UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

Coordenação de Estágios do DEE

Relatório da Disciplina Estágio Integrado

ESTÁGIO NA INORPEL –

Indústria Nordestina de Produtos Elétricos

Desenvolvimento e Implantação do
Sistema Integrado de Controle Operacional - SICO

Discente: Otacílio de Araújo Ramos Neto
Orientador: Carlos Alberto da Rocha
Supervisor: Ricardo Joaquim M. de Brito
Empresa: INORPEL – Indústria Nordestina de Produtos Elétricos

Campina Grande – PB
Agosto/2006

UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

Coordenação de Estágios do DEE

Relatório da Disciplina Estágio Integrado

ESTÁGIO NA INORPEL –

Indústria Nordestina de Produtos Elétricos

Desenvolvimento e Implantação do Sistema Integrado de Controle Operacional - SICO

Discente: Otacílio de Araújo Ramos Neto
Matrícula: 20321127
Orientador: Carlos Alberto da Rocha
Supervisor: Ricardo Joaquim M. de Brito
Empresa: INORPEL – Indústria Nordestina de Produtos Elétricos

Campina Grande – PB
Agosto/2006

Biblioteca Setorial do CDSA. Fevereiro de 2021.

Sumé - PB

INORPEL

Cabedelo(PB), 03 de maio de 2006.

A
UFCG
Coordenação de Graduação em Engenharia Elétrica

DECLARAÇÃO

Declaramos, para os devidos fins, que **OTACÍLIO DE ARAÚJO RAMOS NETO** do Curso de Engenharia Elétrica, concluiu estágio na Inorpel Indústria Nordestina de Produtos Elétricos Ltda. localizada na Rodovia BR 230 Km 05 Cabedelo – PB, perfazendo um total de **30**(TRINTA) horas semanais, no período de **06/09/2005 à 05/03/2006.**

Ricardo Joaquim M. de Brito
Diretor Presidente

Inorpel – Indústria Nordestina de Produtos Elétricos Ltda.
Rodovia BR 230, Km 05 – Cabedelo (PB) • Fone (83) 228 – 1522 • Fax (83) 228 – 3001
C.G.C. : 08.720.054/0001 – 33 • E-mail : inorpel@inorpel.com.br

Agradecimentos

A todos que, de alguma forma, contribuíram para a minha formação profissional.

Sumário

Lista de Figuras
Lista de Siglas
1. Introdução 1
2. INORPEL 2
2.1 Organização 3
2.2 Serviços 4
3. Processo Produtivo 6
4. Tarefas Desenvolvidas 11
4.1 Justificativa 11
4.2 Objetivos 13
4.3 Estudos 14
4.3.1 SGBD 14
4.3.2 Sistema Operacional 16
4.3.3 Interface 18
4.3.4 Interpretador de Scripts 18
4.3.5 WAP 19
4.3.6 ScriptCase 20
4.4 Trabalhos Desenvolvidos 21
4.5 Interface web para uso na *intranet* 27
4.6 Interface WAP 29
4.7 Interface via *web site* 31
5. Resultados obtidos 34
6. Conclusões 35
7. Referências Bibliográficas 37
Anexo 1 - Funções em C 39
Anexo 2 - Arquivos WML 44

Lista de Figuras

Figura 1 - Sede da INORPEL 2
Figura 2a – Sala de treinamento 3
Figura 2b – Entrada do refeitório 3
Figura 3 – Organograma funcional da INORPEL 3
Figura 4 – Call center em funcionamento 5
Figura 5 – Fluxograma Elaboração e Encaminhamento de Proposta 9
Figura 6 – Fluxograma Planejamento e Execução de Serviços 10
Figura 7 – Fluxograma Acompanhamento e Controle de Execução de Serviços 10
Figura 8 – Acesso WAP 20
Figura 9 – Entradas (esquerda) e saídas (direita) de dados do SICO 22
Figura 10 – Tela principal do phpPgAdmin 23
Figura 11a – Criação de uma tabela 23
Figura 11b – Nome e tipo dos campos da tabela 24
Figura 12 – Tela principal do ScriptCase 25
Figura 13 – Fluxo de dados quando o SICO envia um e-mail 26
Figura 14a – Tela de login do SICO 28
Figura 14b – Menu com as opções do SICO 28
Figura 15 – Adicionando um equipamento a um item de contrato 29
Figura 16 – Exemplos de telas da interface WAP 30
Figura 17 – Fluxo de dados ao receber um relatório 30
Figura 18 – Contratos em andamento do cliente 31
Figura 19 – Atividades de um serviço 32
Figura 20 – Etapas das atividades 33

Lista de Siglas

ACID - Atomicity Consistency Isolation Durability
ADSL - Asymmetrical Digital Subscriber Line
BSC - Base Station Controller
BSD - Berkeley Software Distribution
BTS - Base Transceiver Station
CDMA - Code Division Multiple Access
CDPD - Cellular Digital Packet Data
CPU - Central Processor Unit
CSRG - Computer Systems Research Group
DCL - Data Control Language
DDL - Data Definition Language
DECT - Digital Enhanced Cordless Telecommunications
DHCP - Dynamic Host Configuration Protocol
DML - Data Manipulation Language
FTP - File Transference Protocol
GPRS - General Packet Radio Service
GSM - Global System for Mobile Communication
HTML - Hiper Text Markup Language
HTTP - Hiper Text Transfer Protocol
INORPEL - Indústria Nordestina de Produtos Elétricos
ISDN - Integrated Services Digital Network
MER - Modelo Entidade Relacionamento
NFS - Network File System
NIS - Network Information Service
PDA - Personal Digital Assistant
PDC - Personal D Cellular
PHP – P Hypertext Preprocessor
PHS - Personal Handyphone Systems
PPP - Point-to-Point Protocol
RAD - Rapid Application Development
SGBD - Sistema Gerenciador de Banco de Dados
SICO - Sistema Integrado de Controle Operacional
SLIP - Serial Line IP
SO - Sistema Operacional
SQL - Structured Query Language
TDMA - Time Division Multiple Access
TETRA - Terrestrial Trunked Ratio
TRAU - Transcoding Rate Adaption Unit
VPN – Virtual Private Network
WAP - Wireless Application Protocol
WBMP - Wireless Bitmap
WML – Wireless Markup Language
WWW - World Wide Web
X11R6 - X11 Release 6
iDEN - Integrated Digital Enhanced Network

Resumo

Este relatório descreve o estágio realizado por Otacílio de Araújo Ramos Neto, na Indústria Nordestina de Produtos Elétricos – INORPEL, como requisito para obtenção do grau de Engenheiro Eletricista. O relatório caracteriza a empresa, apresenta os estudos feitos pelo aluno e, por fim, as tarefas realizadas. O principal trabalho foi o desenvolvimento do Sistema Integrado de Controle Operacional – SICO. Esse sistema utiliza um banco de dados único para armazenar as informações coletadas dos vários departamentos, possui uma interface web para uso na *intranet* da INORPEL, uma interface WAP para envio de relatórios pelas equipes de campo e uma interface no *web site* da INORPEL que disponibiliza informações sobre o andamento das atividades para os clientes.

1. Introdução

Este relatório tem a finalidade de descrever as atividades desenvolvidas no estágio realizado por Otacílio de Araújo Ramos Neto na empresa Indústria Nordestina de Produtos Elétricos - INORPEL, durante o período de setembro de 2005 a maio de 2006. Ele relata ao longo do texto as atividades desenvolvidas na INORPEL e os objetivos alcançados. Inicialmente é apresentada a INORPEL e o seu processo produtivo, em seguida os estudos realizados e os trabalhos desenvolvidos. No capítulo cinco são comentados os resultados obtidos e no capítulo seis as conclusões a respeito dos trabalhos realizados.

A INORPEL é uma empresa, localizada no município de Cabedelo, no estado da Paraíba, atuante, principalmente, no ramo de prestação de serviços na área de telecomunicações. Seus clientes são, normalmente, empresas de grande porte, como Siemens e Alcatel.

A atividade realizada durante o estágio foi, quase que totalmente, o desenvolvimento de um sistema de controle operacional para auxiliar os serviços da empresa.

O sistema desenvolvido durante o estágio foi chamado de Sistema Integrado de Controle Operacional - SICO. Ele foi concebido objetivando manter uma base de dados única entre os departamentos da empresa e auxiliar o processo de acompanhamento das atividades realizadas, tanto pelos gestores da INORPEL, como também pelos seus clientes. Para o desenvolvimento do SICO, diferentes ferramentas e também tecnologias foram utilizadas, todas trabalhando em conjunto e de forma integrada. O sistema resultante consegue integrar o acompanhamento das atividades de maneira que uma informação do andamento de determinada atividade, após ser recebida pelo sistema, é disponibilizada automaticamente a todos os interessados na mesma, seja um funcionário da empresa ou um cliente.

2. INORPEL

A INORPEL – Indústria Nordestina de Produtos Elétricos Ltda., foi fundada em 1974, no município de Campina Grande – Paraíba. Inicialmente, a empresa atuava na fabricação de produtos elétricos e antenas para telecomunicações, sendo que, em 1977, fatores de mercado levaram sua diretoria a direcionar seus objetivos para a prestação de serviços de engenharia. A INORPEL surgiu como empresa pioneira na Paraíba, elaborando e executando projetos de telecomunicações nas áreas de redes, comutação e transmissão. Sua sede, que pode ser visualizada na fig. 1, conta com uma área de 8.830m^2, localizada no Km 05 da BR 230, município de Cabedelo, na Paraíba.

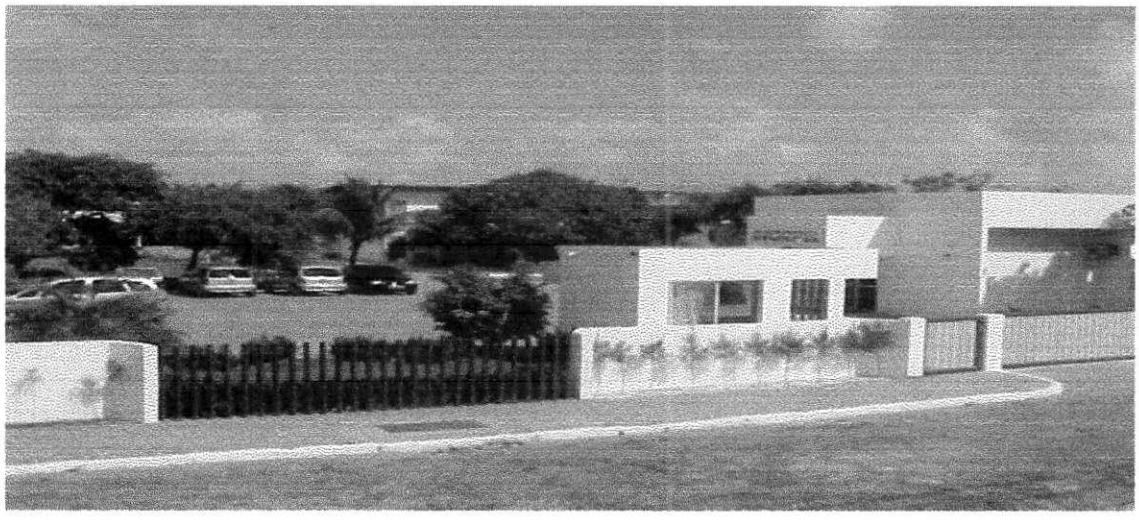
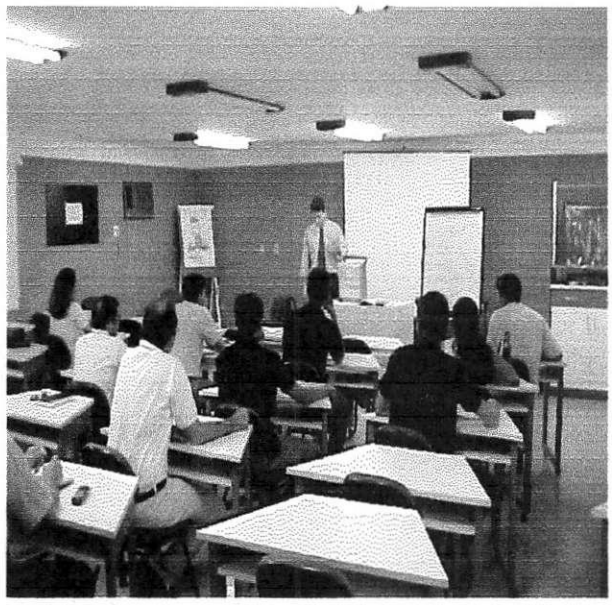


Figura 1 - Sede da INORPEL

As instalações da INORPEL, que podem ser, novamente, visualizadas na fig. 2, oferecem ambiente de trabalho climatizado, contando com um centro de treinamento, ambulatório, refeitório, áreas de lazer, estacionamento interno, posto de serviços, segurança eletrônica e humana, instalações elétricas e de comunicação.



a) Sala de treinamento b) Entrada do refeitório

Figura 2 - Instalações da INORPEL

2.1 Organização

A INORPEL está organizada de acordo com o organograma apresentado na fig. 3 e seu quadro de pessoal é composto por engenheiros, administradores e técnicos de nível médio, treinados, capacitados, e distribuídos entre os vários departamentos da empresa. O trabalho desenvolvido é fundamentado em procedimentos administrativos, técnicos e operacionais baseados na Gestão INORPEL de Qualidade Total – GIQT.

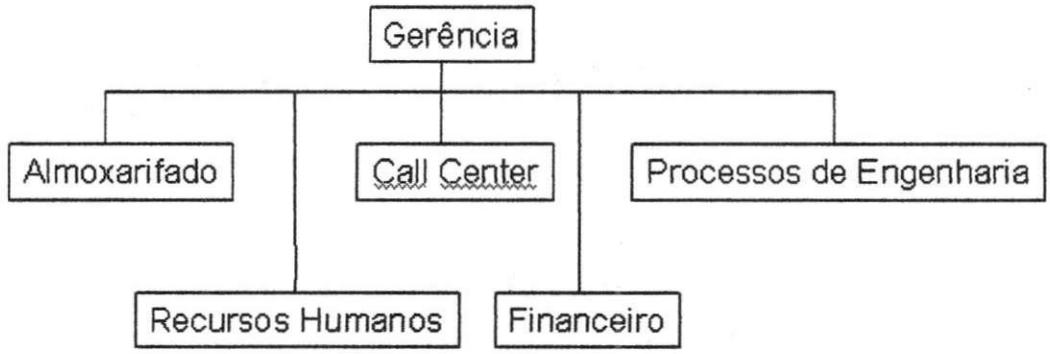


Figura 3 - Organograma funcional da INORPEL

2.2 Serviços

A INORPEL atua principalmente na área de telecomunicações, desenvolvendo atividades de projetos, montagens, instalações e testes de sistemas de comunicação e transmissão digital. Como exemplos, podem ser citados:

- **Infra-estrutura** – construção de galerias e caixas subterrâneas, instalação de sistemas elétricos de alta e baixa tensão e sistemas irradiantes;
- **Rádio transmissão/transporte** – prospecções, vistorias, estudos de viabilidade e testes de propagação, projetos para instalação de enlaces de rádio e multiplex, instalação, manutenção e testes de multiplexadores e de rádio digital de alta e baixa capacidade;
- **Sistema móvel pessoal GSM** – Instalação, manutenção e teste de micro BTS, BSC e TRAU;
- **Sistema de comunicação via satélite** – Recepção e transmissão de sinais via satélite;
- **Redes de comunicação** – Projetos, instalações e certificação de redes para comunicação de dados, voz e imagem, e instalação, testes e manutenção de acessos ISDN, ADSL, TC DATA, TC FRAME, TC ISDN, TC IP, TC PAC, DDR digital e analógico.

Além das atividades de engenharia, a empresa atua também na área de *telemarketing*, podendo desempenhar toda uma gama de serviços como vendas, pós-vendas, *help-desk*, cobrança, agendamentos, pesquisas, etc., negociados segundo modelos de terceirização flexíveis. Os serviços oferecidos de *call center* contemplam o fornecimento da

gestão integrada, recursos humanos (gerentes, supervisores e operadores) e a infra-estrutura operacional necessária, como pode ser visualizado na fig. 4.

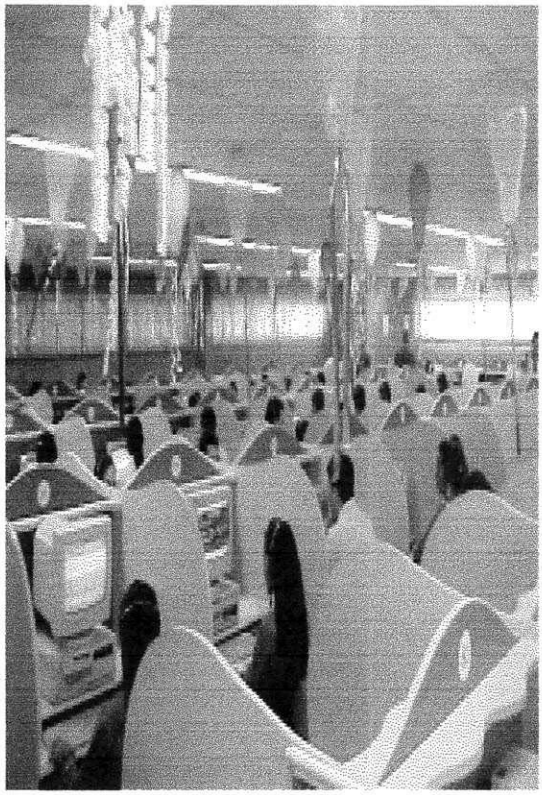


Figura 4 - Call center em funcionamento

notificar os clientes quando informações novas estiverem disponíveis para eles, e deve notificar, também, os responsáveis pelo acompanhamento das equipes de campo;

- **Facilidades de *backup*** – De nada adianta todo o cuidado na análise e implementação do sistema, se não for definida uma boa política de *backup* (a cópia de segurança de qualquer sistema de informação é algo tão importante, como negligenciado, por não ser necessário durante o dia-a-dia, até que algum problema sério aconteça);
- **Alta disponibilidade** – O sistema deve funcionar, sem interrupções, o maior tempo possível, já que deve ser capaz de receber relatórios de equipes de campo a qualquer momento.

4.3 Estudos

Para a implementação do SICO, após a definição das tarefas a serem realizadas, foi realizado um breve período de estudo. Nesse período, foram estudadas as características desejadas para o sistema a ser implantado na INORPEL e levantadas as tecnologias necessárias.

4.3.1 SGBD

O primeiro problema a ser estudado foi a forma como seria implementada a persistência dos dados. Existem várias formas de se armazenar dados em um computador e todas elas podem ser consideradas "banco de dados", como por exemplo, o sistema de

arquivos de um sistema operacional Unix ou Windows, ou mesmo planilhas. Dados podem ser armazenados até mesmo em arquivos do tipo texto e, ainda assim, ser um banco de dados, porém com um mecanismo de armazenamento e acesso muito ineficiente. Outra forma de armazenar dados pode ser por meio de um banco de dados relacional.

Relação é o termo matemático para o que se conhece mais comumente por tabela. Em uma relação, os dados são armazenados na forma de linhas e colunas. Cada linha na relação é um registro. As colunas não mudam de número com a inserção de novos registros, enquanto as linhas aumentam de número para cada novo registro inserido. As colunas possuem um nome que não se repete na relação e também um tipo (inteiro, *caracter*, numérico, etc.). Um conjunto de relações é um banco de dados.

O PostgreSQL foi o Sistema Gerenciador de Banco de Dados - SGBD escolhido para administrar o banco de dados do sistema. Ele é o mais avançado SGBD de código aberto, armazenando os dados de forma relacional e suas transações são ACID. O seu código fonte aberto é derivado da versão 4.2 do POSTGRES, desenvolvida no Departamento de Ciência da Computação da Universidade da Califórnia Berkeley. Ele suporta grande parte da linguagem de consulta SQL, padrão 2003, e recursos modernos tais como:

- Consultas complexas com o uso de SQL;
- Chaves estrangeiras para uma maior segurança quanto à integridade dos dados;
- Gatilhos que possibilitam a execução de código antes ou depois de determinados eventos no banco de dados;
- Armazenamento de consultas complexas na forma de visões;
- Transações que garantem que, caso uma operação não possa ser completada após o seu início, o banco de dados continue com a sua integridade preservada.

Além disso, o PostgreSQL pode ser melhorado pela da adição de novos recursos pelo usuário, tais como:

- Tipos de dados;
- Funções;
- Operadores;
- Agregados;
- Métodos de indexação;
- Linguagens procedurais.

Além de armazenar dados, todo SGBD precisa fornecer maneiras eficientes de acessar os dados armazenados. Hoje em dia, qualquer SGBD implementa a linguagem SQL e não poderia ser diferente com o PostgreSQL. SQL é uma linguagem estruturada para manipulação e definição de bancos de dados relacionais.

4.3.2 Sistema Operacional

Depois de escolhido o SGBD, foi escolhido o sistema operacional da máquina que funciona como servidor de banco de dados. O sistema escolhido foi o FreeBSD por possuir uma história de estabilidade de longa data, contar com diversas ferramentas de desenvolvimento, distribuídas junto com o sistema, tal como compilador C, e por ser (como todo sistema do tipo Unix) capaz de executar *scripts shell* e realizar tarefas em *batch*. As principais características do FreeBSD são as seguintes:

- Multitarefa preemptiva, com ajuste dinâmico de prioridade para permitir um compartilhamento eficiente do computador entre aplicações e usuários, mesmo sob grande quantidade de tarefas;
- Facilidades multi-usuário que permitem que muitos usuários usem o sistema FreeBSD, simultaneamente, para uma variedade de tarefas, significando, por exemplo, que periféricos, como impressoras e *drivers* de fita, podem ser

compartilhados entre todos os usuários no sistema e que limites de recursos individuais possam ser alocados para usuários ou grupos de usuários, protegendo o sistema contra esgotamento de recursos críticos;

- Pilha TCP/IP robusta, com suporte a padrões industriais como SLIP, PPP, NFS, DHCP e NIS, significando que o sistema FreeBSD pode trabalhar facilmente com outros sistemas operacionais, como também atuar como um servidor, provendo funções vitais como NFS (acesso remoto a arquivos), serviços de e-mail, ou atuando como um servidor da internet com serviços tais como www, FTP, roteamento e *firewall*;
- Proteção de memória garantindo que aplicações (ou usuários) não possam interferir com os demais, significando que um problema em uma aplicação não afeta outra de forma nenhuma;
- Sistema operacional de 32-bit, desde o seu projeto inicial;
- Utiliza o padrão *X Windows System* (X11R6) para prover uma interface gráfica com o usuário;
- Disponibilidade de ferramentas de desenvolvimento C, C++, Fortran, Perl, etc. *scripts shell* para processamento de comandos em batch.

O FreeBSD é baseado no código do 4.4BSD-Lite, disponibilizado pelo Computer System Research Group (CSRG), da Universidade da Califórnia Berkeley. O desenvolvimento do projeto FreeBSD, desde 1993, como um *patch* não oficial para o 386BSD, garante ao sistema muitas horas de testes e uso, sem contar o tempo de uso do código já herdado do 4.4BSD-Lite.

4.3.3 Interface

O SICO utiliza como principal interface com o usuário um *web site* dinâmico. O servidor *Web* utilizado para isso foi o *Apache Web Server*. O *Apache* foi escolhido tanto pela sua conhecida estabilidade, como também pela sua estrutura modular que permite que ele seja melhorado através de módulos escritos por terceiros. Utilizando-se esse recurso foi desenvolvido um módulo que adiciona capacidade ao *Apache* para interpretar *scripts* dentro de arquivos. São estes *scripts* que tornam os sites dinâmicos.

4.3.4 Interpretador de Scripts

P Hipertext Preprocessor-PHP é o interpretador de *scripts* estudado e utilizado no sistema. O PHP utiliza uma linguagem de programação interpretada, com sintaxe parecida com C, Java e Perl, e que é especialmente adequado para a criação de sites, pois pode ser mesclada dentro de arquivos html. Seu funcionamento é simples: o PHP é adicionado ao servidor Web como módulo e, quando um arquivo com código php é requisitado pelo cliente, o módulo PHP instalado no servidor processa o arquivo executando o código php dentro dele; o resultado é, então, enviado para o cliente.

O PHP terá duas funções no SICO: tornar as páginas dinâmicas e recuperar os dados do banco de dados. Além de fornecer as estruturas de controle comuns a todas as linguagens de programação, o PHP também fornece os recursos necessários para a conexão com o SGBD PostgreSQL (além de outros). É, por meio desse recurso, que os dados armazenados no banco de dados serão exibidos no *browser* do usuário. Além das páginas html, as páginas wml do

sistema também utilizam o PHP para que possam fornecer uma interface dinâmica e com acesso aos dados armazenados.

4.3.5 WAP

Foi também necessário estudar como implementar uma maneira dos colaboradores da empresa enviarem relatórios das atividades de campo. A tecnologia escolhida foi o *Wireless Application Protocol* - WAP, que "*pode ser definido como o conjunto de regras que governam a transmissão e recepção de dados entre aplicações em computadores e dispositivos sem fios como telefones celulares*" (BULBROOK, 2001: 2). O WAP é um protocolo de especificação aberta que permite aos usuários com dispositivos sem fio acessar informações e serviços instantaneamente. Além do protocolo para comunicação, há uma coleção de linguagens, ferramentas e uma infra-estrutura para implementação de serviços para dispositivos móveis. O protocolo foi projetado para trabalhar com muitos tipos de redes sem fio, como por exemplo: CDPD, CDMA, GSM, PDC, PHS, TDMA, FLEX, ReFLEX, iDEN, TETRA, DECT, DataTAC, Mobitex e GPRS. O WAP conta, também, com um ambiente de aplicação. Ele torna possível o uso de serviços similares aos da *Word Wide Web* e pode ser implementado em qualquer sistema operacional, incluindo PalmOS, EPOC, Windows CE, FLEXOS, OS/2, JavaOS, etc, provendo, também, interoperabilidade entre diferentes famílias de dispositivos.

O protocolo WAP utiliza a *Wireless Markup Language* –WML, uma nova linguagem de marcação; a WML, é uma linguagem de *script* WMLScript e possui um formato e tipo de bitmap denominado de *Wireless Bitmap* (ou WBMP). A utilização desses novos recursos foi necessária porque o protocolo http, junto com a linguagem html, é muito ineficiente para a transmissão e exibição de dados em dispositivos como telefones celulares.

O protocolo WAP reduz a sobrecarga de informações desnecessárias do HTTP para alguns poucos bytes por requisição, ao invés de uma centena de bytes. Embora o WAP utilize um novo protocolo de comunicação de dados, não é necessário aos desenvolvedores de conteúdo tratar com esse novo protocolo diretamente, pois é utilizado um *gateway* que "traduz" as informações vindas do servidor via HTTP para a pilha de protocolos WAP, antes que as informações cheguem ao dispositivo móvel. De modo análogo, o *gateway* "traduz" as requisições vindas do aparelho móvel para o protocolo http, antes de enviá-las ao servidor de conteúdo, como ilustrado na fig. 8.

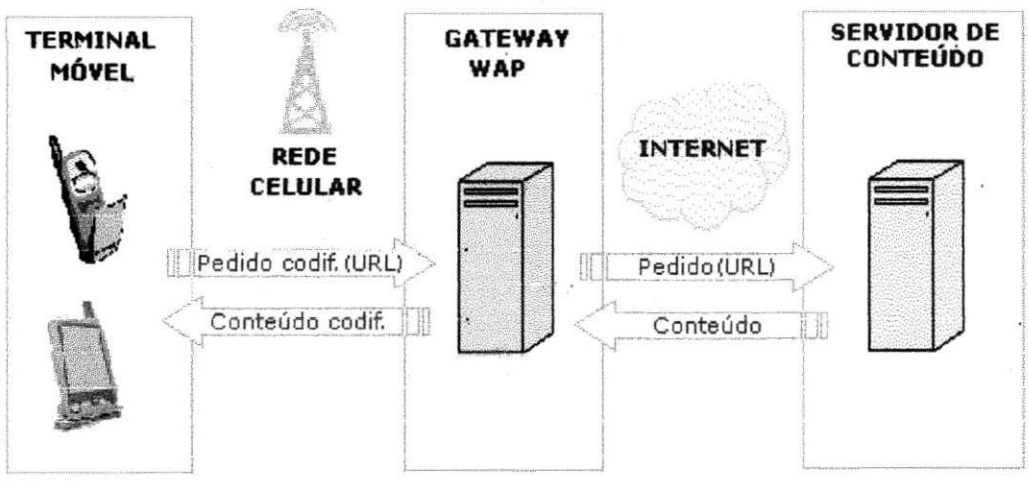


Figura 8 - Acesso WAP

Desta forma, pode-se escrever as páginas na forma de texto, utilizando os marcadores WML, e o *gateway* WAP encarrega-se de realizar as conversões necessárias antes de enviá-la ao dispositivo móvel.

4.3.6 ScriptCase

O ScriptCase foi a última ferramenta estudada antes do início da implementação do sistema da INORPEL. O ScriptCase é um *framework* para desenvolvimento de aplicações

para a Web. Seu uso é necessário porque o desenvolvimento de uma aplicação com interface Web pode ser um processo demorado e improdutivo se for feito sem o auxílio de alguma ferramenta especializada. O ScriptCase pode ser usado para desenvolver aplicativos diretamente para a Web, utilizando html e PHP, e com acesso nativo a banco de dados. O desenvolvedor tem, apenas, que informar alguns parâmetros como tabelas ou visões para acesso pela aplicação e também opções para configuração da aparência da aplicação produzida. As aplicações criadas com o ScriptCase não são capazes de implementar regras de negócios complexas, mas, no caso do sistema da INORPEL, essas foram implementadas utilizando gatilhos no banco de dados (código executado automaticamente quando determinados eventos acontecem).

4.4 Trabalhos Desenvolvidos

O trabalho realizado na INORPEL foi de desenvolvimento do sistema de controle das atividades. O sistema foi chamado de Sistema Integrado de Controle Operacional – SICO. O desenvolvimento foi, totalmente, realizado pelo estagiário, de acordo com as especificações da gerência. Na fig. 9 é apresentado o diagrama de entradas e saídas do SICO, sendo que as entradas são apresentadas do lado esquerdo e as saídas do lado direito. As entradas fornecem dados para o SICO, enquanto as saídas recebem dados. Os dados fornecidos pelo SICO podem ter sidos calculados de acordo com valores armazenados no banco de dados como, também, podem ser avisos de eventos ocorridos no sistema (como o recebimento de um relatório de uma equipe de campo).

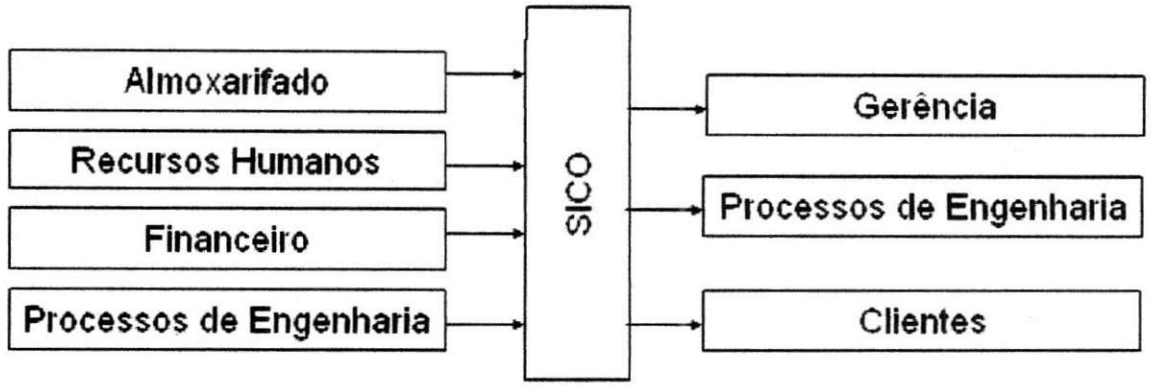


Figura 9 - Entradas (esquerda) e saídas (direita) de dados do SICO

Após a especificação dos recursos desejados pela gerência, foi criada a base de dados. A ferramenta utilizada para a manipulação da base de dados foi o phpPgAdmin.

O phpPgAdmin é a interface para PostgreSQL, escrita em PHP, cuja tela principal pode ser visualizada na fig. 10. Da mesma forma que a interface web desenvolvida para o SICO, o phpPgAdmin é um aplicativo criado para a Web. Isso permite a manutenção do banco de dados a partir de qualquer computador conectado à *intranet* da empresa ou à Internet, via VPN. Dessa forma, é possível manipular o banco de dados até mesmo de casa (como no momento em que este relatório foi escrito), sem a necessidade de instalar um aplicativo no PC.

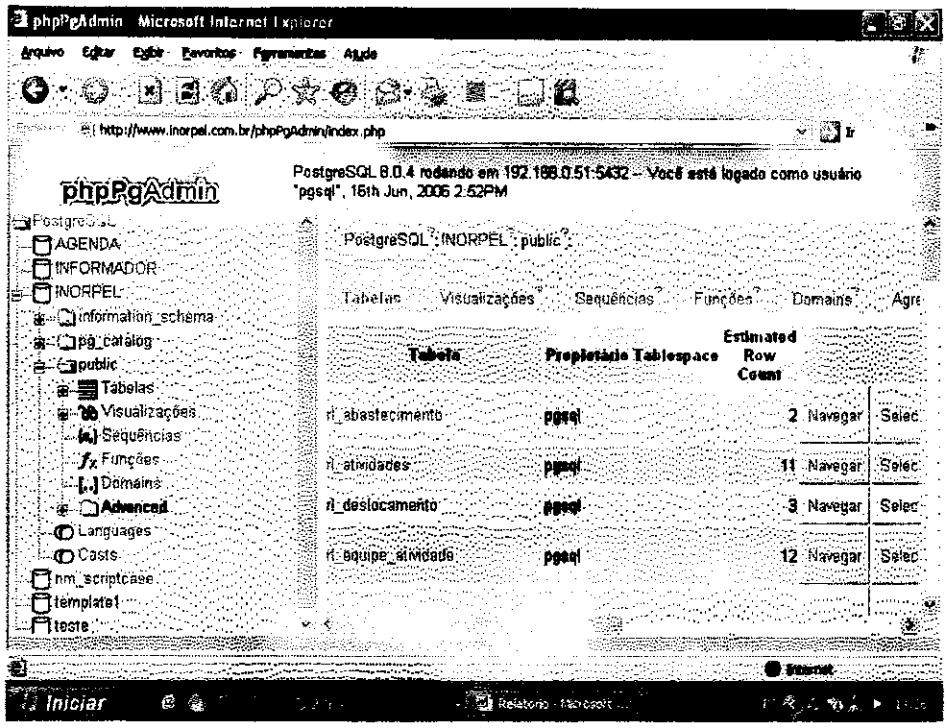


Figura 10 - Tela principal do phpPgAdmin

O phpPgAdmin permite a criação de tabelas facilmente. Um exemplo de seção para a criação de uma tabela é dado na fig. 11a e 11b.

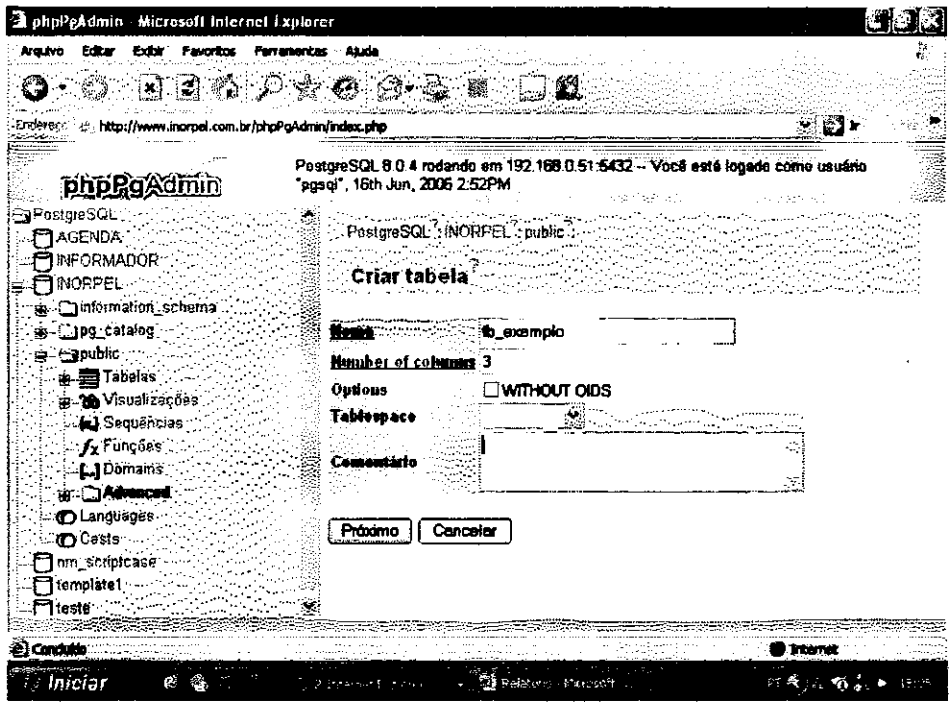


Figura 11a - Criação de uma tabela

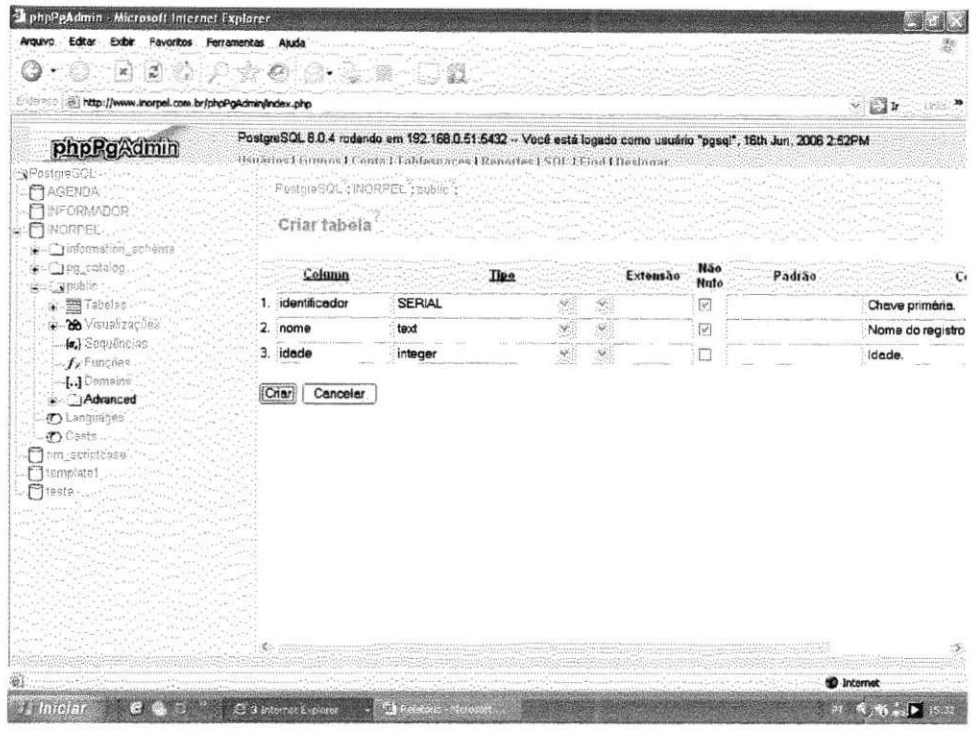


Figura 11b - Nome e tipo dos campos da tabela

Além de tabelas, o phpPgAdmin permite criar, modificar e remover outros objetos da base de dados, tais como:

- Usuários e grupos;
- Bancos de dados;
- *Schemas*;
- Tabelas, índices, restrições, gatilhos, regras e privilégios;
- Visões, seqüências e funções;
- Objetos avançados;
- Relatórios.

Para a criação da interface Web do SICO foi utilizado o ScriptCase, que é, da mesma forma que o phpPgAdmin, um aplicativo desenvolvido para a *Web*. Assim, compartilhando as mesmas facilidades do phpPgAdmin no que diz respeito à sua disponibilidade em qualquer computador da empresa (ou mesmo fora dela), sem a

necessidade de instalação do ScriptCase na máquina específica. A tela principal do ScriptCase pode ser visualizada na fig. 12.

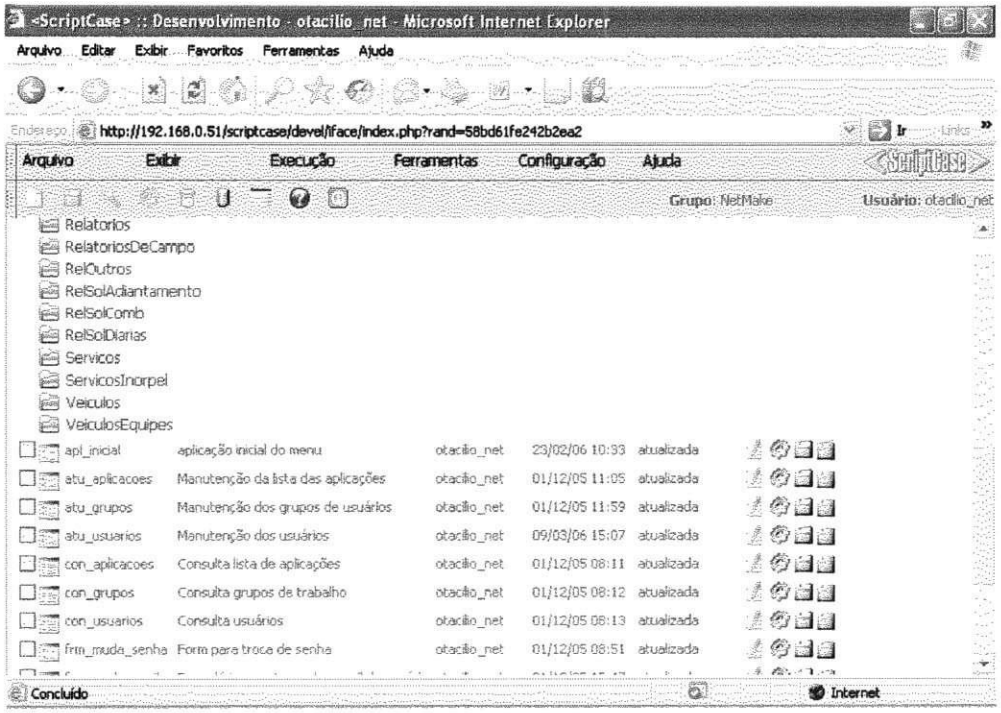


Figura 12 - Tela principal do ScriptCase

A criação de um aplicativo para a *Web,* utilizando o ScriptCase pode ser realizada facilmente, agilizando assim o desenvolvimento do SICO. Sem essa característica, a criação da interface seria demasiadamente demorada. Depois de criada, a aplicação pode ser ligada às outras, criando um conjunto de aplicações interligadas que, juntas, formam o SICO.

O criação das aplicações é realizado por meio de um processo passo a passo, onde o desenvolvedor escolhe que tipo de aplicação deseja criar e, em seguida, informa ao ScriptCase parâmetros da aplicação. Depois de criada, a aplicação pode ser modificada para a realização de ajustes, removida, etc.

Depois de criada a interface e o banco de dados do sistema, foi escrito um conjunto de funções em C, que permitem a implementação da interface entre o SICO e os recursos do sistema operacional.

Ao todo, um conjunto de 13 funções em C foram escritas para realizar a interface do PostgreSQL com o SO, (ver Anexo 1) e uma função adicional para o envio de e-mail de dentro de gatilhos (o PostgreSQL não possui, nativamente, nenhuma função para o envio de e-mails). Os procedimentos para a interface são praticamente idênticos, diferenciando-se apenas por causa do número de parâmetros. Eles são compilados como um módulo para ligação dinâmica com o processo do PostgreSQL. Depois de compilados, para cada função em C, é criada uma função dentro do banco de dados, que é a abstração utilizada pelo PostgreSQL para permitir aos usuários do banco de dados executar as funções escritas em C.

As funções C são instaladas com permissões de segurança de forma que apenas o administrador do banco de dados possa executá-las diretamente. Este cuidado é necessário para que usuários comuns não driblem os recursos de segurança do PostgreSQL e do sistema operacional, e executem comandos no sistema (com as permissões do usuário sob o qual está executando o PostgreSQL). Após as funções em C serem compiladas, instaladas, e criadas as funções dentro do banco de dados que realizam a interface com as funções C, foi escrito um *script* Bash para o envio de e-mails. O script recebe parâmetros das funções C que, por sua vez, recebem os parâmetros dos gatilhos que são disparados quando um novo registro é inserido no banco de dados. O fluxo dos dados desde a interface web até o e-mail pode ser visualizado na fig. 13.

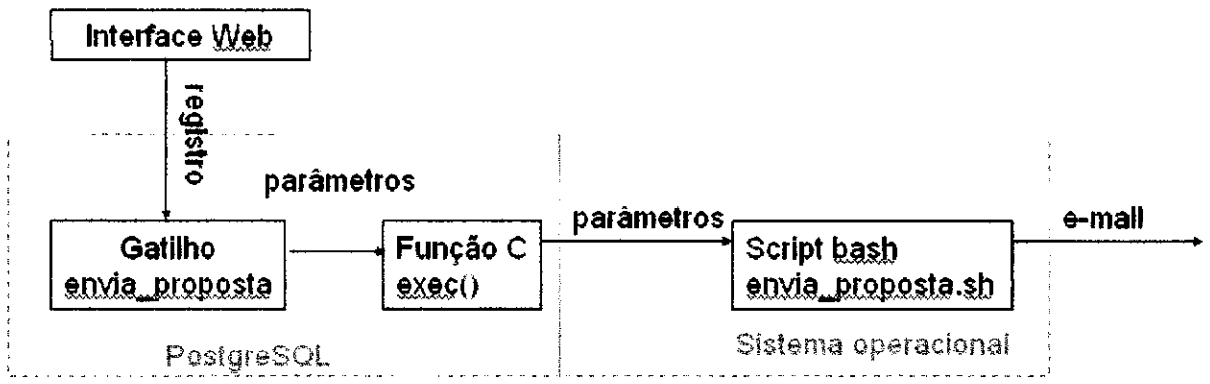


Figura 13 - Fluxo de dados quando o SICO envia um e-mail

Não só e-mails, mas qualquer comando que possa ser executado no *shell* do sistema operacional pode ser executado pelo do PostgreSQL por meio de funções em C (este recurso de executar comandos do sistema operacional por meio de funções do PostgreSQL foi utilizado também para a manipulação de arquivos no sistema de arquivos da máquina onde o SICO está instalado).

Foram criadas três interfaces para o usuário, sendo que cada uma delas destina-se a um usuário em particular. São elas:

- Interface *w*eb para uso na *intranet* da INORPEL;
- Interface WAP para acesso via telefones celulares;
- Interface via *web site* para uso pelos clientes da INORPEL.

4.5 Interface *web* para uso na *intranet*

A interface *web* para uso na *intranet* serve basicamente para a adição, edição e remoção dos registros do banco de dados. Todas as "regras de negócio" são implementadas como gatilhos, restrições e chaves (primárias e estrangeiras). Para acesso à interface é necessário, primeiramente, conectar-se por meio de login e senha, conforme pode ser visualizado na fig. 14a. Após conectar-se, o usuário tem acesso à tela com o menu principal do SICO, conforme pode ser visualizado na fig. 14b.

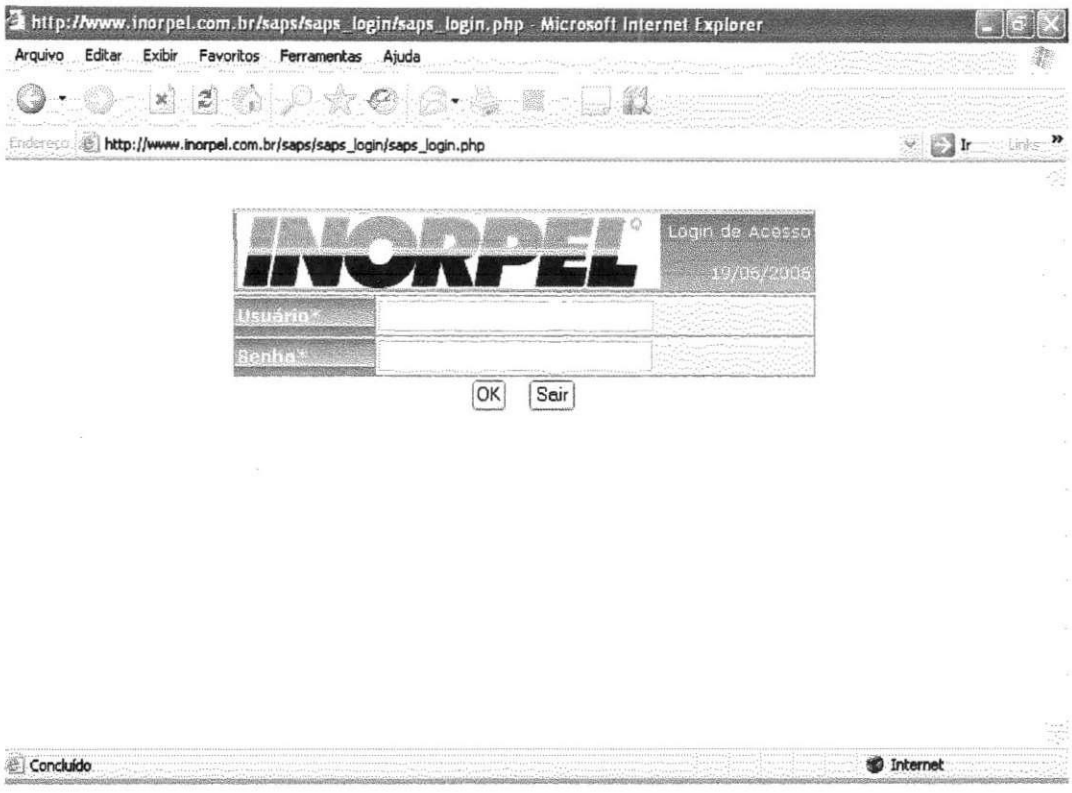


Figura 14a - Tela de login do SICO

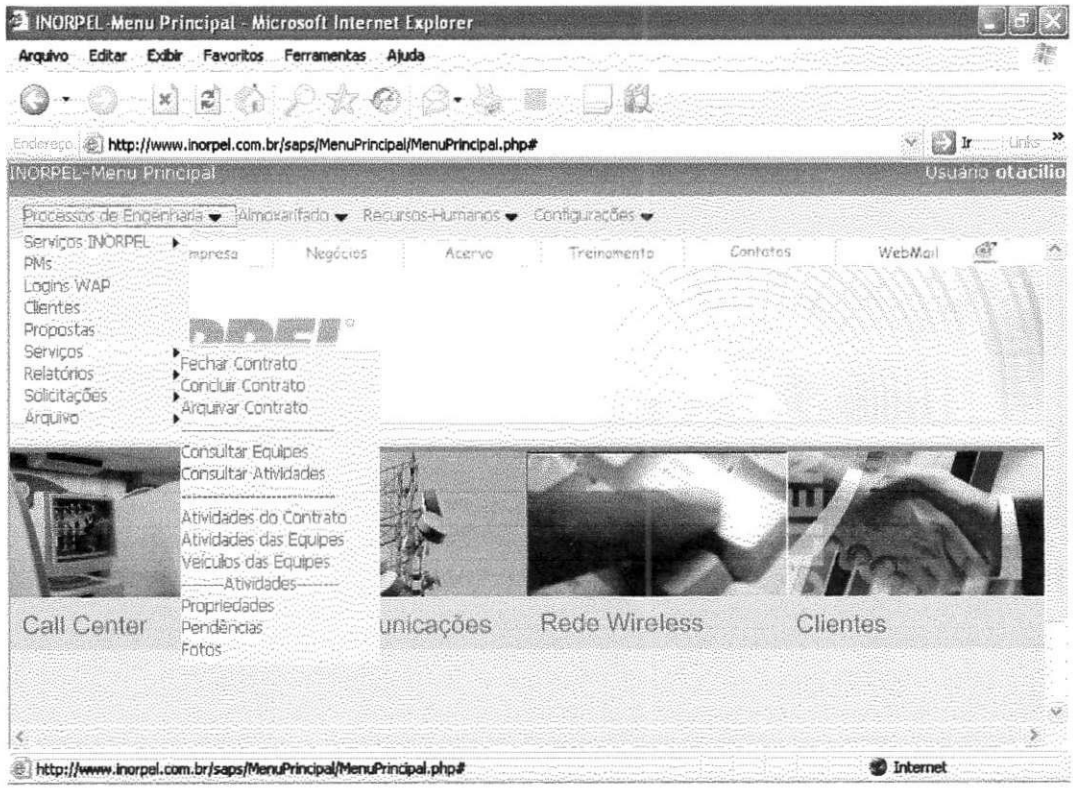


Figura 14b - Menu com as opções do SICO

Clicando em "Processos de Engenharia", "Propostas", a composição de custos pode ser realizada. Na fig. 15, por exemplo, é ilustrada a adição de um equipamento a um item de contrato.

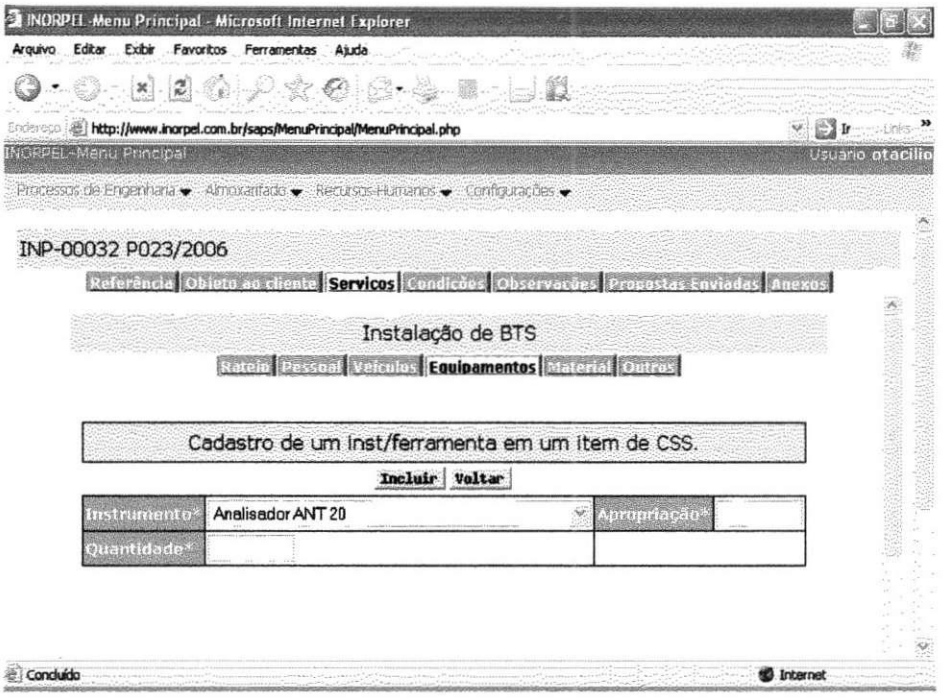


Figura 15 - Adicionando um equipamento a um item de contrato

4.6 Interface WAP

A interface *Wireless Application Protocol* - WAP permite aos funcionários da INORPEL enviar relatórios das atividades em campo através de dispositivos móveis (como telefones celulares), de qualquer lugar coberto pela rede de telefonia móvel, com suporte a GPRS. Da mesma forma que a interface *Web,* disponível na *intranet* da empresa, a interface WAP funciona como um *web site*, porém específico para dispositivos equipados com o *browser* WAP.

A interface WAP foi desenvolvida pela construção de um conjunto de arquivos texto tal como uma página html, sendo que o browser WAP disponível no aparelho celular interpreta o texto da página servido pelo *gateway* WAP e o exibe na tela, conforme pode ser

visualizado na fig. 16. Os arquivos correspondentes às telas apresentadas na fig. 16 podem ser consultados no Anexo 2.

Login e Senha *Validação* *Menu principal*

Figura 16 – Exemplos de telas da *interface* WAP

Quando um funcionário envia um relatório através do telefone celular, um gatilho, na tabela responsável por armazenar o relatório, envia um e-mail automaticamente para os funcionários responsáveis por receber os relatórios das equipes de campo. Por exemplo, o fluxo de informações correspondente à tabela rl_deslocamento é apresentado na fig. 17.

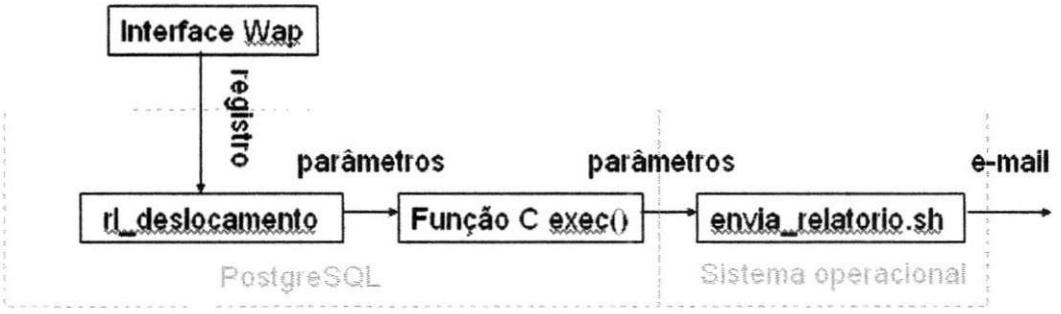


Figura 17 - Fluxo de dados ao receber um relatório

Várias tabelas possuem gatilhos que executam comandos do sistema operacional quando determinado evento ocorre no banco de dados. Isso garante ao sistema uma grande interação com programas externos ao banco de dados.

4.7 Interface via *web site*

A terceira interface do sistema é a interface via *web site,* que permite aos clientes consultar o andamento das atividades contratadas com a INORPEL. A interface via *web site* solicita, também, um login e senha para permitir o acesso do usuário. Depois de conectar-se, é apresentada ao usuário a tela da fig. 18. Após escolher um contrato em andamento, o cliente pode observar a situação das etapas daquele contrato. Como, por exemplo, na fig. 19.

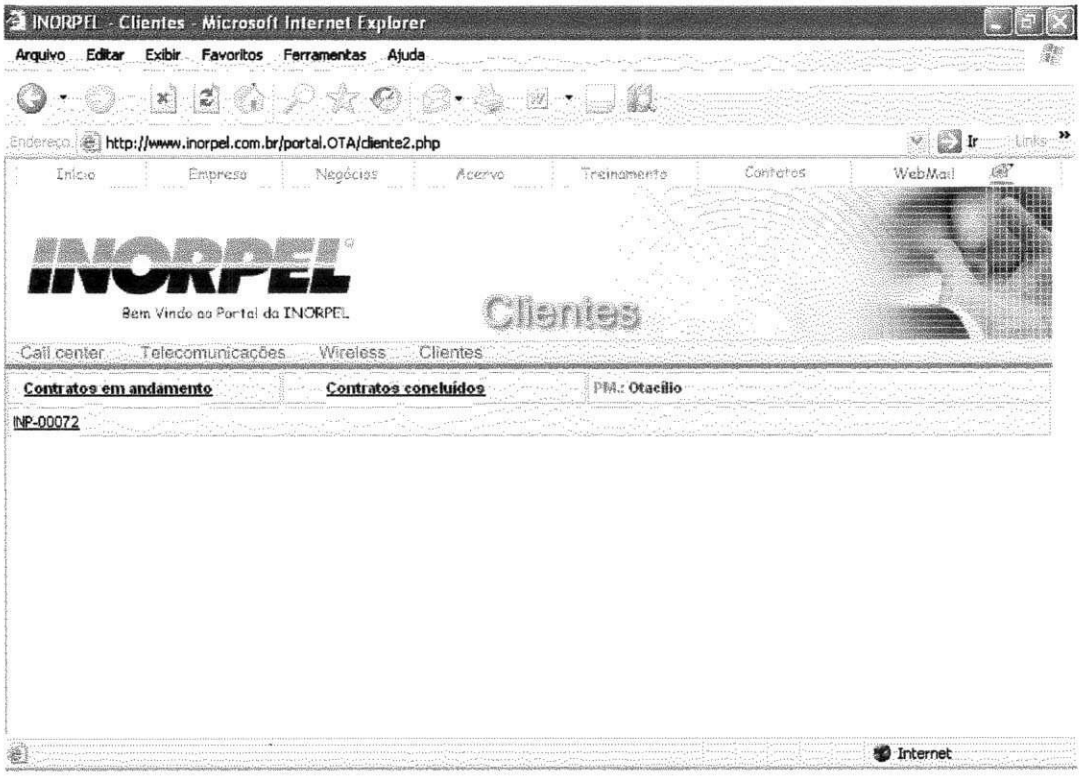


Figura 18 - Contratos em andamento do cliente

Os estados das atividades são atribuídos pelo sistema quando é recebido um relatório de campo. Ao receber o primeiro relatório de uma etapa de atividade, a data de início da execução é armazenada no sistema e a atividade passa a estar então "Em andamento". Ao receber o último relatório de campo da etapa da atividade, a data de conclusão é armazenada no sistema e a atividade passa a apresentar o estado "Concluído".

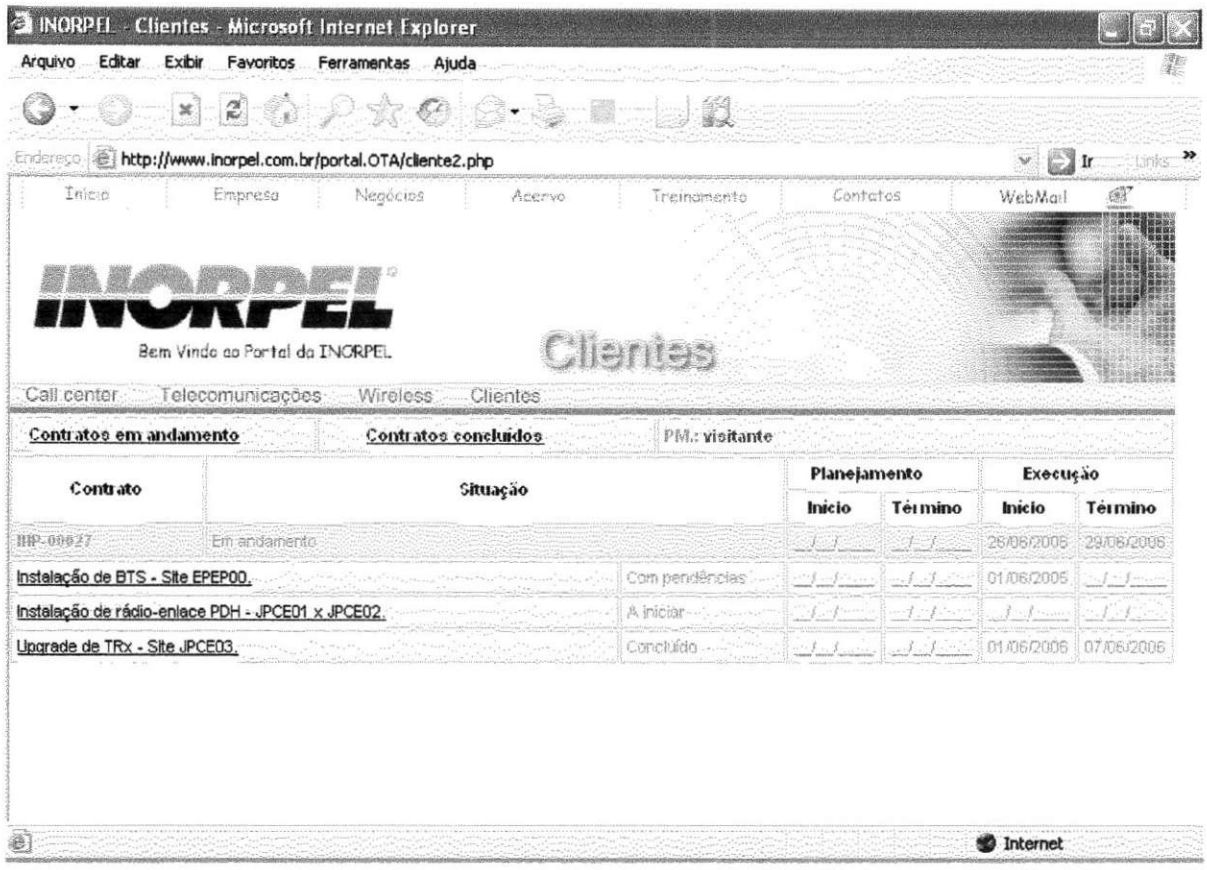


Figura 19 - Atividades de um serviço

Clicando em uma atividade é possível obter mais detalhes sobre ela e sobre o andamento das suas etapas. Clicando nas etapas é possível ver as fotos do local antes e depois da realização do trabalho (caso a etapa disponha de fotos). Por exemplo, na fig. 20 são exibidas as etapas da atividade "Upgrade de TRx – Site JPCE03".

Ao receber um relatório de campo, o SICO automaticamente atualiza, também, a interface disponível para o cliente, tornando possível que este acompanhe o andamento das atividades, em paralelo aos gestores da INORPEL sem a necessidade de que os gestores fiquem atualizando, constantemente, o *web site* da empresa. O SICO realiza essa tarefa, como informa ao gestor a recepção do novo relatório.

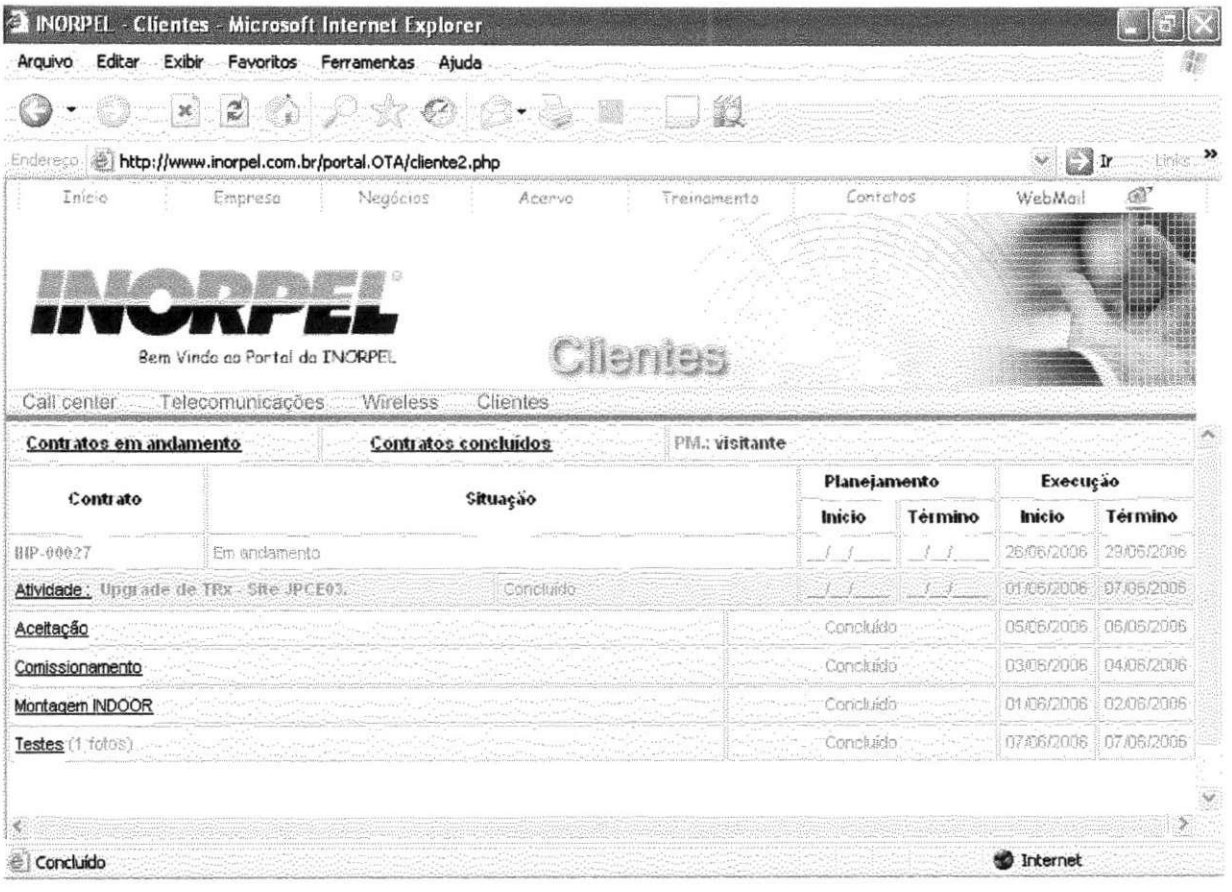


Figura 20 - Etapas das atividades

5. Resultados obtidos

Na época em que este relatório foi escrito, o SICO ainda não estava totalmente em operação, devido ao período de baixa nas atividades de prestação de serviços que sempre ocorre no início do ano. Mesmo assim, os testes iniciais, como também o próprio uso do sistema em algumas atividades reais de prestação de serviço, demonstrou que o sistema auxilia bem e de forma integrada a realização das atividades. Os recursos para arquivamento de anexos, composições de custos, contratos e arquivamento de propostas permitiram uma centralização no armazenamento dos arquivos recebidos, bem como a disponibilidade desses arquivos a todos os que podem ter acesso às informações, nesta fase do processo produtivo. Os recursos de comunicação integrados no SICO mostraram-se, também, muito valiosos. A facilidade de enviar relatórios, em qualquer lugar com cobertura GSM, permite aos técnicos enviar seus relatórios de maneira simples e padronizada, tornando a análise mais rápida e eficiente por parte dos responsáveis pelas equipes de campo.

Anexos

Anexo 1 - Funções em C

A seguir são apresentadas duas do conjunto de 13 funções C utilizadas para permitir a execução de comandos do sistema operacional por funções do PostgreSQL.

```
/*25/10/2005 Otacilio de Araujo Ramos Neto
  otacilio_neto@yahoo.com.br*/
/*Licenca:
  Voce pode fazer o que quiser com este software, exceto
  Remover esta licenca,
  dizer de qualquer forma que nao foi Otacilio de Araujo Ramos Neto que
  escreveu este software.
  Se voce utilizar ou distribuir este software voce automaticamente esta
  concordando com os termos desta licenca.

  Este software eh distribuido acreditando-se que ele eh totalmente funcional,
  mas sem nenhuma garantia implicita ou explicita disto, qualquer dano de
  qualquer tipo deve ser assumido pelo usuario deste software.
 */

#include "postgres.h"
#include <string.h>
#include "fmgr.h"

#include <sys/types.h>
#include <unistd.h>
#include <sys/wait.h>

#include <sys/time.h>
#include <sys/resource.h>
#include <stdlib.h>

#include <syslog.h>
#include <stdarg.h>

extern char **environ;

int erro;

PG_FUNCTION_INFO_V1(exec1);

Datum exec1(PG_FUNCTION_ARGS){

pid_t child; /*Pid do processo filho criado*/
int status;  /*status de terminacao do processo filho*/

/*Argumentos passados a funcao*/

text*param0 = PG_GETARG_TEXT_P(0);

/*Variaveis das strings de comando*/

char *args[2];
```

```
erro = 0;

/*+1 para o caracter de nulo*/
args[0] = (char *) palloc(VARSIZE(param0)+1-VARHDRSZ);
memcpy(args[0],VARDATA(param0),VARSIZE(param0)-VARHDRSZ);

args[0][VARSIZE(param0)-VARHDRSZ]='\0';
args[1]=NULL;

/*Fork em um novo processo*/
if(!(child=vfork())){
/*O processo filho entra aqui*/

/*Substitui a imagem de memoria do processo*/
syslog(LOG_INFO,"PostgreSQL executando o comando %s\n",args[0]);
if(execve(args[0],args,environ)==-1){
/*Nao foi possivel substituir a imagem do processo*/
syslog(LOG_ERR,"Não foi possivel executar %s\n",args[0]);
exit(-1);
}

}else if(child!=-1){

/*O pai em casos normais entra aqui*/
if(wait(&status)!=child){
/*Houve algum problema no processo filho*/
syslog(LOG_ERR,"Pid retornado pelo processo filho nao corresponde ao esperado pelo
pai. Mensagem da funcao exec do PostgreSQL\n");
erro = -1;

}else if(!WIFEXITED(status)){
/*Processo filho nao terminou normalmente*/
syslog(LOG_ERR,"Comando executado nao terminou normalmente. Mensagem da funcao
exec do PostgreSQL\n");
erro = -1;
}

}else{
/*O pai caso nao tenha conseguido dar um fork entra aqui*/
syslog(LOG_ERR,"Funcao exec do PostgreSQL nao conseguiu dar um fork\n");
erro = -1;
}

pfree(args[0]);

if(erro){
/*Eh, tivemos problemas na execucao do processo filho.
Observe as mensagens do syslog para ver o que pode ter ocorrido.*/
PG_RETURN_INT32(-1);
}

/* O processo filho terminou normalmente. Vamos
retornar o valor de retorno dele.*/

PG_RETURN_INT32(WEXITSTATUS(status));
}
```

```
PG_FUNCTION_INFO_V1(exec13);

Datum exec13(PG_FUNCTION_ARGS){

pid_t child; /*Pid do processo filho criado*/
int status;  /*status de terminacao do processo filho*/

/*Argumentos passados a funcao*/

text*param0 = PG_GETARG_TEXT_P(0);
text*param1 = PG_GETARG_TEXT_P(1);
text   *param2 = PG_GETARG_TEXT_P(2);
text*param3 = PG_GETARG_TEXT_P(3);
text*param4= PG_GETARG_TEXT_P(4);
text*param5 = PG_GETARG_TEXT_P(5);
text   *param6 = PG_GETARG_TEXT_P(6);
text   *param7 = PG_GETARG_TEXT_P(7);
text   *param8 = PG_GETARG_TEXT_P(8);
text   *param9 = PG_GETARG_TEXT_P(9);
text   *param10 = PG_GETARG_TEXT_P(10);
text   *param11 = PG_GETARG_TEXT_P(11);
text   *param12 = PG_GETARG_TEXT_P(12);

/*Variaveis das strings de comando*/


char *args[14];

erro = 0;

/*+1 para o caracter de nulo*/

args[0] = (char *) palloc(VARSIZE(param0)+1-VARHDRSZ);
args[1] = (char *) palloc(VARSIZE(param1)+1-VARHDRSZ);
args[2] = (char *) palloc(VARSIZE(param2)+1-VARHDRSZ);
args[3] = (char *) palloc(VARSIZE(param3)+1-VARHDRSZ);
args[4] = (char *) palloc(VARSIZE(param4)+1-VARHDRSZ);
args[5] = (char *) palloc(VARSIZE(param5)+1-VARHDRSZ);
args[6] = (char *) palloc(VARSIZE(param6)+1-VARHDRSZ);
args[7] = (char *) palloc(VARSIZE(param7)+1-VARHDRSZ);
args[8] = (char *) palloc(VARSIZE(param8)+1-VARHDRSZ);
args[9] = (char *) palloc(VARSIZE(param9)+1-VARHDRSZ);
args[10] = (char *) palloc(VARSIZE(param10)+1-VARHDRSZ);
args[11] = (char *) palloc(VARSIZE(param11)+1-VARHDRSZ);
args[12] = (char *) palloc(VARSIZE(param12)+1-VARHDRSZ);


memcpy(args[0],VARDATA(param0),VARSIZE(param0)-VARHDRSZ);
memcpy(args[1],VARDATA(param1),VARSIZE(param1)-VARHDRSZ);
memcpy(args[2],VARDATA(param2),VARSIZE(param2)-VARHDRSZ);
memcpy(args[3],VARDATA(param3),VARSIZE(param3)-VARHDRSZ);
memcpy(args[4],VARDATA(param4),VARSIZE(param4)-VARHDRSZ);
memcpy(args[5],VARDATA(param5),VARSIZE(param5)-VARHDRSZ);
memcpy(args[6],VARDATA(param6),VARSIZE(param6)-VARHDRSZ);
```

```
memcpy(args[7],VARDATA(param7),VARSIZE(param7)-VARHDRSZ);
memcpy(args[8],VARDATA(param8),VARSIZE(param8)-VARHDRSZ);
memcpy(args[9],VARDATA(param9),VARSIZE(param9)-VARHDRSZ);
memcpy(args[10],VARDATA(param10),VARSIZE(param10)-VARHDRSZ);
memcpy(args[11],VARDATA(param11),VARSIZE(param11)-VARHDRSZ);
memcpy(args[12],VARDATA(param12),VARSIZE(param12)-VARHDRSZ);


args[0][VARSIZE(param0)-VARHDRSZ]='\0';
args[1][VARSIZE(param1)-VARHDRSZ]='\0';
args[2][VARSIZE(param2)-VARHDRSZ]='\0';
args[3][VARSIZE(param3)-VARHDRSZ]='\0';
args[4][VARSIZE(param4)-VARHDRSZ]='\0';
args[5][VARSIZE(param5)-VARHDRSZ]='\0';
args[6][VARSIZE(param6)-VARHDRSZ]='\0';
args[7][VARSIZE(param7)-VARHDRSZ]='\0';
args[8][VARSIZE(param8)-VARHDRSZ]='\0';
args[9][VARSIZE(param9)-VARHDRSZ]='\0';
args[10][VARSIZE(param10)-VARHDRSZ]='\0';
args[11][VARSIZE(param11)-VARHDRSZ]='\0';
args[12][VARSIZE(param12)-VARHDRSZ]='\0';
args[13]=NULL;

/*Fork em um novo processo*/
if(!(child=vfork())){
/*O processo filho entra aqui*/

/*Substitui a imagem de memoria do processo*/
syslog(LOG_INFO,"PostgreSQL executando o comando %s %s %s %s %s %s %s %s %s
%s %s %s
%s\n",args[0],args[1],args[2],args[3],args[4],args[5],args[6],args[7],args[8],args[9],ar
gs[10],args[11],args[12]);
if(execve(args[0],args,environ)==-1){
/*Nao foi possivel substituir a imagem do processo*/
syslog(LOG_ERR,"Não foi possivel executar %s %s %s %s %s %s %s %s %s %s %s %s
%s\n",args[0],args[1],args[2],args[3],args[4],args[5],args[6],args[7], args[8], args[9],
args[10],args[11],args[12]);
exit(-1);
}

}else if(child!=-1){

/*O pai em casos normais entra aqui*/
if(wait(&status)!=child){
/*Houve algum problema no processo filho*/
syslog(LOG_ERR,"Pid retornado pelo processo filho nao corresponde ao esperado pelo
pai. Mensagem da funcao exec do PostgreSQL\n");
erro = -1;

}else if(!WIFEXITED(status)){
/*Processo filho nao terminou normalmente*/
syslog(LOG_ERR,"Comando executado nao terminou normalmente. Mensagem da funcao
exec do PostgreSQL\n");
erro = -1;
}

}else{
```

```
/*O pai caso nao tenha conseguido dar um fork entra aqui*/
syslog(LOG_ERR,"Funcao exec do PostgreSQL nao conseguiu dar um fork\n");
erro = -1;
}


pfree(args[0]);
pfree(args[1]);
pfree(args[2]);
pfree(args[3]);
pfree(args[4]);
pfree(args[5]);
pfree(args[6]);
pfree(args[7]);
pfree(args[8]);
pfree(args[9]);
pfree(args[10]);
pfree(args[11]);
pfree(args[12]);

if(erro){
/*Eh, tivemos problemas na execucao do processo filho.
Observe as mensagens do syslog para ver o que pode ter ocorrido.*/
PG_RETURN_INT32(-1);
}

/*O processo filho terminou normalmente. Vamos
retornar o valor de retorno dele.*/

PG_RETURN_INT32(WEXITSTATUS(status));
                                                }
```