilações provocadas pelas variações da temperatura, como mostra a Figura 3.20 do Anexo VI. No entanto, não foi possível identificar de forma precisa a presença de alterações provocadas por algum tipo de incrustação pela análise dos gráficos.

Para facilitar ainda mais a vida do usuário, o software é capaz de exibir um diagnóstico de incrustação em função dos parâmetros normalizados. Este recurso de SISMODES confirmou o que foi comentado com a análise dos gráficos, ou seja, a variação dos parâmetros normalizados em relação aos valores de partida do sistema indicou a inexistência de incrustação, pois o diagnóstico obtido no final do monitoramento (3250 minutos) foi o seguinte: "No momento não está havendo potencial de incrustação". Este diagnóstico foi dado por SISMODES porque, neste instante, as condições de variação dos parâmetros normalizados em relação ao valor da partida do sistema não foram satisfeitas: o

coeficiente de transferência de massa (CTM) teve um aumento de 59,71% e o gradiente médio de pressão efetivo da membrana (NDP) diminuiu 12,19%.

O monitoramento baseado na normalização do sistema se mostrou mais eficiente do que a simples análise da variação das variáveis de medidas, tendo em vista houve a ocorrência de vários alarmes vermelhos, devido a variação da pressão, indicando claramente a presença de incrustação. No entanto, como foi observado com os parâmetros normalizados, estas variações não ocorrem devido a presença de incrustações, mas sim devido a temperatura. Ao longo de todo monitoramento, nenhum diagnóstico de incrustação foi indicado pela combinação dos parâmetros normalizados, comprovando desse modo a eficiência dessa metodologia para se monitorar verdadeiramente um sistema de dessalinização.

Para finalizar, além de indicar o tipo e a localização da incrustação no sistema, SISMODES indica uma série de ferramentas experimentais que podem ser usadas para se confirmar o diagnóstico, bem como aponta as ações corretivas que devem ser tomadas pelos operadores / técnicos para acabar ou inibir a incrutação.

Isto é de suma importância, pois permite que o usuário faça uma interpretação dos dados normalizados em qualquer lugar do mundo em tempo de operação, possibilitando assim um diagnóstico mais rápido e preciso do que está ocorrendo com o sistema de dessalinização. Com isso, é possível elaborar estratégias de manutenção preventiva visando redução de custos.

Além disso, a utilização de free softwares como Linux, tomcat e java minimizam o custo do projeto. As vantagens desses softwares não são apenas o custo, mas também a qualidade, confiabilidade e a popularidade no mundo da informática.

CAPÍTULO V

Sugestões para trabalhos futuros

Para melhorar a monitoração remota do sistema de dessalinização piloto e como conseqüência o diagnóstico dos tipos de incrustações, bem como a sua localização no sistema, aumentando assim a confiança nos resultados obtidos com os parâmetros normalizados, o presente trabalho sugere que as pesquisas a serem realizadas desenvolvam os seguintes pontos:

1. Instalação de sensores de condutividade da alimentação do 1º arranjo com o objetivo de se obter a concentração de sais da alimentação em tempo de operação;
2. Instalação de sensores de condutividade na alimentação do 2º arranjo de membranas, no produto do 1º arranjo e no concentrado;
3. Instalação de sensores de vazão na alimentação do 2º arranjo, no produto do 1º e 2º arranjos e no concentrado;
4. Instalação de sensores de pressão na alimentação do 2º arranjo e no produto total;
5. Desenvolvimento e utilização de ferramentas apropriadas para cada tipo de incrustação para confirmar o diagnóstico previsto pelo "software" através dos parâmetros normalizados;
6. Incrementar ao "software" existente o cadastro e a consulta das análises físico-químicas do permeado e do concentrado, possibilidando também os cálculos do ISL, dos potenciais de incrustação por sulfato de cálcio e sílica deste último efluente;
7. Utilizar um pacote gráfico mais robusto e de melhor qualidade visual.

Referências Bibliográficas

ALMEIDA, D. R. *Desenvolvimento do Módulo de Gerência Distribuída de Dessalinizadores com Capacidade de Monitoração Remota.* Relatório de Iniciação Científica PIBIC/CNPq, Universidade Federal da Paraíba, Campina Grande, 2002.

AMJAD, Z. *Reverse Osmosis - Membrane Technology, Water Chemistry and Industrial Applications.* Van Nostrand - Reinhold, New York, 1992.

ARGONAVIS. Texto retirado de um curso de java localizado no seguinte site: http://www.argonavis.com.br/cursos/java/j550/j550_9.pdf em 2003.

BENNETT, P. B., *Scale and deposit control for reverse osmosis systems,* Membrane Technology Conference Proceedings, AWWA, New Orlens, LA, pp. 691 – 693, 1996.

BRADLEY, R. *Design considerations for reverse osmosis systems,* In Zahid Amjad (Ed.), Reverse Osmosis - Membrane Technology, Water Chemistry and Industrial Applications, Van Nostrand-Reinhold, New York, 1992.

BRANDT, D. C.; LEITNER, G. F.; LEITNER, W. E. *Reverse osmosis membrane states of íhe art.* In Zailid Amjad (Ed.), Reverse Osmosis - Membrane Technology, Water Chemistry and Industrial Applications, Van Nosírand-Reinhold, New York, 1992.

DATE, C. J. *Uma introdução aos sistemas de bancos de dados.* São Paulo: Editora Edgard Blucher, 1998.

EL-MANHARAWY, S.; HAFEZ, A. *Molar ratios as a useful tool for predictions of scaling potencial incide RO systems.* Desalination, vol. 136, pp. 243-254, 2001.

FILMTEC, *Manual Técnico.* 1995.

HABERT, A. C.; BORGES, C. P.; NÓBREGA, R. *Processos de separação com membranas.* Escola Piloto em Engenharia Química, COPPE/UFRJ - Programa de Engenharia Química. 1997.

HORSTMANN, C. S.; CORNELL, G. *Core Java 2 volume I – Fundamentos.* São Paulo: Editora Makron Books Ltda, 2001a.

HORSTMANN, C. S.; CORNELL, G. *Core Java 2 volume II – Recursos avançados.* São Paulo: Editora Makron Books Ltda, 2001b.

HUITING, H.; KAPPELHOF, J. W. N. M.; BOSKLOPPER, Th. G. J. *Operation of NF/RO plant: from reactive to proactive.* Desalination, vol. 139, pp. 183-189, 2001.

HYDRANAUTICS. Membrane System Design Software, RO System Design. Version 800©, 2002.

JACQUES SAWVÉ. Texto retirado do conteúdo da disciplina *Desenvolvimento de Aplicações Corporativas Avançadas,* oferecida pelo Departamento de Sistemas e Computação da Universidade Federal de Campina Grande no ano de 2002 localizado no seguinte site: http://www.dsc.ufcg.edu.br/~jacques/cursos/j2ee/html/jsp/livros.htm.

JOYCE, A.; LOUREIRO, D.; RODRIGUES, C.; CASTRO, *S. Small reverse osmosis units Pv systems for water purification in rural places.* Desalination, vol. 137, pp. 39-44, 2001.

KERR, T. J.; McHALE. B. B. *Applications in general microbiology: A laboratory manual.* 6th ed., Hunter Textbooks Inc., Winston-Salem, 2001.

KIM, C. G.; YOON, T. I.; LEE, M. J. *Characterization and control of foulants occurring from RO disc-tube-type, membrane treating, fluorine manufacturing, process wastewater.* Desalination, vol. 151, pp. 283-292, 2002.

MALLEVIALLE, L.; ODENDAAL, P. E.; WIESNER, M. R. *Water Treatment Membrane Processes.* American Water Works Association Research Foundation; Lyonnaise des Eaux; Water Research Comission of South Africa. McGraw-Hill, Washington, DC, USA, 1996.

MINDLER, A. B.; EPSTEIN, A. C. *Measurements and control in reverse osmosis desalinations.* Desalination, vol. 59, pp. 343-379, 1986.

NING, R. Y.; NETWIG, J. P. *Complete elimination of acid injection in reverse osmosis plants.* Desalination, vol. 143, pp. 29-34, 2002.

OZAKI, H.; LI, H. *Rejection of organic compounds by ultra-low pressure reverse osmosis membrane.* Water Research. vol. 36, pp. 123-130, 2002.

SCHNEIDER, R. P.; TSUTIYA, M. T. *Membranas filtrantes para o tratamento de água, esgoto e água de reuso.* ABES, 1ª ed., São Paulo, 2001.

SHEIKHOLESLAMI, R.; BRIGHT, J. *Sílica and metals remova/ by pretreatment to prevent fouling of reverse osmosis membranes.* Desalination, vol. 143, pp. 255-267, 2002.

SOUSA, S. E. H. *Monitoramento e desempenho do sistema de dessalinização via osmose inversa da Cia. De Tecidos Norte de Minas-COTEMINAS-CG.* Relatório de defesa de estágio do curso de Engenharia Química da UFPB, 2001.

SOUSA, S. E. H. *Normalização de Sistemas de Dessalinização via Osmose Inversa.* Dissertação de Mestrado em Engenharia Química, UFCG, Campina Grande-PB, 2003.

SRIDHAR, S.; KALE, A.; KHAN, A. A. *Reverse osmosis of edble vegetable oil industry effluent.* Journal of Membrane Science, vol. 205, pp. 83-90, 2002.

STUMM, W.; MORGAN, J. J. *Aquatic Chemistry.* Wiley-Interscience, New York, 1996.

TAYLOR, J. P.; JACOBS.E. P. *Reverse osmosis and nanofilfration.* In Joel Mallevialle *et al.* (eds), water Treatment membrane process. Mc Graw-HilL New York, 1996.

VALLEY, G. *Membrane Process Optimization Technology.* Desalination and Water Purification Research and Development Program Report Nº 100, Water Treatment Engineering and Research Group. PerLorica Inc., Califórnia, USA, 2003.

VAN DE LISDONK, C. A. C.; VAN PAASSEN J. A. M.; SCHIPPERS, J. C. *Monitoring scaling in nanofiltration and reverse osmosis membrane systems.* Desalination, vol. 132, pp. 101-108, 2000.

VAN DER HOEK, J. P.; BONNÉ, P. A. C.; VAN SOEST E. A. M.; GRAVELAND, A. 21st IWSA Congress, Madrid, ss1, pp. 11 – 16, 1997.

VROUWENVELDER, J. S.; MANOLARAKIS, S. A., VEENENDAAL, H. R.; KOOIJ, D. Van der. *Biofouling potential of chemicals used for scale control in RO and NF membranes.* Desalination, vol. 132, pp. 1-10, 2000.

VROUWENVELDER, J. S.; KOOIJ, D. Van der. *Diagnosis, prediction and prevention of biofouling of NF and RO membranes.* Desalination, vol. 139, pp. 65-71, 2001.

VROUWENVELDER, J. S.; KOOIJ, D. Van der. *Diagnosis of fouling problems of NF and RO membrane installations by a quick scan.* Desalination, vol. 153, pp. 121-124, 2002.

VROUWENVELDER, J. S.; KAPPELHOF, J.W.N.M.; HEIJMAN, S. G. J..; SCHIPPERS, J. C.; KOOIJ, D. Van der. *Tools for fouling diagnosis of NF and RO membranes and assessment of the fouling potential of feed water.* Desalination, vol. 157, pp. 361 - 365, 2003.

WIESNER, M. R.; APTEL, P. *Mass transport and permeate flux and fouling in pressure-driven processes.* In: Water Treatment Membrane Processes (MALLEVIALLE, J.; ODENDAAL, P. E.; WIESNER, M. R., eds), pp. 4.1-4.30. McGraw-Hill, New York, 1996.

WMLCLUB. Texto retirado de um artigo sobre JSP localizado no seguinte site: http://br.wmlclub.com/articulos/index.htm em 2003.

ANEXO I

Documentação da classe Normalizacao.java

sismodes.beans

Class Normalizacao

```
java.lang.Object
  |
  +--sismodes.beans.Normalizacao
```

---

public class **Normalizacao**
extends java.lang.Object
implements java.io.Serializable

Um bean que implementa as equações do processo de osmose inversa com o objetivo de fazer a normalização do sistema

See Also:
Serialized Form

Field Summary

static java.lang.String	BIOFOULING
static java.lang.String	COLOIDES_SILICA
static java.lang.String	MENSAGEM_OK
static int	NUMERO_BIOFOULING
static int	NUMERO_COLOIDES_SILICA
static int	NUMERO_OK
static int	NUMERO_ORGANICOS
static int	NUMERO_OXIDOS_METALICOS
static int	NUMERO_SCALING
static java.lang.String	ORGANICOS

static java.lang.String	OXIDOS_METALICOS
static java.lang.String	SCALING

Constructor Summary

Normalizacao()
Normalizacao(sismodes.beans.Variaveis v, sismodes.beans.Limites l, sismodes.beans.Dessalinizador d)

Method Summary

boolean	equals(Normalizacao n) Verifica a igualdade entre dois parâmetros
static java.lang.String	formataValor(float valor) Retorna o valor de um parâmetro normalizado formatado tal como o usuário visualisa no browser
int	getAnoOperacao() Formato: Gravando no banco de dados: ano - mes - dia hora : min Consultando o banco de dados: dia - mes - ano hora : min
int	getAnoReferencia()
float	getConcentracaoAS() Retorna a concentração da alimentação simulada
float	getConcentracaoC() Retorna o total de sólidos do concentrado
float	getConcentracaoCS() Retorna o total de sólidos do concentrado simulado
float	getConcentracaoP() Retorna o total de sólidos do permeado
float	getConcentracaoPermeadoN() Retorna a Concentração do permeado normalizada
float	getConcentracaoPS() Retorna o total de sólidos do permeado simulado
java.lang.String	getDataOperacao()
sismodes.beans.Dessalinizador	getDessalinizador()
java.lang.String	getDiagnostico() Retorna o diagnóstico de incrustação, informando a sua localização no equipamento e o seu tipo.

int	getDiaOperacao() Formato: Gravando no banco de dados: ano - mes - dia hora : min Consultando o banco de dados: dia - mes - ano hora : min
int	getDiaReferencia() Formato: Gravando no banco de dados: ano - mes - dia hora : min Consultando o banco de dados: dia - mes - ano hora : min
float	getDiferencialPressaoN() Retorna o gradiente médio de pressão efetivo da membrana (NDP)
float	getDiferencialPressaoNS() Retorna o gradiente médio de pressão efetivo da membrana (NDP) simulado
float	getFatorConcentracao() Retorna o fator de concentração média logaritmica
float	getFatorConcentracaoS() Retorna o fator de concentração média logarítmica simulado
float	getFatorTemperatura() Retorna o fator de correção de temperatura
float	getFatorTemperaturaS() Retorna o fator de correção de temperatura simulado
sismodes.beans.Limites	getLimites()
int	getMesOperacao() Formato: Gravando no banco de dados: ano - mes - dia hora : min Consultando o banco de dados: dia - mes - ano hora : min
int	getMesReferencia()
float	getPassagemSalN() Retorna a passagem de sal normalizada
float	getPassagemSalNS() Retorna a passagem de sal normalizada simulada
float	getPressaoOsmoticaA() Retorna a pressão osmótica da alimentação
float	getPressaoOsmoticaAS() Retorna a pressão osmótica da alimentação simulada
float	getPressaoOsmoticaC() Retorna a pressão osmótica do concentrado
float	getPressaoOsmoticaCS() Retorna a pressão osmótica do concentrado simulada
float	getPressaoOsmoticaP() Retorna a pressão osmótica do permeado

float	**getPressaoOsmoticaPS**() Retorna a pressão osmótica do permeado simulada
float	**getRecuperacao**() Retorna a recuperação do sistema
float	**getRecuperacaoS**() Retorna a recuperação de projeto (simulada) do sistema
int	**getTempoOperacao**()
float	**getTransferenciaMassaN**() Retorna o coeficiente de transferência de massa normalizada
float	**getTransferenciaMassaNS**() Retorna o coeficiente de transferência de massa normalizada simulado
float	**getVariacaoCTM**()
float	**getVariacaoDPN**()
float	**getVariacaoPSN**()
sismodes.beans.Variaveis	**getVariaveis**()
float	**getVazaoN**() Retorna a vazão do permeado normalizada
void	**setConcentracaoPermeadoN**(float cpn)
void	**setDessalinizador** (sismodes.beans.Dessalinizador dessalinizador)
void	**setDiferencialPressaoN**(float dpn)
void	**setDiferencialPressaoNS**(float dpns)
void	**setLimites**(sismodes.beans.Limites limites)
void	**setPassagemSalN**(float psn)
void	**setTransferenciaMassaN**(float ctmn)
void	**setVariaveis**(sismodes.beans.Variaveis variaveis)
void	**setVazaoN**(float qn)

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

NUMERO_OK

public static final int **NUMERO_OK**

NUMERO_SCALING

public static final int **NUMERO_SCALING**

NUMERO_OXIDOS_METALICOS

public static final int **NUMERO_OXIDOS_METALICOS**

NUMERO_COLOIDES_SILICA

public static final int **NUMERO_COLOIDES_SILICA**

NUMERO_ORGANICOS

public static final int **NUMERO_ORGANICOS**

NUMERO_BIOFOULING

public static final int **NUMERO_BIOFOULING**

MENSAGEM_OK

public static final java.lang.String **MENSAGEM_OK**

SCALING

public static final java.lang.String **SCALING**

OXIDOS_METALICOS

```
public static final java.lang.String OXIDOS_METALICOS
```

COLOIDES_SILICA

```
public static final java.lang.String COLOIDES_SILICA
```

ORGANICOS

```
public static final java.lang.String ORGANICOS
```

BIOFOULING

```
public static final java.lang.String BIOFOULING
```

Constructor Detail

Normalizacao

```
public Normalizacao()
```

Normalizacao

```
public Normalizacao(sismodes.beans.Variaveis v,
                    sismodes.beans.Limites l,
                    sismodes.beans.Dessalinizador d)
```

Method Detail

getVariaveis

```
public sismodes.beans.Variaveis getVariaveis()
```

---

setVariaveis

```
public void setVariaveis(sismodes.beans.Variaveis variaveis)
```

---

getLimites

```
public sismodes.beans.Limites getLimites()
```

---

setLimites

```
public void setLimites(sismodes.beans.Limites limites)
```

---

getDessalinizador

```
public sismodes.beans.Dessalinizador getDessalinizador()
```

---

setDessalinizador

```
public void setDessalinizador(sismodes.beans.Dessalinizador dessalinizador)
```

---

getDataOperacao

```
public java.lang.String getDataOperacao()
```

---

getDiaReferencia

```
public int getDiaReferencia()
```

Formato:
Gravando no banco de dados: ano - mes - dia hora : min
Consultando o banco de dados: dia - mes - ano hora : min

---

getMesReferencia

```
public int getMesReferencia()
```

---

getAnoReferencia

```
public int getAnoReferencia()
```

---

getDiaOperacao

```
public int getDiaOperacao()
```

Formato:
Gravando no banco de dados: ano - mes - dia hora : min
Consultando o banco de dados: dia - mes - ano hora : min

---

getMesOperacao

```
public int getMesOperacao()
```

Formato:
Gravando no banco de dados: ano - mes - dia hora : min
Consultando o banco de dados: dia - mes - ano hora : min

---

getAnoOperacao

```
public int getAnoOperacao()
```

Formato:
Gravando no banco de dados: ano - mes - dia hora : min
Consultando o banco de dados: dia - mes - ano hora : min

---

getTempoOperacao

```
public int getTempoOperacao()
```

---

getRecuperacao

```
public float getRecuperacao()
```

Retorna a recuperação do sistema

---

getFatorConcentracao

```
public float getFatorConcentracao()
```

Retorna o fator de concentração média logarítmica

---

getPressaoOsmoticaA

```
public float getPressaoOsmoticaA()
```

Retorna a pressão osmótica da alimentação

---

getConcentracaoP

```
public float getConcentracaoP()
```

Retorna o total de sólidos do permeado

---

getPressaoOsmoticaP

```
public float getPressaoOsmoticaP()
```

Retorna a pressão osmótica do permeado

---

getConcentracaoC

```
public float getConcentracaoC()
```

Retorna o total de sólidos do concentrado

---

getPressaoOsmoticaC

```
public float getPressaoOsmoticaC()
```

Retorna a pressão osmótica do concentrado

---

getFatorTemperatura

```
public float getFatorTemperatura()
```

Retorna o fator de correção de temperatura

---

getRecuperacaoS

```
public float getRecuperacaoS()
```

Retorna a recuperação de projeto (simulada) do sistema

---

getFatorConcentracaoS

```
public float getFatorConcentracaoS()
```

Retorna o fator de concentração média logarítmica simulado

---

getConcentracaoAS

```
public float getConcentracaoAS()
```

Retorna a concentração da alimentação simulada

---

getPressaoOsmoticaAS

```
public float getPressaoOsmoticaAS()
```

Retorna a pressão osmótica da alimentação simulada

---

getConcentracaoPS

```
public float getConcentracaoPS()
```

Retorna o total de sólidos do permeado simulado

---

getPressaoOsmoticaPS

```
public float getPressaoOsmoticaPS()
```

Retorna a pressão osmótica do permeado simulada

---

getConcentracaoCS

```
public float getConcentracaoCS()
```

Retorna o total de sólidos do concentrado simulado

---

getPressaoOsmoticaCS

```
public float getPressaoOsmoticaCS()
```

Retorna a pressão osmótica do concentrado simulada

---

getFatorTemperaturaS

```
public float getFatorTemperaturaS()
```

Retorna o fator de correção de temperatura simulado

---

getDiferencialPressaoN

```
public float getDiferencialPressaoN()
```

Retorna o gradiente médio de pressão efetivo da membrana (NDP)

---

setDiferencialPressaoN

```
public void setDiferencialPressaoN(float dpn)
```

---

getDiferencialPressaoNS

```
public float getDiferencialPressaoNS()
```

Retorna o gradiente médio de pressão efetivo da membrana (NDP) simulado

---

setDiferencialPressaoNS

```
public void setDiferencialPressaoNS(float dpns)
```

---

getVazaoN

```
public float getVazaoN()
```

Retorna a vazão do permeado normalizada

---

setVazaoN

```
public void setVazaoN(float qn)
```

---

getPassagemSalN

```
public float getPassagemSalN()
```

Retorna a passagem de sal normalizada

---

setPassagemSalN

```
public void setPassagemSalN(float psn)
```

---

getConcentracaoPermeadoN

```
public float getConcentracaoPermeadoN()
```

Retorna a Concentração do permeado normalizada

---

setConcentracaoPermeadoN

```
public void setConcentracaoPermeadoN(float cpn)
```

---

getTransferenciaMassaN

```
public float getTransferenciaMassaN()
```

Retorna o coeficiente de transferência de massa normalizada

---

setTransferenciaMassaN

```
public void setTransferenciaMassaN(float ctmn)
```

---

getTransferenciaMassaNS

```
public float getTransferenciaMassaNS()
```

Retorna o coeficiente de transferência de massa normalizada simulado

---

getPassagemSalNS

```
public float getPassagemSalNS()
```

Retorna a passagem de sal normalizada simulada

---

getDiagnostico

```
public java.lang.String getDiagnostico()
```

Retorna o diagnóstico de incrustação, informando a sua localização no equipamento e o seu tipo.

---

getVariacaoCTM

```
public float getVariacaoCTM()
```

---

getVariacaoDPN

```
public float getVariacaoDPN()
```

---

getVariacaoPSN

```
public float getVariacaoPSN()
```

---

formataValor

```
public static java.lang.String formataValor(float valor)
```

Retorna o valor de um parâmetro normalizado formatado tal como o usuário visualisa no browser

---

equals

```
public boolean equals(Normalizacao n)
```

Verifica a igualdade entre dois parâmetros

Parameters:

`Normalizacao` - o objeto normalizacao return true se forem iguais

ANEXO II

Análises fisico-químicas

UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
DEPARTAMENTO DE ENGENHARIA QUÍMICA
LABORATÓRIO DE REFERÊNCIA EM DESSALINIZAÇÃO
ANÁLISE FÍSICO-QUÍMICA E ORGANOLÉPTICA DE ÁGUA

Laudo Nº.:54/2003	Data da Coleta: 18/08/2003
Interessado: Apolônio M. de Brito	Resp. pela Coleta: Apolônio M. de Brito
Município: Livramento – PB	Data da Entrega da Amostra: 28/08/2003
Localidade: Sítio Sussuarana	Tipo de Recipiente: Garrafa Vidro Ron
Procedência: Cacimba	Data da Análise: 03/09/2003

PARÂMETROS	RESULTADOS	VMP (*)
Condutividade Elétrica, µmho/cm a 25 °C	873,00	---
Potencial Hidrogeniônico, pH	6,97	6,5 a 8,5
Turbidez, (uT)	0,73	1,0 a 5,0
Cor	**Objetável**	Não objetável
Odor	**Objetável**	Não objetável
Sabor	**Objetável**	Não objetável
Dureza em Cálcio, mg/L Ca^{++}	45,00	---
Dureza em Magnésio, mg/L Mg^{++}	22,20	---
Dureza Total, mg/L $CaCO_3$	205,00	500,00
Sódio, mg/L Na^{+}	109,34	---
Potássio, mg/L K^{+}	14,26	---
Estrôncio, mg/L Sr^{++}	---	---
Bário, mg/L Ba^{++}	---	1,00
Ferro Total, mg/L	0,08	0,30
Manganês, mg/L Mn^{++}	---	0,05
Alcalinidade em Hidróxidos, mg/L $CaCO_3$	---	---
Alcalinidade em Carbonatos, mg/L $CaCO_3$	0,00	---
Alcalinidade em Bicarbonatos, mg/L $CaCO_3$	148,80	---
Alcalinidade Total, mg/L $CaCO_3$	148,80	---
Sulfato, mg/L SO_4^{--}	42,00	400,00
Cloreto, mg/L Cl^{-}	177,50	250,00
Nitrato, mg/L NO_3^{-}	0,58	10,00
Nitrito, mg/L NO_2^{-}	0,08	1,00
Sílica, mg/L SiO_2	19,20	---
ILS (Índice de Saturação de Langelier)	- 0,61	≤ 0
Total de Sólidos Dissolvidos, mg/L	**611,78**	1.000,0

(*)VMP - Valor Máximo Permissível ou recomendável pela Legislação Brasileira (PORTARIA 36/90 MS)

LAUDO:

De acordo com os resultados analíticos acima relacionados, **esta água não se encontra dentro dos padrões de potabilidade**, no que se refere aos parâmetros físico-químicos.

OBSERVAÇÕES:

1- Os resultados se referem única e exclusivamente à amostra de água analisada neste laboratório.
2- Os dados de identificação da amostra foram fornecidos pelo interessado.
3- A divulgação dos resultados desta análise, assim como sua utilização para quaisquer fins, é de exclusiva responsabilidade do interessado.

Análise realizada por: Prof. Kepler B. França (CRQ – 01.303.119)

Visto da Coordenação: Prof. Kepler B. França Data: 04/09/2003

LABDES

Laboratório de Referência em Dessalinização

Laudo Nº.: 05/2004	Data da Coleta: 05/03/2004
Interessado: CAGEPA	Resp. pela Coleta: Interessado
Município: Bananeiras - PB	Data da Entrega da Amostra: 05/03/2004
Localidade: Barragem Jandaia	Tipo de Recipiente: Garrafa plástica
Procedência: Barragem	Data da Análise: 12/03/2004

PARÂMETROS	RESULTADOS	VMP (*)
Condutividade Elétrica, µmho/cm a 25 °C	1.543	---
Potencial Hidrogeniônico, pH	7,63	6,0 a 9,5
Turbidez, (uT)	50	5,0
Oxigênio Dissolvido, mg/l	2,51	---
Cor	Não objetável	Não objetável
Odor	Não objetável	Não objetável
Sabor	**Objetável**	Não objetável
Dureza em Cálcio (Ca^{++}), mg/L	52	---
Dureza em Magnésio (Mg^{++}), mg/L	40	---
Dureza Total ($CaCO_3$), mg/L	299	500,0
Sódio (Na^{+}), mg/L	**251**	200,0
Potássio (K^{+}), mg/L	12,4	---
Ferro Total, mg/L	0,5	0,3
Manganês (Mn^{++}), mg/L	---	0,1
Alcalinidade em Hidróxidos, mg/L ($CaCO_3$)	0,0	---
Alcalinidade em Carbonatos, mg/L ($CaCO_3$)	0,0	---
Alcalinidade em Bicarbonatos, mg/L ($CaCO_3$)	110	---
Alcalinidade Total, mg/L ($CaCO_3$)	110	---
Sulfato (SO_4^{--}), mg/L	140	250,0
Cloreto (Cl^{-}), mg/L	**408**	250,0
Nitrato (NO_3^{-}), mg/L	0,31	10,0
Nitrito (NO_2^{-}), mg/L	0,05	1,0
Sílica, mg/L (SiO_2)	14,1	---
ILS (Índice de Saturação de Langelier)	-0,186	≤ 0
Total de Sólidos Dissolvidos, mg/L	**1.053**	1.000

(*)VMP - Valor Máximo Permissível ou recomendável pela Legislação Brasileira (PORTARIA 1469/00 MS).

LAUDO:

De acordo com os resultados analíticos acima relacionados, **esta água não se encontra dentro dos padrões de potabilidade**, no que se refere aos parâmetros físico-químicos.

OBSERVAÇÕES:

1- Os resultados se referem única e exclusivamente à amostra de água analisada neste laboratório.
2- Os dados de identificação da amostra foram fornecidos pelo interessado.

A divulgação dos resultados desta análise, assim como sua utilização para quaisquer fins, é de exclusiva responsabilidade do interessado.

Análise realizada por: Prof. Kepler B. França (CRQ – 01.303.119)

Visto da Coordenação: Prof. Kepler B. França Prof. Márcio L. F. Pereira Data: 12/03/2004

UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
DEPARTAMENTO DE ENGENHARIA QUÍMICA
LABORATÓRIO DE REFERÊNCIA EM DESSALINIZAÇÃO
ANÁLISE FÍSICO-QUÍMICA E ORGANOLÉPTICA DE ÁGUA

Laudo N^{O}.: 01/2004	Data da Coleta: 23/01/2004
Interessado: SEMARH	Resp. pela Coleta: Interessado
Município: São João do Cariri - PB	Data da Entrega da Amostra: 23/01/2004
Localidade: Sítio Malhada	Tipo de Recipiente: Garrafa plástica
Procedência: Poço	Data da Análise: 28/01/2004

PARÂMETROS	RESULTADOS	VMP (*)
Condutividade Elétrica, µmho/cm a 25 °C	3.350	---
Potencial Hidrogeniônico, pH	8,43	6,0 a 9,5
Turbidez, (uT)	0,0	5,0
Oxigênio Dissolvido, mg/l	6,01	---
Cor	Não objetável	Não objetável
Odor	Não objetável	Não objetável
Sabor	**Objetável**	Não objetável
Dureza em Cálcio, mg/L Ca^{++}	78	---
Dureza em Magnésio, mg/L Mg^{++}	73	---
Dureza Total, mg/L $CaCO_3$	498	500,0
Sódio, mg/L Na^{+}	**608**	200,0
Potássio, mg/L K^{+}	18	---
Ferro Total, mg/L	0,01	0,3
Manganês, mg/L Mn^{++}	---	0,1
Alcalinidade em Hidróxidos, mg/L $CaCO_3$	0,0	---
Alcalinidade em Carbonatos, mg/L $CaCO_3$	24	---
Alcalinidade em Bicarbonatos, mg/L $CaCO_3$	320	---
Alcalinidade Total, mg/L $CaCO_3$	344	---
Sulfato, mg/L SO_4^{-}	170	250,0
Cloreto, mg/L Cl^{-}	**909**	250,0
Nitrato, mg/L NO_3^{-}	0,04	10,0
Nitrito, mg/L NO_2^{-}	0,01	1,0
Sílica, mg/L SiO_2	21	---
ILS (Índice de Saturação de Langelier)	**1,254**	≤ 0
Total de Sólidos Dissolvidos, mg/L	**2.281**	1.000

(*)VMP - Valor Máximo Permissível ou recomendável pela Legislação Brasileira (PORTARIA 1469/00 MS)

LAUDO:
De acordo com os resultados analíticos acima relacionados, **esta água não se encontra dentro dos padrões de potabilidade,** no que se refere aos parâmetros físico-químicos.

OBSERVAÇÕES:
1- Os resultados se referem única e exclusivamente à amostra de água analisada neste laboratório.
2- Os dados de identificação da amostra foram fornecidos pelo interessado.

A divulgação dos resultados desta análise, assim como sua utilização para quaisquer fins, é de exclusiva responsabilidade do interessado.

Análise realizada por: Prof. Kepler B. França (CRQ – 01.303.119)

Visto da Coordenação: Prof. Kepler B. França Data: 28/01/2004

LABDES

Laboratório de Referência em Dessalinização

Laudo Nº.: 08/2004	Data da Coleta: 18/03/2004
Interessado: Prefeitura Municipal de Solidão	Resp. pela Coleta: Interessado
Município: Solidão - PE	Data da Entrega da Amostra: 29/03/2004
Localidade: Sítio Barra	Tipo de Recipiente: Garrafa plástica
Procedência: Poço artesiano	Data da Análise: 02/04/2004

PARÂMETROS	RESULTADOS	VMP (*)
Condutividade Elétrica, µmho/cm a 25 °C	5.280	---
Potencial Hidrogeniônico, pH	6,89	6,0 a 9,5
Turbidez, (uT)	0,77	5,0
Oxigênio Dissolvido, mg/l	3,56	---
Cor	Não objetável	Não objetável
Odor	Não objetável	Não objetável
Sabor	**Objetável**	Não objetável
Dureza em Cálcio (Ca^{++}), mg/L	415	---
Dureza em Magnésio (Mg^{++}), mg/L	108	---
Dureza Total ($CaCO_3$), mg/L	**1.489**	500,0
Sódio (Na^{+}), mg/L	**598**	200,0
Potássio (K^{+}), mg/L	2,6	---
Ferro Total, mg/L	0,01	0,3
Manganês (Mn^{++}), mg/L	---	0,1
Alcalinidade em Hidróxidos, mg/L ($CaCO_3$)	0,0	---
Alcalinidade em Carbonatos, mg/L ($CaCO_3$)	0,0	---
Alcalinidade em Bicarbonatos, mg/L ($CaCO_3$)	500	---
Alcalinidade Total, mg/L ($CaCO_3$)	500	---
Sulfato (SO_4^{--}), mg/L	60	250,0
Cloreto (Cl^{-}), mg/L	**1.690**	250,0
Nitrato (NO_3^{-}), mg/L	0,13	10,0
Nitrito (NO_2^{-}), mg/L	0,09	1,0
Sílica, mg/L (SiO_2)	16,6	---
ILS (Índice de Saturação de Langelier)	**0,493**	≤ 0
Total de Sólidos Dissolvidos, mg/L	**3.501**	1.000

(*)VMP - Valor Máximo Permissível ou recomendável pela Legislação Brasileira (PORTARIA 1469/00 MS).

LAUDO:

De acordo com os resultados analíticos acima relacionados, **esta água não se encontra dentro dos padrões de potabilidade**, no que se refere aos parâmetros físico-químicos.

OBSERVAÇÕES:

1- Os resultados se referem única e exclusivamente à amostra de água analisada neste laboratório.
2- Os dados de identificação da amostra foram fornecidos pelo interessado.

A divulgação dos resultados desta análise, assim como sua utilização para quaisquer fins, é de exclusiva responsabilidade do interessado.

Análise realizada por: Prof. Kepler B. França (CRQ – 01.303.119)

Visto da Coordenação: Prof. Kepler B. França Data: 05/04/2004

LABDES
Laboratório de Referência em Dessalinização

Laudo N^{O}.: 16/2004	Data da Coleta: 13/05/2004
Interessado: Jose Waltemar	Resp. pela Coleta: Interessado
Município: Patos - PB	Data da Entrega da Amostra: 14/05/2004
Localidade: BR 361 KM 03	Tipo de Recipiente: Garrafa plástica
Procedência: Poço	Data da Análise: 18/05/2004

PARÂMETROS	RESULTADOS	VMP (*)
Condutividade Elétrica, µmho/cm a 25 °C	7.410	---
Potencial Hidrogeniônico, pH	6,70	6,0 a 9,5
Turbidez, (uT)	1,67	5,0
Oxigênio Dissolvido, mg/l	1,76	---
Cor	Não objetável	Não objetável
Odor	Não objetável	Não objetável
Sabor	**Objetável**	Não objetável
Dureza em Cálcio (Ca^{++}), mg/L	769	---
Dureza em Magnésio (Mg^{++}), mg/L	417	---
Dureza Total ($CaCO_3$), mg/L	**3.660**	500,0
Sódio (Na^{+}), mg/L	**420**	200,0
Potássio (K^{+}), mg/L	13,6	---
Ferro Total, mg/L	0,23	0,3
Manganês (Mn^{++}), mg/L	---	0,1
Alcalinidade em Hidróxidos, mg/L ($CaCO_3$)	0,0	---
Alcalinidade em Carbonatos, mg/L ($CaCO_3$)	0,0	---
Alcalinidade em Bicarbonatos, mg/L ($CaCO_3$)	427	---
Alcalinidade Total, mg/L ($CaCO_3$)	427	---
Sulfato (SO_4^{--}), mg/L	**245**	250,0
Cloreto (Cl^{-}), mg/L	**2.856**	250,0
Nitrato (NO_3^{-}), mg/L	1,20	10,0
Nitrito (NO_2^{-}), mg/L	0,49	1,0
Sílica, mg/L (SiO_2)	41	---
ILS (Índice de Saturação de Langelier)	**0,573**	≤ 0
Total de Sólidos Dissolvidos Secos a 180°C, mg/L	**5.232**	1.000

(*)VMP - Valor Máximo Permissível ou recomendável pela Legislação Brasileira (PORTARIA 1469/00 MS).

LAUDO:

De acordo com os resultados analíticos acima relacionados, **esta água não se encontra dentro dos padrões de potabilidade,** no que se refere aos parâmetros físico-químicos.

OBSERVAÇÕES:

1- Os resultados se referem única e exclusivamente à amostra de água analisada neste laboratório.
2- Os dados de identificação da amostra foram fornecidos pelo interessado.

A divulgação dos resultados desta análise, assim como sua utilização para quaisquer fins, é de exclusiva responsabilidade do interessado.

Análise realizada por: Prof. Kepler B. França (CRQ 01.303.119)

Visto da Coordenação: Prof. Kepler B. França — Data: 18/05/2004

Laboratório de Referência de Dessalinização - SRH/MMA- UFCG/CCT/DEQ
Av. Aprígio Veloso 882 Bodocongó, 58.109-970 - C. Grande, PB - Fone /Fax: 0xx-83-310.1116.
labdes@labdes.ufcg.edu.br

UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
DEPARTAMENTO DE ENGENHARIA QUÍMICA
LABORATÓRIO DE REFERÊNCIA EM DESSALINIZAÇÃO
ANÁLISE FÍSICO-QUÍMICA E ORGANOLÉPTICA DE ÁGUA

Laudo N^{O}.: 56/2003	Data da Coleta: 14/07/2003
Interessado: Aluízio Guimarães Santos	Resp. pela Coleta: Aluízio Guimarães Santos
Município: Cubati – PB	Data da Entrega da Amostra: 14/07/2003
Localidade: Sítio Cacimba de Besta (Poço Novo)	Tipo de Recipiente: Garrafa plástica
Procedência: Poço	Data da Análise: 17/07/2003

PARÂMETROS	RESULTADOS	VMP (*)
Condutividade Elétrica, µmho/cm a 25 °C	9.940,00	---
Potencial Hidrogeniônico, pH	6,94	6,5 a 8,5
Turbidez, (uT)	0,40	1,0 a 5,0
Cor	Não objetável	Não objetável
Odor	Não objetável	Não objetável
Sabor	Objetável	Não objetável
Dureza em Cálcio, mg/L Ca^{++}	630,00	---
Dureza em Magnésio, mg/L Mg^{++}	612,00	---
Dureza Total, mg/L $CaCO_3$	4.125,00	500,0
Sódio, mg/L Na^{+}	1.214,70	---
Potássio, mg/L K^{+}	76,60	---
Estrôncio, mg/L Sr^{++}		---
Bário, mg/L Ba^{++}		1,0
Ferro Total, mg/L	0,09	0,3
Manganês, mg/L Mn^{++}		0,05
Alcalinidade em Hidróxidos, mg/L $CaCO_3$	0,00	---
Alcalinidade em Carbonatos, mg/L $CaCO_3$	50,00	---
Alcalinidade em Bicarbonatos, mg/L $CaCO_3$	285,00	---
Alcalinidade Total, mg/L $CaCO_3$	335,00	---
Sulfato, mg/L SO_4^{--}	520,00	400,0
Cloreto, mg/L Cl^{-}	3.976,00	250,0
Nitrato, mg/L NO_3^{-}	0,80	10,0
Nitrito, mg/L NO_2^{-}	0,82	1,0
Sílica, mg/L SiO_2	65,00	---
ILS (Índice de Saturação de Langelier)	0,60	≤ 0
Total de Sólidos Dissolvidos, mg/L	7.473,70	1.000,0

(*)VMP - Valor Máximo Permissível ou recomendável pela Legislação Brasileira (PORTARIA 36/90 MS)

LAUDO:
De acordo com os resultados analíticos acima relacionados, esta água não se encontra dentro dos padrões de potabilidade, no que se refere aos parâmetros físico-químicos.

OBSERVAÇÕES:
1- Os resultados se referem única e exclusivamente à amostra de água analisada neste laboratório.
2- Os dados de identificação da amostra foram fornecidos pelo interessado.
3- A divulgação dos resultados desta análise, assim como sua utilização para quaisquer fins, é de exclusiva responsabilidade do interessado.

Análise realizada por: Prof. Kepler B. França (CRQ – 01.303.119)

Visto da Coordenação: Prof. Kepler B. França Data: 17/07/2003

Prof. Dr. Kepler B. França
Coordenador do Laboratório de Referência em Dessalinização

UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE CIÊNCIAS E TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA QUÍMICA
LABORATÓRIO DE REFERÊNCIA EM DESSALINIZAÇÃO

Laudo Nº.: 23/2003	Data da Coleta: 30/04/2003
Órgão Resp.: Sr. Bartolomeu	Resp. pela Coleta : Sr. Bartolomeu
Município: Lagoa Seca_PB	Data da Entrega da Amostra: 08/05/2003
Localidade: Lagoa Seca_PB	Tipo de Recipiente: Pet
Procedência: Poço	Data da Análise: 12/05/2003

PARÂMETROS	RESULTADOS	VMP (*)
Condutividade Elétrica, µmho/cm a 25 °C	13.400,00	---
Potencial Hidrogeniônico, pH	6,93	6,5 a 8,5
Turbidez, (uT)	0,14	1,0 a 5,0
Cor	Objetável	Não objetável
Odor	Objetável	Não objetável
Sabor	Objetável	Não objetável
Dureza em Cálcio, mg/L Ca^{++}	340,00	---
Dureza em Magnésio, mg/L Mg^{++}	741,00	---
Dureza Total, mg/L $CaCO_3$	3.937,50	500,0
Sódio, mg/L Na^{+}	2.077,90	---
Potássio, mg/L K^{+}	52,00	---
Estrôncio, mg/L Sr^{++}	---	---
Bário, mg/L Ba^{++}	---	1,0
Ferro Total, mg/L	0,10	0,3
Manganês, mg/L Mn^{++}	---	0,05
Alcalinidade em Hidróxidos, mg/L $CaCO_3$	0,00	---
Alcalinidade em Carbonatos, mg/L $CaCO_3$	0,00	---
Alcalinidade em Bicarbonatos, mg/L $CaCO_3$	26,00	---
Alcalinidade Total, mg/L $CaCO_3$	26,00	---
Sulfato, mg/L SO_4^{-}	72,00	400,0
Cloreto, mg/L Cl^{-}	5.697,8	250,0
Nitrato, mg/L NO_3^{-}	0,09	10,0
Nitrito, mg/L NO_2^{-}	0,01	1,0
Sílica, mg/L SiO_2	25,00	---
ILS (Índice de Saturação de Langelier)	- 0.80	≥ 0
Total de Sólidos Dissolvidos, mg/L	9.037,50	1.000,0

(*)VMP - Valor Máximo Permissível ou recomendável pela Legislação Brasileira (PORTARIA 36/90 MS)

LAUDO:
De acordo com os resultados analíticos acima relacionados, "esta água não se encontra dentro dos padrões de potabilidade", no que se refere aos parâmetros físico-químicos.

OBSERVAÇÕES:
1- Os resultados se referem única e exclusivamente à amostra de água analisada neste laboratório.
2- Os dados de identificação da amostra foram fornecidos pelo interessado.
3- A divulgação dos resultados desta análise, assim como sua utilização para quaisquer fins, é de exclusiva responsabilidade do interessado.

Análise realizada por: Prof. Kepler B. França (CRQ – 01.303.119)

Laboratório de Referência de Dessalinização – MMA/SHR
Av. Aprígio Veloso 882 Bodocongó, 58.109-970 – C.Grande,PB - Fone: 0xx-83-310.1116

ANEXO III

Documentação da classe NormalizacaoDAO.java

sismodes

Class NormalizacaoDAO

```
java.lang.Object
  |
  +--sismodes.NormalizacaoDAO
```

---

public class **NormalizacaoDAO**
extends java.lang.Object

Esta classe trata das tarefas referentes a Normalização .

Field Summary

static int	ALL
static java.lang.String	CHECKED
static int	FLAG
static int	NAO_VISIVEL
static java.lang.String	NO_CHECKED
static int	VISIBLE
static int	VISIVEL

Constructor Summary

NormalizacaoDAO(sismodes.ConnectionDB connection, sismodes.Sistema sistema) Construtor que recebe uma conexão ao banco de dados do sistema

Method Summary	
java.util.Collection	getNormalizacoes(int idDessalinizador, java.lang.String dataInicio, java.lang.String dataFim) Retorna as normalizacoes do dessalinizador especificado no parâmetro e que está entre as datas início e fim. Se o idDessalinizador for zero serão listados todos os dessalinizadores.
java.util.Collection	getNormalizacoes(int idDessalinizador, java.lang.String dataInicio, java.lang.String dataFim, int quantidade) Retorna as normalizacoes do dessalinizador especificado no parâmetro e que está entre as datas início e fim. Se o idDessalinizador for zero serão listados todos os dessalinizadores.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

VISIVEL

```
public static final int VISIVEL
```

NAO_VISIVEL

```
public static final int NAO_VISIVEL
```

ALL

```
public static final int ALL
```

FLAG

```
public static final int FLAG
```

VISIBLE

```
public static final int VISIBLE
```

CHECKED

```
public static final java.lang.String CHECKED
```

NO_CHECKED

```
public static final java.lang.String NO_CHECKED
```

Constructor Detail

NormalizacaoDAO

```
public NormalizacaoDAO(sismodes.ConnectionDB connection,
                       sismodes.Sistema sistema)
```

Construtor que recebe uma conexão ao banco de dados do sistema
Parameters:
`connection` - um objeto Connection que contém uma conexão ao BD
`sistema` - uma referência ao objeto Sistema.

Method Detail

getNormalizacoes

```
public java.util.Collection getNormalizacoes(int idDessalinizador,
                                             java.lang.String dataInicio,
                                             java.lang.String dataFim)
                                      throws java.lang.Exception
```

Retorna as normalizacoes do dessalinizador especificado no parâmetro e que está entre as datas início e fim.
Se o idDessalinizador for zero serão listados todos os dessalinizadores.
Parameters:
`idDessalinizador` - o id do dessalinizador das normalizacoes.
`dataInicio` - a data inicio das normalizacoes
`dataFim` - a data fim das normalizacoes

---

getNormalizacoes

```
public java.util.Collection getNormalizacoes(int idDessalinizador,
                                             java.lang.String dataInicio,
                                             java.lang.String dataFim,
                                             int quantidade)
                                      throws java.lang.Exception
```

Retorna as normalizacoes do dessalinizador especificado no parâmetro e que está entre as datas início e fim.
Se o idDessalinizador for zero serão listados todos os dessalinizadores.
Parameters:
`idDessalinizador` - o id do dessalinizador das normalizacoes.
`dataInicio` - a data inicio das normalizacoes
`dataFim` - a data fim das normalizacoes
`quantidade` - a quantidades de normalizacoes que vai estar na coleção. Se for <= 0 serão listadas todas as normalizacoes

ANEXO IV

Documentação da classe Analise.java

sismodes.beans

Class Analise

```
java.lang.Object
  |
  +--sismodes.beans.Analise
```

---

public class **Analise**
extends java.lang.Object
implements java.io.Serializable

Um bean que contém os dados da análise fisico-química

See Also:
Serialized Form

Constructor Summary

Analise()

Method Summary

boolean	**equals**(Analise analise) Verifica se duas análises são iguais
java.lang.String	**getAlcalinidadeBicarbonatos**()
java.lang.String	**getAlcalinidadeCarbonatos**()
java.lang.String	**getAlcalinidadeHidroxidos**()
java.lang.String	**getAlcalinidadeTotal**()
int	**getAnoAnalise**()
int	**getAnoColeta**()
int	**getAnoEntrega**()

java.lang.String	**getBario**()
java.lang.String	**getCloreto**()
java.lang.String	**getCondutividade**()
java.lang.String	**getCor**()
int	**getDiaAnalise**()
int	**getDiaColeta**()
int	**getDiaEntrega**()
java.lang.String	**getDurezaCalcio**()
java.lang.String	**getDurezaMagnesio**()
java.lang.String	**getDurezaTotal**()
java.lang.String	**getEstroncio**()
java.lang.String	**getFerro**()
int	**getIdAnalise**()
int	**getIdPoco**()
float	**getIndiceLangelier**() Retorna o índice de saturação de Langelier
java.lang.String	**getLaudo**()
java.lang.String	**getLaudoDesc**() Retorna a descrição do laudo
java.lang.String	**getLocalidade**()
java.lang.String	**getManganes**()
int	**getMesAnalise**()
int	**getMesColeta**()

int	getMesEntrega()
java.lang.String	getNitrato()
java.lang.String	getNitrito()
java.lang.String	getOdor()
java.lang.String	getOrgao()
float	getPhSaturacao() Retorna o pH de saturação do carbonato de cálcio
java.lang.String	getPotassio()
java.lang.String	getPotencial()
java.lang.String	getProcedencia()
java.lang.String	getRealizadorAnalise()
java.lang.String	getRecipiente()
java.lang.String	getRespColeta()
java.lang.String	getSabor()
java.lang.String	getSilica()
java.lang.String	getSodio()
java.lang.String	getSulfato()
java.lang.String	getTotalSolidos()
java.lang.String	getTurbidez()
void	setAlcalinidadeBicarbonatos (java.lang.String alcalinidadeBicarbonatos)
void	setAlcalinidadeCarbonatos (java.lang.String alcalinidadeCarbonatos)

void	**setAlcalinidadeHidroxidos** (java.lang.String alcalinidadeHidroxidos)
void	**setAlcalinidadeTotal**(java.lang.String alcalinidadeTotal)
void	**setAnoAnalise**(int anoAnalise)
void	setAnoColeta(int anoColeta)
void	**setAnoEntrega**(int anoEntrega)
void	**setBario**(java.lang.String bario)
void	**setCloreto**(java.lang.String cloreto)
void	setCondutividade(java.lang.String condutividade)
void	**setCor**(java.lang.String cor)
void	setDiaAnalise(int diaAnalise)
void	setDiaColeta(int diaColeta)
void	**setDiaEntrega**(int diaEntrega)
void	**setDurezaCalcio**(java.lang.String durezaCalcio)
void	**setDurezaMagnesio**(java.lang.String durezaMagnesio)
void	**setDurezaTotal**(java.lang.String durezaTotal)
void	**setEstroncio**(java.lang.String estroncio)
void	setFerro(java.lang.String ferro)
void	setIdAnalise(int idAnalise)
void	**setIdPoco**(int idPoco)
void	**setLaudo**(java.lang.String laudo)
void	setLaudoDesc(java.lang.String laudoDesc) Altera a descrição do laudo

void	**setLocalidade**(java.lang.String localidade)
void	**setManganes**(java.lang.String manganes)
void	**setMesAnalise**(int mesAnalise)
void	**setMesColeta**(int mesColeta)
void	**setMesEntrega**(int mesEntrega)
void	**setNitrato**(java.lang.String nitrato)
void	**setNitrito**(java.lang.String nitrito)
void	**setOdor**(java.lang.String odor)
void	**setOrgao**(java.lang.String orgao)
void	**setPotassio**(java.lang.String potassio)
void	**setPotencial**(java.lang.String potencial)
void	**setProcedencia**(java.lang.String procedencia)
void	**setRealizadorAnalise**(java.lang.String realizadorAnalise)
void	**setRecipiente**(java.lang.String recipiente)
void	**setRespColeta**(java.lang.String respColeta)
void	**setSabor**(java.lang.String sabor)
void	**setSilica**(java.lang.String silica)
void	**setSodio**(java.lang.String sodio)
void	**setSulfato**(java.lang.String sulfato)
void	**setTotalSolidos**(java.lang.String totalSolidos)
void	**setTurbidez**(java.lang.String turbidez)

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Analise

public **Analise**()

Method Detail

getIdAnalise

public int **getIdAnalise**()

---

getIdPoco

public int **getIdPoco**()

---

getLaudo

public java.lang.String **getLaudo**()

---

getOrgao

public java.lang.String **getOrgao**()

---

getLocalidade

public java.lang.String **getLocalidade**()

---

getProcedencia

public java.lang.String **getProcedencia**()

---

getDiaColeta

public int **getDiaColeta**()

---

getMesColeta

```
public int getMesColeta()
```

---

getAnoColeta

```
public int getAnoColeta()
```

---

getRespColeta

```
public java.lang.String getRespColeta()
```

---

getDiaEntrega

```
public int getDiaEntrega()
```

---

getMesEntrega

```
public int getMesEntrega()
```

---

getAnoEntrega

```
public int getAnoEntrega()
```

---

getRecipiente

```
public java.lang.String getRecipiente()
```

---

getDiaAnalise

```
public int getDiaAnalise()
```

---

getMesAnalise

```
public int getMesAnalise()
```

---

getAnoAnalise

```
public int getAnoAnalise()
```

---

getCondutividade

```
public java.lang.String getCondutividade()
```

---

getPotencial

```
public java.lang.String getPotencial()
```

---

getTurbidez

```
public java.lang.String getTurbidez()
```

---

getCor

```
public java.lang.String getCor()
```

---

getOdor

```
public java.lang.String getOdor()
```

---

getSabor

```
public java.lang.String getSabor()
```

---

getDurezaCalcio

```
public java.lang.String getDurezaCalcio()
```

---

getDurezaMagnesio

```
public java.lang.String getDurezaMagnesio()
```

---

getDurezaTotal

```
public java.lang.String getDurezaTotal()
```

---

getSodio

```
public java.lang.String getSodio()
```

---

getPotassio

```
public java.lang.String getPotassio()
```

---

getEstroncio

```
public java.lang.String getEstroncio()
```

---

getBario

```
public java.lang.String getBario()
```

---

getFerro

```
public java.lang.String getFerro()
```

---

getManganes

```
public java.lang.String getManganes()
```

---

getAlcalinidadeHidroxidos

```
public java.lang.String getAlcalinidadeHidroxidos()
```

---

getAlcalinidadeCarbonatos

```
public java.lang.String getAlcalinidadeCarbonatos()
```

---

getAlcalinidadeBicarbonatos

```
public java.lang.String getAlcalinidadeBicarbonatos()
```

---

getAlcalinidadeTotal

```
public java.lang.String getAlcalinidadeTotal()
```

---

getSulfato

```
public java.lang.String getSulfato()
```

---

getCloreto

```
public java.lang.String getCloreto()
```

---

getNitrato

```
public java.lang.String getNitrato()
```

---

getNitrito

```
public java.lang.String getNitrito()
```

---

getSilica

```
public java.lang.String getSilica()
```

---

getTotalSolidos

```
public java.lang.String getTotalSolidos()
```

---

getRealizadorAnalise

```
public java.lang.String getRealizadorAnalise()
```

---

setIdAnalise

```
public void setIdAnalise(int idAnalise)
```

---

setIdPoco

```
public void setIdPoco(int idPoco)
```

---

setLaudo

```
public void setLaudo(java.lang.String laudo)
```

---

setOrgao

```
public void setOrgao(java.lang.String orgao)
```

---

setLocalidade

```
public void setLocalidade(java.lang.String localidade)
```

---

setProcedencia

```
public void setProcedencia(java.lang.String procedencia)
```

---

setDiaColeta

```
public void setDiaColeta(int diaColeta)
```

---

setMesColeta

```
public void setMesColeta(int mesColeta)
```

---

setAnoColeta

```
public void setAnoColeta(int anoColeta)
```

---

setRespColeta

```
public void setRespColeta(java.lang.String respColeta)
```

---

setDiaEntrega

```
public void setDiaEntrega(int diaEntrega)
```

---

setMesEntrega

```
public void setMesEntrega(int mesEntrega)
```

---

setAnoEntrega

```
public void setAnoEntrega(int anoEntrega)
```

---

setRecipiente

```
public void setRecipiente(java.lang.String recipiente)
```

---

setDiaAnalise

```
public void setDiaAnalise(int diaAnalise)
```

---

setMesAnalise

```
public void setMesAnalise(int mesAnalise)
```

---

setAnoAnalise

```
public void setAnoAnalise(int anoAnalise)
```

---

setCondutividade

```
public void setCondutividade(java.lang.String condutividade)
```

---

setPotencial

```
public void setPotencial(java.lang.String potencial)
```

---

setTurbidez

```
public void setTurbidez(java.lang.String turbidez)
```

---

setCor

```
public void setCor(java.lang.String cor)
```

---

setOdor

```
public void setOdor(java.lang.String odor)
```

---

setSabor

```
public void setSabor(java.lang.String sabor)
```

---

setDurezaCalcio

```
public void setDurezaCalcio(java.lang.String durezaCalcio)
```

---

setDurezaMagnesio

```
public void setDurezaMagnesio(java.lang.String durezaMagnesio)
```

---

setDurezaTotal

```
public void setDurezaTotal(java.lang.String durezaTotal)
```

---

setSodio

```
public void setSodio(java.lang.String sodio)
```

---

setPotassio

```
public void setPotassio(java.lang.String potassio)
```

---

setEstroncio

```
public void setEstroncio(java.lang.String estroncio)
```

---

setBario

```
public void setBario(java.lang.String bario)
```

---

setFerro

```
public void setFerro(java.lang.String ferro)
```

---

setManganes

```
public void setManganes(java.lang.String manganes)
```

---

setAlcalinidadeHidroxidos

```
public void setAlcalinidadeHidroxidos(java.lang.String alcalinidadeHidroxidos)
```

---

setAlcalinidadeCarbonatos

```
public void setAlcalinidadeCarbonatos(java.lang.String alcalinidadeCarbonatos)
```

---

setAlcalinidadeBicarbonatos

```
public void setAlcalinidadeBicarbonatos(java.lang.String alcalinidadeBicarbonato
```

---

setAlcalinidadeTotal

```
public void setAlcalinidadeTotal(java.lang.String alcalinidadeTotal)
```

---

setSulfato

```
public void setSulfato(java.lang.String sulfato)
```

---

setCloreto

```
public void setCloreto(java.lang.String cloreto)
```

---

setNitrato

```
public void setNitrato(java.lang.String nitrato)
```

---

setNitrito

```
public void setNitrito(java.lang.String nitrito)
```

---

setSilica

```
public void setSilica(java.lang.String silica)
```

---

setTotalSolidos

```
public void setTotalSolidos(java.lang.String totalSolidos)
```

---

setRealizadorAnalise

```
public void setRealizadorAnalise(java.lang.String realizadorAnalise)
```

---

getLaudoDesc

```
public java.lang.String getLaudoDesc()
```

Retorna a descrição do laudo

---

setLaudoDesc

```
public void setLaudoDesc(java.lang.String laudoDesc)
```

Altera a descrição do laudo

---

getPhSaturacao

```
public float getPhSaturacao()
```

Retorna o pH de saturação do carbonato de cálcio

---

getIndiceLangelier

```
public float getIndiceLangelier()
```

Retorna o índice de saturação de Langelier

---

equals

```
public boolean equals(Analise analise)
```

Verifica se duas análises são iguais

Parameters:

`analise` - uma análise

Returns:

true se os objetos tiverem os mesmos parâmetros

ANEXO V

Código fonte de jspChart.jsp

```
<jsp:useBean id="grafico" scope="session" class="javaside.Rbl.jspChart" /><%
      int iW, iH   ; // Taille de l image a generer
      int iColor   ; // Couleur du fond
      int iPres    ; // Presentation
      boolean b1   ; // Legend
      boolean b2   ; // Bullet
      String sFormat   ; // Format (gif/png)

      String t = null ;
      t = request.getParameter("width") ;
      if (t == null)
      iW = 400 ;
      else
      iW = java.lang.Integer.parseInt(t) ;

      t = request.getParameter("height") ;
      if (t == null)
      iH = 350 ;
      else
      iH = java.lang.Integer.parseInt(t) ;

      t = request.getParameter("color") ;
      if (t == null)
      iColor = -1 ;
      else
      iColor = java.lang.Integer.parseInt(t, 16) ;

      t = request.getParameter("pres") ;
      if (t == null)
      iPres = 1 ;
      else
      iPres = java.lang.Integer.parseInt(t) ;

      t = request.getParameter("format") ;
      if (t == null)
      sFormat = "gif" ;
      else
      sFormat = t ;

      t = request.getParameter("b1") ;
      if (t == null)
      b1 = false ;
      else
      b1 = t.equalsIgnoreCase("on") ;

      t = request.getParameter("b2") ;
      if (t == null)
      b2 = false ;
      else
      b2 = t.equalsIgnoreCase("on") ;

   // Initialisation et definition de la taille de l image
       grafico.init(iW, iH) ;

       grafico.setFontA("Dialog", 0, 8) ;
       grafico.setFontTitre("Dialog", 3, 10) ;
       grafico.setFontLegend("Dialog", 0, 8) ;

       grafico.setCol(2) ;
       grafico.addCol(0, 55255, "Tretis") ;
       grafico.addCol(1, 255,   "Lighoas") ;

       grafico.setTitle("--------------------          Test acxChart JSP") ;
       grafico.setLegend("Semaine 42/2000 ", "ELOI EH HOMOSSEXUAL !!");

       grafico.setRotate(true);
       grafico.setBkColor( iColor );

   grafico.addRow("Lun . ; 1 ;");
   grafico.addRow("Mar . ; 10 ;");
   grafico.addRow("Mer . ; 15 ;");
   grafico.addRow("Jeu . ; 30 ;");
   grafico.addRow("Ven . ; 20 ;");
   grafico.addRow("Sam . ; 8 ;");
   grafico.addRow("Dim . ; 3 ;");

```

```
77          grafico.setPress(iPres);
78          grafico.setOrigine( 40, 50) ;
79          grafico.setXPress(b1, b2, false);
80          //      grafico.setMinMax(0, 35) ;
81
82          grafico.build(true) ;
83
84          response.reset();
85          response.setContentType("image/" + sFormat );
86          response.addHeader("Content-Disposition","filename=acx." + sFormat);
87
88
89          if (sFormat.equalsIgnoreCase("png"))
90           response.getOutputStream().write(grafico.getImage(1));
91          else
92           response.getOutputStream().write(grafico.getImage(0));
93
94          response.flushBuffer();
95
96      grafico.clear() ;
97  %>
98
```

ANEXO VI

Gráficos das Variáveis

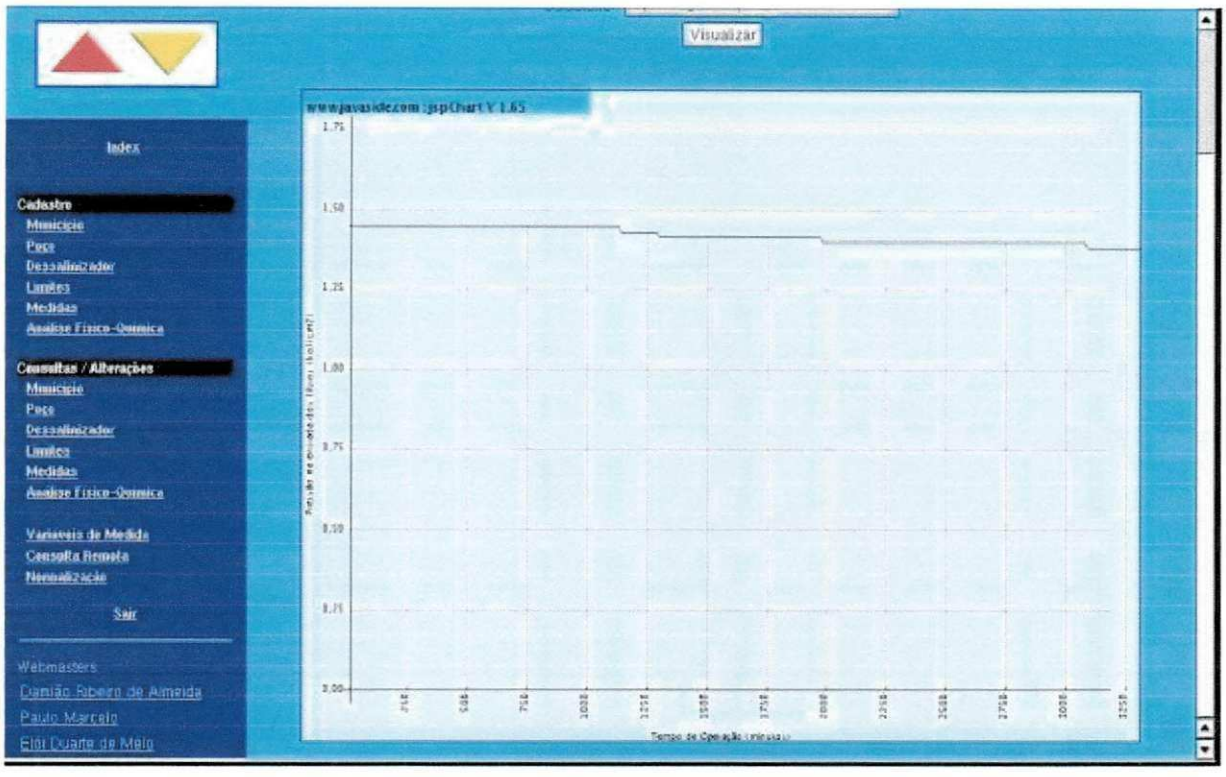


Figura 3.15: Gráfico da pressão de entrada dos filtros em função do tempo

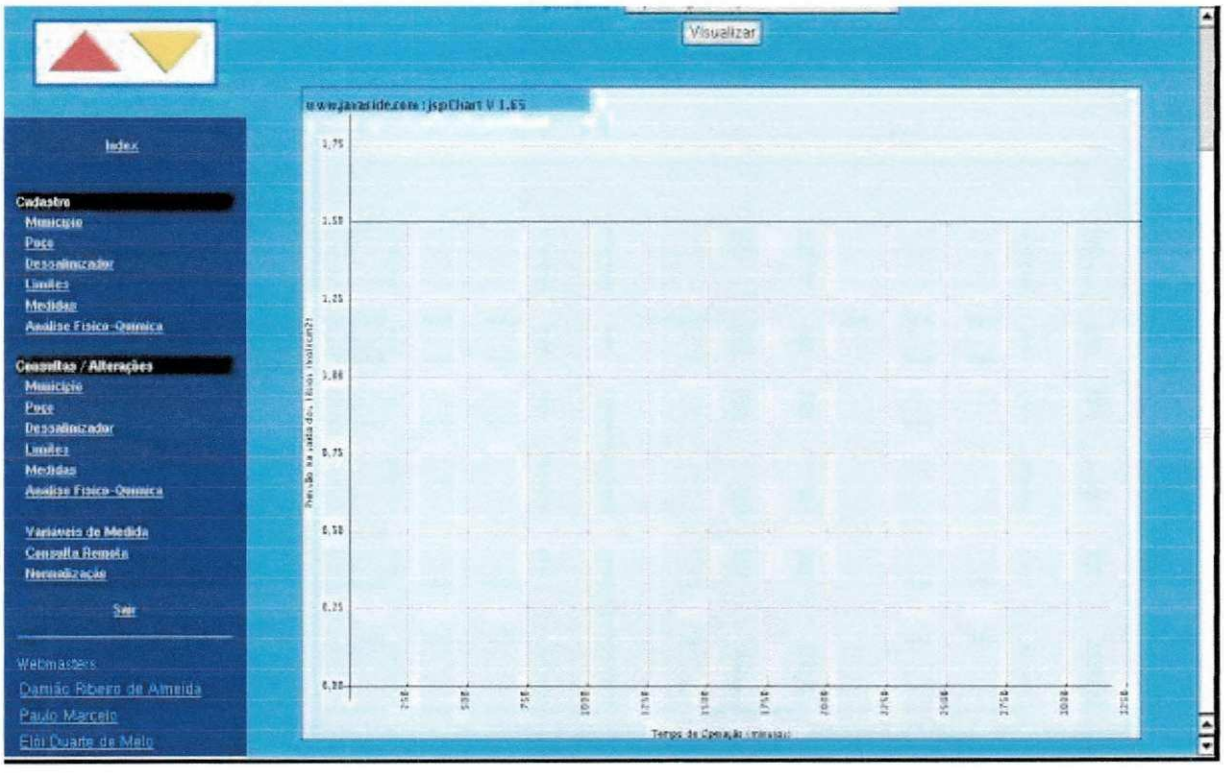


Figura 3.16: Gráfico da pressão de saída dos filtros em função do tempo

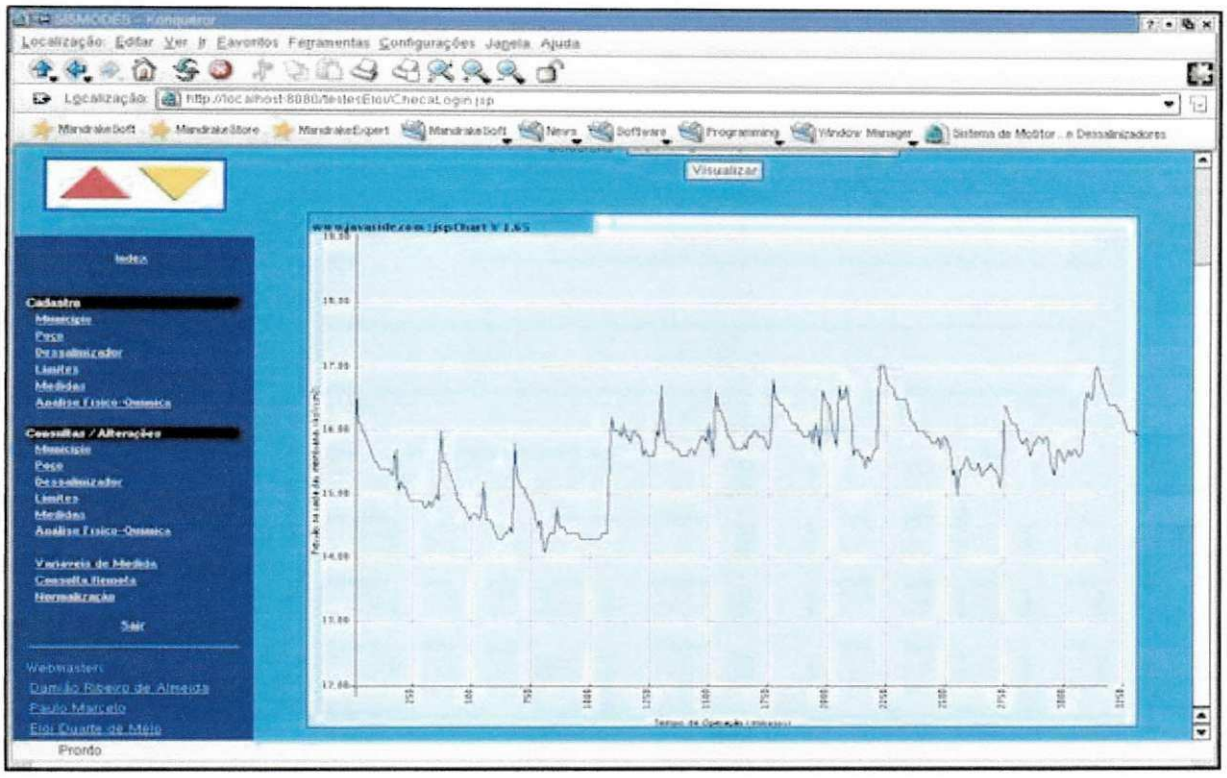


Figura 3.17: Gráfico da pressão de saída das membranas em função do tempo

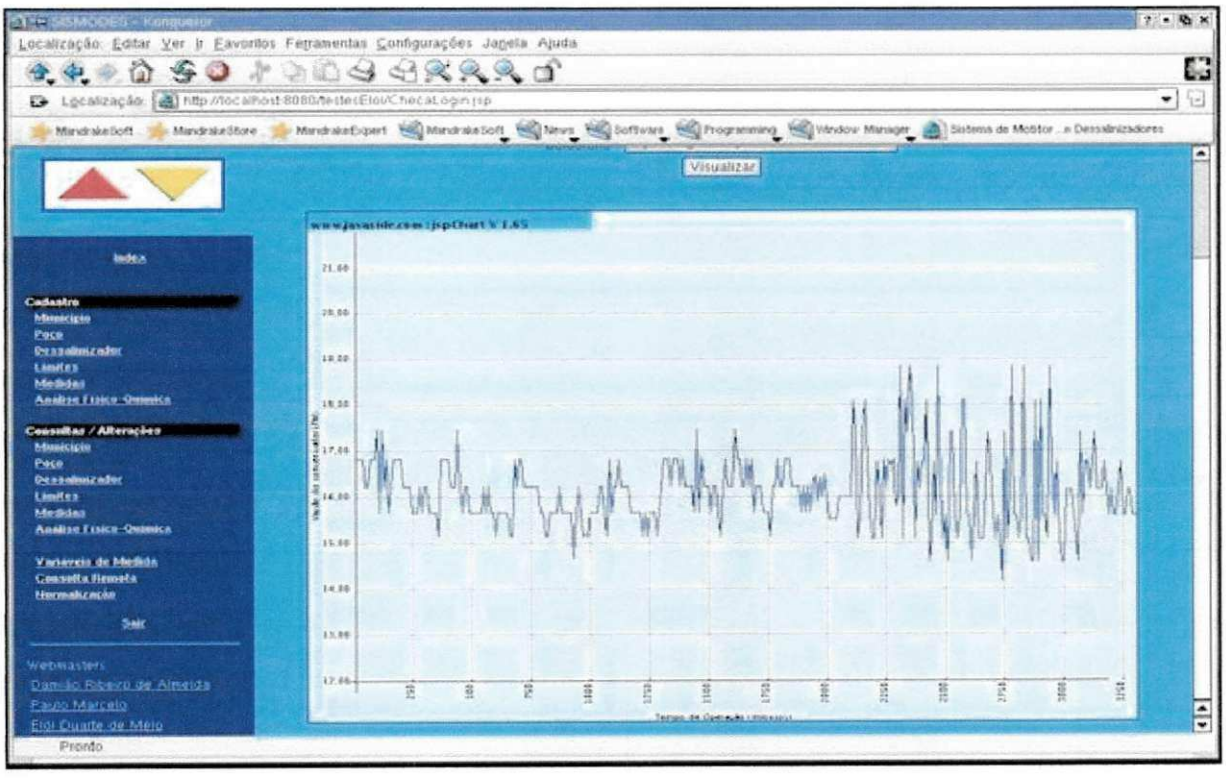


Figura 3.18: Gráfico da vazão do concentrado em função do tempo

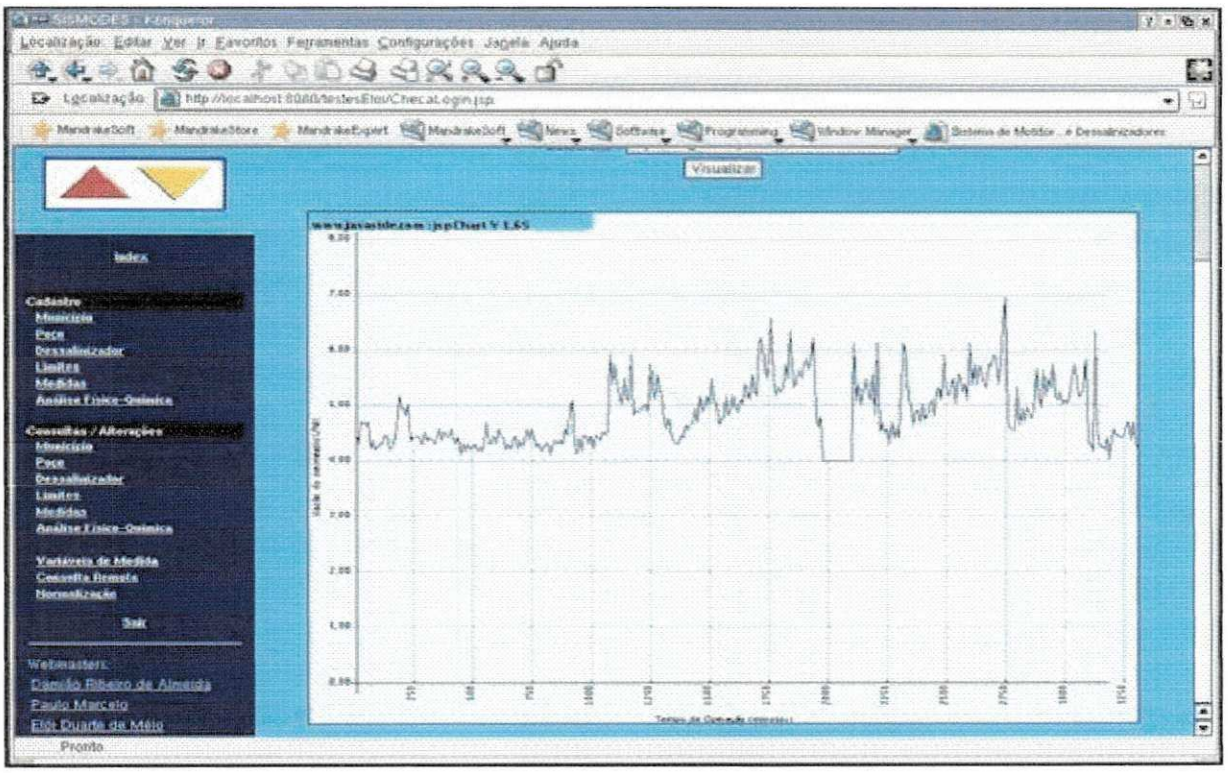


Figura 3.19: Gráfico da vazão do permeado em função do tempo

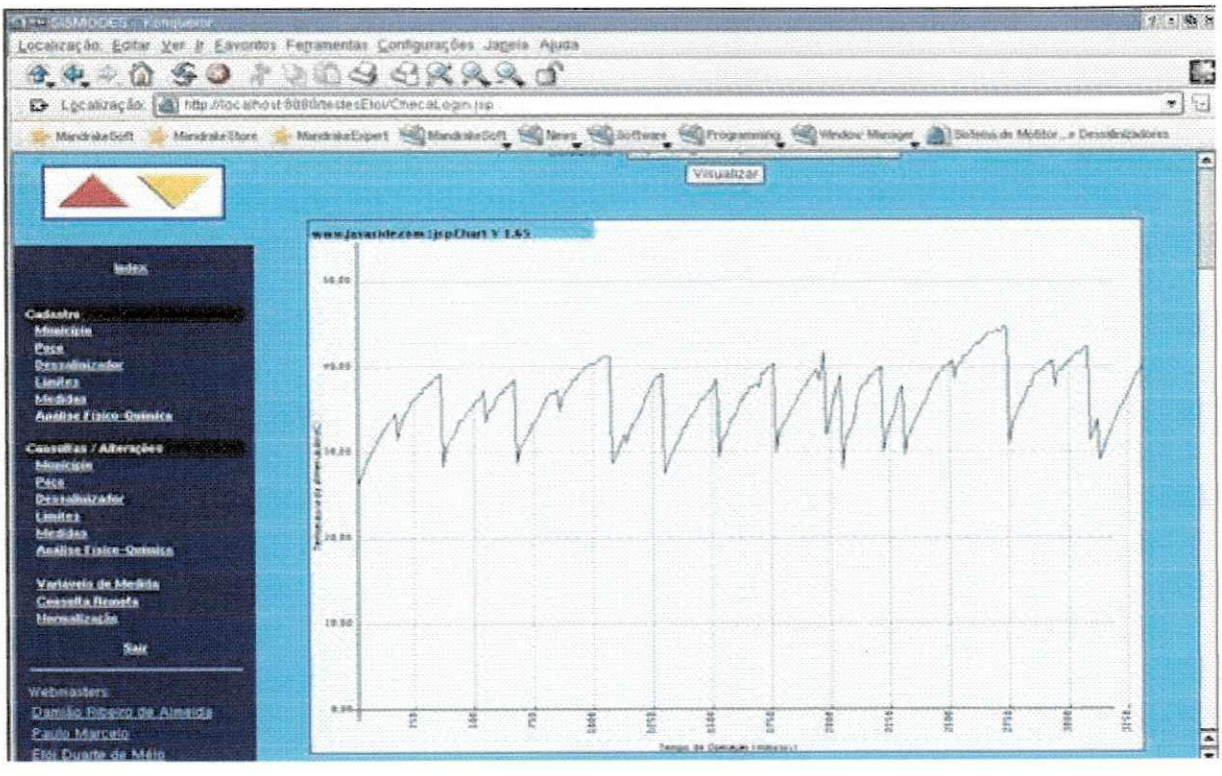


Figura 3.20: Gráfico da temperatura em função do tempo

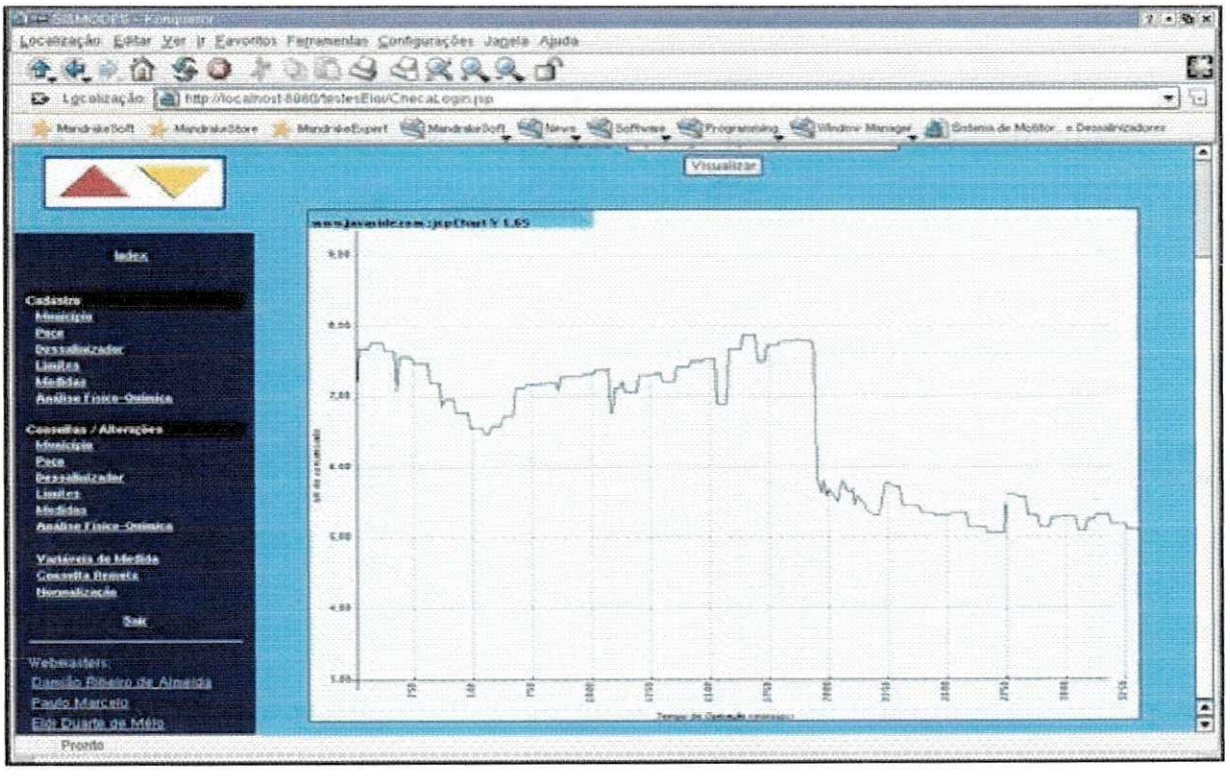


Figura 3.21: Gráfico do pH do concentrado em função do tempo

ANEXO VII

Variáveis de medidas de um sistema localizado em Caturité – PB

DADOS OPERACIONAIS										
Data	Pressão (Kgf/cm^2)			Temp.	Vazão (m^3/h)			Concentração (mg/L)		
	Aliment.	Conc.	Perm.	(°C)	Conc.	Perm.	Aliment.	Aliment.	Conc.	Perm.
06/09/1999	12,5	11,5	0,9	25,0	2,4	1,9	4,3	2570,0	4576,9	61,4
13/09/1999	12,5	11,9	0,9	24,0	2,4	1,9	4,3	2530,0	4505,8	60,3
18/09/1999	12,9	12,0	0,9	28,0	2,4	1,8	4,2	2550,0	4416,9	60,8
02/10/1999	13,0	12,0	0,9	27,0	2,4	1,9	4,3	2470,0	4399,0	58,7
12/10/1999	12,5	11,5	0,9	27,0	2,4	1,9	4,3	2350,0	4185,6	55,5
17/10/1999	12,5	11,5	0,9	23,0	2,4	1,9	4,3	2380,0	4239,0	56,3
24/10/1999	13,0	11,5	0,9	22,0	2,4	1,9	4,3	2360,0	4203,4	55,8
01/11/1999	13,0	12,0	0,9	20,0	2,4	1,9	4,3	2450,0	4363,5	58,2
06/11/1999	12,8	11,5	0,9	24,0	2,4	1,9	4,3	2495,0	4443,5	59,4
22/11/1999	13,2	12,0	1,0	25,0	2,4	1,8	4,2	2540,0	4399,6	60,6
28/11/1999	13,0	12,0	0,9	25,0	2,4	1,9	4,3	2490,0	4373,8	59,2
05/12/1999	13,0	11,5	0,9	27,0	2,4	1,9	4,3	2440,0	4345,7	57,9
21/12/1999	13,1	12,0	0,9	26,0	2,4	1,9	4,3	2340,0	4167,8	55,3
26/12/1999	13,1	12,0	0,9	27,0	2,4	1,9	4,3	2380,0	4239,0	56,3
02/01/2000	13,1	12,0	0,9	27,0	2,4	1,9	4,3	2420,0	4310,1	57,4
10/01/2000	13,5	12,0	0,9	27,0	2,4	1,9	4,3	2565,0	4505,4	61,3
17/01/2000	13,5	12,0	0,9	28,0	2,4	1,8	4,2	2710,0	4693,6	65,2
29/01/2000	13,5	12,0	0,9	29,0	2,4	1,9	4,3	2680,0	4772,5	64,4
06/02/2000	13,5	12,0	0,9	29,0	2,4	1,9	4,3	2670,0	4754,7	64,1
20/02/2000	**14,0**	**12,2**	**0,9**	**29,0**	**2,4**	**1,8**	**4,2**	**2630,0**	**4555,2**	**63,0**
05/03/2000	**13,8**	**12,5**	**0,9**	**30,0**	**2,4**	**1,8**	**4,2**	**2860,0**	**4952,9**	**69,4**
13/03/2000	13,8	11,5	0,8	30,0	2,3	1,9	4,2	2880,0	5113,6	70,0
19/03/2000	14,0	11,8	0,9	30,0	2,4	1,9	4,3	2650,0	4654,5	63,6
26/03/2000	13,8	11,5	0,8	29,0	2,4	1,9	4,3	2610,0	4648,0	62,5
02/04/2000	13,8	11,5	0,8	29,0	2,4	1,9	4,3	2700,0	4808,0	64,9
09/04/2000	13,8	11,5	0,8	29,0	2,4	1,8	4,2	2700,0	4676,3	64,9
16/04/2000	13,9	11,5	0,8	29,0	2,4	1,9	4,3	3100,0	5443,4	76,3
30/04/2000	14,2	12,0	0,8	29,0	2,4	1,8	4,2	3100,0	5367,8	76,3
14/05/2000	14,3	12,0	0,8	29,0	2,4	1,8	4,2	3500,0	6058,9	88,2
31/05/2000	14,5	12,0	0,8	30,0	2,4	1,8	4,2	3450,0	5972,5	86,6
05/06/2000	14,2	12,0	0,8	30,0	2,4	1,8	4,2	3210,0	5557,9	79,5
14/06/2000	12,0	10,8	0,8	30,0	2,1	1,5	3,6	3210,0	5446,1	79,5
18/06/2000	**12,0**	**10,8**	**0,8**	**30,0**	**2,1**	**1,5**	**3,6**	**3270,0**	**5547,7**	**81,3**
28/06/2000	**12,0**	**10,8**	**0,8**	**30,0**	**2,1**	**1,5**	**3,6**	**6220,0**	**10531,1**	**184,4**

08/07/2000	12,0	10,8	0,8	30,0	2,1	1,5	3,6	6295,0	10657,5	187,5
19/07/2000	12,0	10,8	0,8	30,0	2,1	1,5	3,6	6370,0	10783,9	190,5
13/08/2000	12,2	10,9	0,8	30,0	2,1	1,5	3,6	6370,0	10783,9	190,5
20/08/2000	12,0	10,5	0,8	30,0	2;1	1,5	3,6	6370,0	10783,9	190,5
03/09/2000	12,2	10,2	0,8	30,0	2,1	1,5	3,6	6370,0	10783,9	190,5
17/09/2000	12,6	10,0	0,8	30,0	2,1	1,5	3,6	6370,0	10783,9	190,5
01/10/2000	12,6	10,2	0,8	30,0	2,1	1,5	3,6	6370,0	10783,9	190,5