**MARCIA DE ALMEIDA ABADE**

# INTEGRAÇÃO DE BASES DE AUTENTICAÇÃO LDAP COM BASES CORPORATIVAS (POSTGRESQL)

Monografia apresentada ao Curso de Administração em Redes Linux (ARL) da Universidade Federal de Lavras como parte das exigências da disciplina Monografia para obtenção do título de Especialista em Administração em Redes Linux.

Orientador
Professor Joaquim Quinteto Uchôa

**LAVRAS**

**MINAS GERAIS – BRASIL**

**2004**

**MARCIA DE ALMEIDA ABADE**

# INTEGRAÇÃO DE BASES DE AUTENTICAÇÃO LDAP COM BASES CORPORATIVAS (POSTGRESQL)

Monografia apresentada ao Curso de Administração em Redes Linux (ARL) da Universidade Federal de Lavras como parte das exigências da disciplina Monografia para obtenção do título de Especialista em Administração em Redes Linux.

Aprovada em 24 de Junho de 2004.

_____
Prof.

_____
Prof.

_____
Prof. Joaquim Quinteto Uchôa
(Orientador)

**LAVRAS**
**MINAS GERAIS – BRASIL**

# Sumário

# Lista de Figuras

# Lista de tabelas

Ao meu colega de trabalho e de estudo, Maurício Heredia, que me deu apoio e força, essenciais ao término deste curso na UFLA.

# Agradecimentos

Agradeço à minha amiga e chefe Cecília Cerapião, que me incentivou e permitiu que eu progredisse em mais um passo na minha carreira profissioal.
Agradeço aos meus familiares e amigos pelo apoio e paciência.

# Resumo

Este projeto visa promover a segurança dos recursos disponibilizados em uma rede corporativa através da administração rígida das contas de usuários.

Com esse foco, a confiabilidade nas informações contidas na base de autenticação depende de sua total compatibilidade com as informações reais da atual situação de cada usuário da rede.

Neste intuito, serão apresentadas técnicas para a atualização do conteúdo das bases de autenticação através da sua integração automática com as bases de informações corporativas.

# Capítulo 1

# Introdução

A tarefa de administrar uma rede de comunicação envolve um grande número de atividades e, sem dúvida, um dos grandes desafios é a manutenção da segurança do acesso às informações e serviços que geralmente estão disponíveis em uma rede corporativa [Paulo L.Geus e Emilio T. Nakamura (2003)].

O primeiro, e talvez mais importante, passo na direção de uma administração eficiente é um controle preciso sobre os usuários autorizados a utilizarem a rede e, conseqüentemente, a acessar os recursos provenientes nesta tecnologia. Esse controle pode ser feito de inúmeras formas e depende muito da estrutura, da arquitetura da rede e da organização administrativa da empresa em questão [Jose Carlos Cordeiro Martins (2003)].

O fato é que não é raro encontrar áreas, departamentos ou até gerências responsáveis por gerir informações sobre os funcionários da empresa, entretanto é raro observar uma integração dessa área com aquela responsável pela administração do controle de acesso aos recursos computacionais da empresa.

Geralmente a área de Recursos Humanos (RH) não tem o hábito de se integrar com a área de Tecnologia da Informação (TI) pois não percebe a relação entre suas atividades. O resultado disso pode ser catastrófico.

Em empresas grandes, funcionários admitidos só obtém acesso aos recursos computacionais da empresa após solicitação por escrito do seu

superior, enquanto que funcionários demitidos continuam com acesso a esses recursos até que uma comunicação oficial ocorra informando o seu desligamento. Além disso muitos outros recursos de administração podem ficar comprometidos pela burocracia. Portais construídos através da utilização de perfis personalizados podem ficar desatualizados com uma simples alteração nas siglas de departamentos, centros de custos, transferências entre áreas, andares e etc.

A situação da área de TI pode ser constrangedora. Apesar da maioria dos processos de atualização cadastral dos funcionários contar com sistemas informatizados de RH, os dados não são utilizados de maneira integrada, possibilitando situações incompatíveis. Pode ocorrer, por exemplo, de um funcionário recém transferido, que deveria ter acesso aos dados da contabilidade, continuar com acesso aos dados de orçamento (seu departamento anterior) até que o uma comunicação formal do superior solicite atualização do perfil do funcionário na área de TI.

Como solução para esses tipos de problemas, este projeto propõe um método de integração de informações entre as bases corporativas e as bases de autenticação. São chamadas bases corporativas aquelas alimentadas pelos sistemas de informação (Folha de Pagamento, Controle de Recursos Humanos, etc) [Fernando Jose Barbin Laurindo (2002)]. E são chamadas bases de autenticação aquelas utilizadas para a manutenção de usuários e seus perfis de acesso à rede corporativa [Jose Carlos Cordeiro Martins (2003)].

Este trabalho irá propor uma solução considerando produtos de código aberto que comportem as funções de base corporativa (PostgreSQL) e de base de autenticação (LDAP). Esses produtos serão integrados através do desenvolvimento de pequenas rotinas em linguagem de programação Perl, que por sua vez, estarão acopladas aos produtos de tal forma a se apresentar como uma interface entre as bases, acionadas automaticamente a cada manipulação

dos dados na base corporativa.

Com essa solução, atualizações em dados armazenados na base corporativa como transferências, demissões, admissões e etc, que tenham correspondência com dados da base de autenticação, implicarão em uma equalização automática das bases.

# Capítulo 2

# Bases Corporativas e Bases de Autenticação

Para o melhor entendimento do projeto, algumas definições importantes são descritas neste capítulo. Tais conceitos são o resultado da utilização prática da arquitetura de dados organizacional assim como dos produtos mencionados.

## 2.1 - Base de dados corporativa

### 2.1.1 - Definição

Bases de dados corporativas são conjuntos de dados e informações de propriedade de uma entidade jurídica, organizadas conforme uma arquitetura de dados.

Essas informações podem estar relacionadas com um número variável de assuntos e funções, e muitas vezes pode ser descrita como a base de conhecimento da organização [Bernard H. Boar (2003)]. Sua importância é cada vez mais vital para o desempenho comercial da empresa e, como conseqüência disso, são cada vez mais complexas as técnicas e tecnologias empregadas na

busca da otimização da armazenagem, acesso e segurança desses dados [Fernando Jose Barbin Laurindo (2002)].

Quanto mais organizada e melhor administrada essa base, maiores são as chances de sucesso dessa organização no mercado competitivo, uma vez que atualmente, a produtividade é resultado direto de eficiência na comunicação, integração e otimização dos recursos humanos e tecnológicos [Fernando Jose Barbin Laurindo (2002)].

### 2.1.2 - Organização

Existem muitas técnicas de organização de bases de dados. Essas técnicas, uma vez definidas e implantadas em uma organização dão origem ao que se chama *Arquitetura de Dados* . Os "arquitetos de dados" são técnicos especializados, responsáveis por modelar, gerir e manter métodos e padrões para o armazenamento, integração e utilização das informações corporativas [David M. Kroenke (1999)] . Esses padrões devem ser aderentes à realidade e à necessidade geral da empresa.

Criar esses métodos e padrões pode se tornar uma tarefa árdua sem o apoio da cúpula da organização. O alto escalão deve estar ciente da importância dessa administração, pois quanto mais "sob controle" esses padrões e métodos mantiverem as informações corporativas, mais facilmente elas poderão ser utilizadas de forma otimizada, segura e eficiente.

Exemplos de métodos e padrões podem ser:

- adoção da centralização dos dados em Sistemas Gerenciadores de Bancos de Dados (SGBD) ou a distribuição de informações por área de negócios;
- padrões de autenticação;
- processos de autorização de acessos;
- utilização de perfis;

- construção de portais corporativos;
- intranet;
- metodologias de desenvolvimento ou adoção de pacotes de mercado;
- documentação e dicionarização de dados, etc.

Neste projeto será considerado um modelo clássico de arquitetura de dados que prevê bases de dados centralizadas hospedando as informações vitais da corporação, como os dados de funcionários. Essa base central servirá de fonte de informações de qualidade para os demais sistemas e aplicações de integração.

### 2.1.3 - Conteúdo

Existe uma grande discussão sobre o que são dados corporativos. A linha mais coerente com o mercado atual e a chamada "era da informação" é aquela que considera: dados corporativos são quaisquer dados ou informações gerados dentro do ambiente organizacional, sejam eles estruturados, consistidos, validados ou não [Bernard H. Boar (2003)].

Essa massa de dados pode incluir desde bases estruturadas como cadastros de funcionários, sistemas de folha de pagamento, controles de estoque, até dados avulsos como planilhas departamentais, memorandos, normas, cartões de ponto e etc.

Para a área de informática isso pode significar uma miscelânea de formatos, linguagens e tecnologias. Aqui entra novamente a importância do arquiteto de dados, que terá que organizar, consistir, integrar e disponibilizar uma massa de informações de origem bastante diversificada, de modo a produzir informações confiáveis como resultado.

## 2.2- Base de dados para autenticação

### 2.2.1 - Definição

Autenticar significa reconhecer como válido. Por exemplo, quando se vai a um cartório autenticar um documento, o processo se resume a uma comparação da cópia com o original e um reconhecimento do mesmo como uma cópia válida. Se simplificarmos bem essa definição, podemos imaginar uma base de dados de autenticação como o conjunto de informações necessárias para a validação de um dado duvidoso. No exemplo do cartório, a base de dados para autenticação seria um cadastro de assinaturas ou a imagem do documento original.

No caso da autenticação em redes, esse processo diz respeito à validação do usuário como apto para o acesso aos recursos disponibilizados [Paulo L.Geus e Emilio T. Nakamura (2003)]. Dessa forma, a base para autenticação precisa conter, no mínimo, informações que permitam a identificação dos usuários existentes. Informações adicionais podem ser úteis para o tratamento de quais recursos estarão disponíveis para cada usuário identificado. Por exemplo, pode haver nesta base uma relação de aplicações classificadas por função e relacionadas a funcionários de determinadas áreas da empresa. Nesse caso é possível armazenar: o identificador e a área de atuação do funcionário em uma estrutura e em outra as aplicações disponíveis para aquela área de atuação. Dessa forma, além do usuário ser autenticado, apenas as aplicações relacionadas a sua área de atuação estariam disponíveis.Esse é o princípio da construção de portais corporativos.

### 2.2.2 - Organização

As bases de dados de autenticação possuem, normalmente, arquitetura hierárquica [Brian Arkills (2003)]. As informações são armazenadas segundo uma lógica de níveis e subníveis. A organização das informações deve levar em consideração as relações hierárquicas entre os dados armazenados.

Uma base de dados para autenticação pode ser organizada de várias formas, inclusive com o emprego de várias tecnologias. Neste projeto será considerada a utilização de uma ferramenta de autenticação (Lightweight Directory Access Protocol - LDAP) aliado a uma base de dados (repositório LBD, BDB, etc) [OpenLDAP Foundation (2004)]. Mas o conceito básico a ser aplicado em uma base de autenticação é a minimização da quantidade de informações e a simplificação da estrutura de armazenamento.

A organização dessas bases normalmente fica a cargo do administrador de redes e não de um arquiteto de dados e, em muitos casos, acaba por se tornar caótica pois passa a conter informações duplicadas às bases corporativas e estruturadas de uma forma não otimizada.

A melhor saída é a manutenção apenas da lista de usuários existentes e, no máximo, a utilização de estruturas hierárquicas simples para alguns níveis de agrupamentos como área de negócios ou grupos de recursos.

Isso quer dizer que a organização das informações exigidas por uma base de autenticação é simples e, na maioria das vezes, não exige o investimento em muitos recursos como um SGBD relacional, espaço em disco, processamento, etc.

### 2.2.3 - Conteúdo

Como já visto, o volume de informações em uma base de autenticação deve ser o menor possível. O administrador deve se lembrar sempre que uma base complexa requer uma administração custosa e a adoção de uma série de tecnologias para manter a estrutura confiável como paralelismo de servidores, disponibilidade, backup, etc. Quanto maior a base, maior o consumo de recursos. Além disso, o administrador de rede não pode perder o foco no objetivo dessa base. Esse não é o local indicado para armazenar dados pessoais de cada funcionário e nem o seu demonstrativo mensal. O objetivo da base de autenticação deve ser única e exclusivamente segurança.

Sendo assim, ao estruturar o conteúdo dessa base, o administrador de rede estará estabelecendo também uma metodologia para atualização e consulta dos dados. A estrutura da empresa e os níveis de segurança esperados devem ser analisados para uma melhor visão da modelagem e do conteúdo da base de autenticação.

A tabela 2.1 mostra um exemplo simples de estrutura para uma base de autenticação. A modelagem foi feita baseada na estrutura hierárquica (ou organograma) da empresa.

**Tabela 2.1 –** Exemplo de estrutura para base de autenticação

| **Presidência** | | | | | |
|---|---|---|---|---|---|
| **Recursos Humanos** | | **Serviços Gerais** | | **Produção** | |
| Benefícios | Pessoal | Montagem | Informática | Operação | Manutenção |
| funcionario01 | funcionario03 | funcionario06 | funcionario09 | funcionario11 | funcionario14 |
| funcionario02 | funcionario04 | funcionario07 | funcionario10 | funcionario12 | funcionario15 |
| | funcionario05 | funcionario08 | | funcionario13 | funcionario16 |

# Capítulo 3

# Produtos

Uma vez compreendidos os conceitos expostos no capítulo 2, serão apresentados neste capítulo produtos disponíveis no mercado de software livre que serão empregados neste projeto. São produtos que servirão para desempenhar as funcionalidades de bases de autenticação e de bases corporativas.

## 3.1- PostgreSQL

### 3.1.1 - Definição

O PostgreSQL é um Sistema Gerenciador de Banco de Dados (SGBD) objeto-relacional. Traz todas as características de um SGBD essencialmente relacional (integridade referencial, tabelas, visões, gatilhos e etc), acrescido de funcionalidades de caráter de orientação a objetos (herança, poliformismo e outras) [PostgreSQL Global Development Group (2003)].

### 3.1.2 - Aplicação

Como todo Sistema Gerenciador de Banco de Dados, o objetivo básico do PostgreSQL é o armazenamento de dados de modo organizado e gerenciável, permitindo a guarda e o resgate de dados e informações de maneira rápida, segura e eficiente.

O PostgreSQL é um produto de código aberto, desenvolvido pela Universidade de Berkley e hoje mantido por uma grande comunidade de desenvolvedores espalhada pelo mundo. Entre os produtos dessa linha, possui grande vantagem técnica e funcional sobre seus "concorrentes" e está ganhando uma boa fatia do mercado corporativa e concorrendo, inclusive, com produtos comerciais como ORACLE e SQLServer [Álvaro Pereira Neto (2003)].

Casos de uso como Shannon Medical Center, Mohawk Software, Vanten Inc, e BASF, disponíveis em [PostgreSQL Global Development Group (2003)] mostram que grandes empresas já se convenceram da robustez e qualidade do PostgreSQL. Elas estão colocando em suas estruturas grandes bases corporativas PostgreSQL, que abastecem aplicações críticas com necessidade de alta disponibilidade, segurança e performance.

### 3.1.3 - Funcionamento básico

A estrutura de armazenamento do PostgreSQL segue o conceito relacional: tabelas de n colunas por n linhas que se relacionam entre si através de colunas comuns [David M. Kroenke (1999)].

A figura 3.1 mostra uma representação gráfica do conceito relacional:

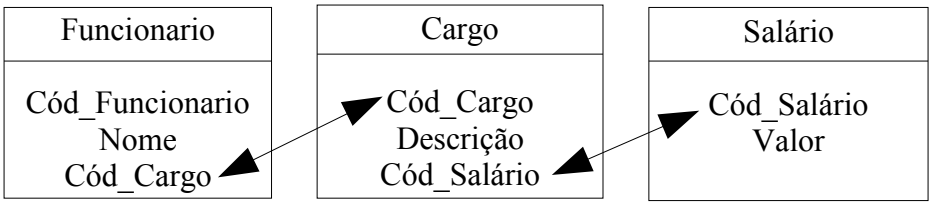


**Figura 3.1 –** Representação do conceito relacional

A organização das informações em tabelas é feita através de análises complexas do tipo e do comportamento dos dados a serem armazenados. Existe uma série de técnicas para a modelagem das informações em estruturas relacionais que não serão exploradas neste trabalho, mas que devem ser observadas sempre que um banco de dados for construído [Felipe Nery R. Machado (1996)]. Caso a modelagem não seja levada a sério, existe uma grande possibilidade de haver duplicidade e/ou incompatibilidade de informações dentro da base de dados, o que é inaceitável em um ambiente corporativo.

A interface entre o administrador de banco de dados e o SGBD depende muito do produto adotado. Mas existe uma linguagem de interface para SGBD que é padrão para a grande maioria dos produtos do mercado. Essa linguagem é o Strutured Query Language (SQL) [William Pereira Alves (2004)].

É através do SQL que toda a manipulação dos dados é feita, desde a criação de tabelas, relacionamentos, visões, o armazenamento dos dados e finalmente o seu resgate. O SQL é uma linguagem caracter, dominada pelos administradores de bancos de dados em geral.

O PostgreSQL respeita a padronização SQL, o que permite uma grande

aderência do produto a plataformas que se utilizam de outros produtos [Denise Lemes Fernandes Neves (2002)].

Pensando nos usuários menos experientes e que não têm atribuições de administradores do SGBD, muitas interfaces gráficas foram desenvolvidas para funcionarem como uma camada mais amigável à linguagem SQL. Uma das mais conhecida para o PostgreSQL é o PGAdmin [PostgreSQL Global Development Group (2003)], que permite a criação e manipulação de tabelas, visões e outras estruturas de bancos de dados além de uma série de funcionalidades relacionadas a manipulação dos dados.

Além deste tipo de interface para administração, o PostgreSQL permite a manipulação dos dados armazenados através de camadas ODBC e JDBC ou com acesso nativo na linguagens C. Dessa forma, aplicações de qualquer nível de complexidade podem ser desenvolvidas para a manipulação dos dados armazenados no PostgreSQL, utilizando-se de qualquer linguagem de programação, como Delphi, Visual Basic, Perl, Java, C, PHP e outras [Álvaro Pereira Neto (2003)].

### 3.1.4 - Funções avançadas

Além do seu compromisso básico de armazenamento de dados, um SGBD pode oferecer um conjunto vasto de funções de apoio à manipulação e administração dos dados.

A interface básica de manipulação dos dados no PostgreSQL é o *psql*, uma ferramenta texto que permite os comandos SQL para busca, inserção, deleção e modificação dos dados hospedados nas estruturas do PostgreSQL [Denise Lemes Fernandes Neves (2002)]. Uma grande quantidade de ferramentas gráficas para a manipulação dos dados do PostgreSQL já foi

desenvolvida e se encontram disponíveis para utilização. O PGAdmin é um exemplo de interface gráfica e será explorado no capítulo 6.

Além dessas tarefas básicas, outras funções de manipulação de dados são necessárias. Geralmente, funções de conversão entre tipos de dados, funções matemáticas, funções de manipulação de *strings* e outra infinidade de funções estão disponíveis em um bom SGBD corporativo.

No caso do PostgreSQL, as funções básicas já vêem embutidas no código do produto e disponíveis para utilização imediata após a instalação. Uma das grandes vantagens do PostgreSQL sobre seus similares está na flexibilidade de permitir a cada usuário que desenvolva funções que o mesmo julgue necessárias e que não estejam disponíveis. Essas funções - *User Functions* - podem ser desenvolvidas em PSQL (linguagem SQL do PostgreSQL), C, Perl, Phyton, Java, Bash e outas [Álvaro Pereira Neto (2003)].

O PostgreSQL também apresenta funcionalidades de replicação, espelhamento, comunicação entre bases, *backup* e outas [Álvaro Pereira Neto (2003b)].

### 3.1.5 - Beneficios

O PostgreSQL se propõe a competir de igual para igual com os melhores produtos de mercado na catgegoria de Sistemas Gerenciadores de Bancos de Dados. Oferece características atraentes além da grande vantagem de ser um produto de código aberto [PostgreSQL Global Development Group (2003)]:

**Confiabilidade:** dispõe de funcionalidades internas de auto recuperação que, aliadas a técnicas de segurança como rotinas de *backup*, espelhamento de discos, *cluster*, etc, conferem alta confiabilidade

ao produto.

**Disponibilidade:** rotinas de *backup* e atualização de estatísticas a quente permitem a disponibilidade 24 x 7. Principalmente em servidores com SO LINUX, se mostra altamente estável, possibilitando altas taxas de disponibilidade.

**Desempenho:** através de técnicas complexas, utilizando-se de algarítmos de busca, indexação, indexação reversa e em cache, o PostgreSQL está entre os primeiros na comparação de performance, só ficando atrás de produtos que não contém funcionalidades essenciais, como integridade referencial e *joins* complexos.

**Flexibilidade:** por se tratar de um produto de código aberto, suas funcionalidades podem ser alteradas de forma a melhor se compatibilizar com as necessidades do ambiente.

**Facilidade de administração:** Devido a sua arquitetura, o PostgreSQL é praticamente "auto-administrável". Com exceção das rotinas de *backup* e restauração, as demais tarefas de administração de espaço, controle de estatísticas e outra infinidade de trabalhos tipicamente desenvolvidos por um Administrador de Banco de Dados (DBA) são automatizados e realizados pelo próprio produto.

**Baixo consumo de recursos:** o ambiente em que o PostgreSQL melhor se comporta é composto por um servidor de hardware básico com SO LINUX. O consumo de disco é bem menor que um SGBD Oracle, por exemplo e o consumo de memória é equivalente. Devido a facilidade de administração, recursos humanos como profissionais para a administração podem ser reduzidos.

### 3.1.6 - Distribuição e Comercialização

O PostgreSQL pode ser adquirido através de download do site www.postgresql.org. Ele também é parte integrante das mídias de distribuição do LINUX RedHat e Conectiva.

O PostgreSQL é distribuído sob a BSDLicence [PostgreSQL Global Development Group (2003)], que permite a utilização comercial e a alteração do código do produto sem o compromisso da redistribuição [FSF (2001)]. Deste modo, o custo da adoção do PostgreSQL se limita ao custo do hardware, uma vez que sua instalação em LINUX não implica em qualquer despesa com licença de uso.

### 3.1.7 - Aplicabilidade no projeto

Neste projeto, o PostgreSQL será utilizado como Sistema Gerenciador de Bancos de Dados para bases corporativas, ou seja, os dados relacionados às informações cadastrais dos funcionários da empresa estarão armazenadas em uma base PostgreSQL obedecendo uma modelagem relacional.

## 3.2 - OpenLDAP

### 3.2.1 - Definição

LDAP é um protocolo leve, que roda diretamente sobre TCP/IP e tem a função de acessar serviços de diretórios [Graham Barr (2003)]. *Serviço de*

*diretórios* nada mais é que uma base de dados simples, organizada em modelo árvore, com níveis e subníveis como galhos e suas ramificações. Pois isso é chamado de serviço de diretórios, por que sua estrutura é bastante parecida com a organização de diretórios de um dispositivo de armazenamento [Brian Arkills (2003)]. Essa base de dados tem uma estrutura simples, mas que permite alta performance e disponibilidade em grandes volumes de consultas simples. É especialmente otimizada para leituras, consultas e buscas.

Devido a essa estrutura de armazenamento, o LDAP é ideal para armazenar informações que se comportam hierarquicamente como, por exemplo, informações de uma estrutura organizacional (presidência, diretorias, gerências, departamentos, supervisões ...), informações sobre níveis de acesso de uma determinada aplicação (aplicação, tipo de acesso, grupo de usuários) ou até mesmo uma estrutura geográfica (países, estados,cidades, municípios...)

O poder de aplicabilidade do LDAP se mostra enorme exatamente devido a alta flexibilidade de organização das informações por ele armazenadas.

### 3.2.2 - Aplicação

O LDAP pode ser utilizado em várias situações, com grande aplicabilidade na função de autenticação.

Podemos dizer que autenticação é o processo pelo qual a identidade de um solicitante de serviços é atestada como sendo pertinente a um grupo de usuários conhecidos e previamente cadastrados [Jose Carlos Cordeiro Martins (2003)].

É um serviço que, basicamente, compara a identificação do solicitante ao seu banco de dados de usuários cadastrados e, baseado nos atributos deste banco de dados, permite ou não a continuidade dos serviços solicitados.

O LDAP fornece esse tipo de serviço, uma vez que pode armazenar informações e atributos de usuários. Além de fornecer o serviço de autenticação, o LDAP se integra a um vasto número de aplicativos, fazendo o papel de autenticador para eles. Esses aplicativos podem ser Sistemas Operacionais (SO) como UNIX, LINUX, Windows 2000, etc., podem ser aplicativos de mercado como Sistemas Gerenciadores de Bancos de Dados (ORACLE, PostgreSQL, MySQL, etc.), ERP (SAP, People Soft, etc.), Qmail, Samba , Squid ou aplicações desenvolvidas em linguagens de programação como PHP, Perl, Java, etc [Brian Arkills (2003)]. Também é possível uma combinação de ferramentas, como por exemplo, um SGBD que é autenticado através do SO que, por sua vez é autenticado no LDAP.

Mais uma vez, a vantagem do LDAP no campo da autenticação de usuários é a sua flexibilidade. As informações relativas ao usuário podem ser armazenadas de tal forma que até a sua posição dentro da hierarquia da árvore no LDAP pode ser traduzida em informação.

Este trabalho considera como ideal a centralização completa da autenticação de todos os recursos oferecidos em uma rede corporativa de uma empresa em uma base LDAP.

### 3.2.3 - Funcionamento básico

**Armazenamento das informações**

O modelo de informação do LDAP é baseados em "entradas". Uma entrada é uma coleção de atributos e têm um identificador único chamado Distinguished Name (DN). O DN é composto por uma sequência de atributos. Cada um dos atributos tem um tipo e um ou mais valores. Os tipos são tipicamente strings mnemônicas, como "cn" para "common name" or "mail"

para "mail address". esses mnemônicos estão definidos em RFC's como a RFC2253, a RFC2251 e a RFC3377. A sintaxe dos valores de cada atributo é definido pelo seu respectivo tipo [Brian Arkills (2003)], [OpenLDAP Foundation (2004)]. Por exemplo, um atributo mail deve seguir a sintaxe example@provider.com

Desta forma, podemos dizer que um DN é a seqüência hierárquica completa dos atributos de um determinado objeto armazenado na base de dados do LDAP, do nível mais baixo para o nível mais alto.

Um exemplo de DN:

uid=fulano,ou=Funcionario,o=empresas&a,c=BR

Esta DN indica que existe um cadastramento hierárquico na base do LDAP e sugere a estrutura representada na figura 3.2:

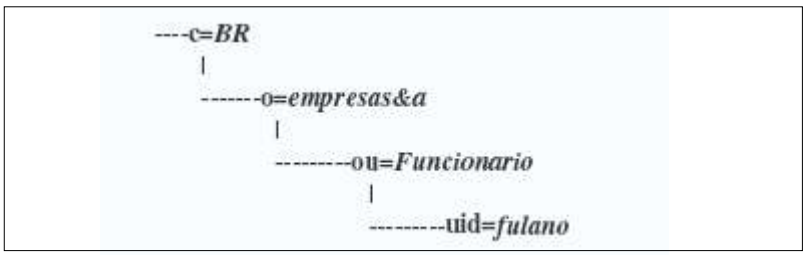


**Figura 3.2** – Cadastramento hierárquico no LDAP

Como é possível observar, as entradas são armazenadas em uma estrutura de ramificações hierárquica. A lógica da estruturação dessas ramificações vai depender da necessidade de cada natureza de informação.

Definida a lógica de organização das informações na base LDAP, também é necessário ter em mente a arquitetura de serviços que será implantada.

O LDAP possui uma construção característica de aplicações cliente-servidor. Um ou mais servidores hospedam as árvores de informações. Os

clientes se conectam ao servidor e solicitam uma informação. O servidor responde com a informação ou com um ponteiro para onde o cliente pode obter a informação (no caso de multiplos servidores). Os servidores e suas bases podem ser estruturados de 4 maneiras: serviço de diretórios local, serviço de diretórios local com referências, serviço de diretórios local distribuído e serviço de diretórios replicado, ou uma combinação deles [OpenLDAP Foundation (2004)].

A escolha da arquitetura deve estar baseada na estrutura de demanda e nas possíveis necessidades de crescimento.

No modelo local, o serviço de diretórios estará disponível apenas para o domínio a que ele pertence, sem qualquer interação com outros servidores de diretórios. esse é o modelo ideal para quem está começando ou se não existe qualquer intenção de conexão com o resto do mundo. A vantagem é a facilidade para manutenções e *upgrades*. A desvantagem pode estar na limitação e falta de flexibilidade de administração.

Se a empresa possui diversas filiais e não se pretende colocar a base de todos os funcionário em um único servidor LDAP, essa não é uma boa solução. Outro problema pode ser a diversidade de objetivos esperados do LDAP dentro da organização, ou seja, os responsáveis pela infra-estrutura de rede utilizam o LDAP para autenticação na rede, os responsáveis pela administração do correio utilizam-na para administração de listas e grupos, os responsáveis pela produção utilizam-na para administração de acessos a aplicações.

Esse tipo de cenário não é incomum e pode exigir administrações diferenciadas da base. Isso pode sugerir um modelo mais abrangente, como por exemplo o modelo árvore de referências.

**Identificação da informação**

Uma entrada é referenciada pelo seu Distinguished Name - DN (ou

"nome distinto"). O DN é único e na construção utiliza o caminho inteiro, desde a entrada até o topo do Diretório [OpenLDAP Foundation (2004)].

O DN é formado pela concatenação do nome da entrada e de todas as áreas ancestrais. Exemplo:

dn:uid=Homo Sapiens, ou=Homo, ou=Homnidae, ou=Primatas, dc=Mammalia

**Acesso à informação**

O LDAP permite operações para consultar e atualizar sua base de dados. Existem operações para adição, modificação e remoção de uma entrada do diretório. A operação LDAP de busca pode abranger a árvore toda (uma busca com escopo *subtree*) ou apenas um ramo, sem descer ou subir para os demais. Além de especificar com filtros quais entradas se deseja encontrar, também é possível especificar quais atributos dessas entradas estão sendo procurados. Se os atributos não forem especificados, todos serão retornados [OpenLDAP Foundation (2004)].

Para tanto, funções em várias linguagens como PHP, Perl e JAVA já estão disponíveis, facilitando o trabalho dos desenvolvedores. Este projeto se utiliza de funções em Perl para executar autenticação e busca de informações na base LDAP.

### 3.2.4 - Funções avançadas

Além do seu compromisso básico de autenticação, o OpenLDAP oferece um conjunto de funções de apoio à manipulação e administração dos dados.

Em relação à manipulação dos dados, o OpenLDAP traz comandos de busca, inserção, deleção e modificação de entradas na sua porção cliente. O cliente OpenLDAP já está disponível para diversas plataformas. Além da

interface texto, disponível como pacote OpenLDAP, muitas interfaces gráficas foram desenvolvidas por terceiros para facilitar a manipulação dos dados hospedados na base OpenLDAP. Um exemplo de interface é o PHPLDAPAdmin, melhor explorado no capítulo 6.

Em relação às funções de administração do ambiente, o OpenLDAP apresenta [OpenLDAP Foundation (2004)]:

**Múltiplos servidores LDAP:** O porte e geografia da organização podem exigir a distribuição de bases de autenticação. O OpenLDAP possui as funcionalidades necessárias para a descentralização da autenticação.

**Replicação:** As funções de replicação podem ser aplicadas a qualquer dos modelos de implantação. É importante em casos onde é necessário maior segurança e disponibilidade. Através da replicação é possível manter servidores de *backup*, prontos para entrar em ação em caso de falhas. Também pode ser muito útil em casos de bases descentralizadas.

**Segurança**: A segurança dos acessos , evitando as informações fiquem expostas, é imprescindível. O LDAP fornece métodos para autenticação de clientes que protegem as informações contidas no servidor. Mas a segurança somente da autenticação pode não ser tudo. Dependendo dos dados disponíveis no servidor OpenLDAP, pode ser bastante interessante proteger toda a transmissão de dados. O uso do SSL permite isso, mantendo as informações seguras de intrusos na rede.

### 3.2.5 - Benefícios

Como resultado das funcionalidades oferecidas pelo OpenLDAP, o produto acumula características que permitem muitas aplicações do serviços de diretórios. São características como:

**Portabilidade**

Existem versões de servidores LDAP para muitas plataformas, o que o torna bastante popular. É fato que ainda existe a preferência por parte dos desenvolvedores para as plataformas UNIX.

Ultimamente, com a popularização do LINUX isto deixou de ser um impeditivo. Ao contrário disso, uma arquitetura bastante encontrada é a hospedagem do servidor LDAP em LINUX com clientes rodando em diversos tipos de ferramentas visuais no Windows. Para os mais céticos, a utilização de componentes livres pode ocorrer em toda a extensão do projeto – servidores e estações LINUX rodando OpenLDAP por exemplo. Para quem ainda está receoso quanto ao LINUX, o OpenLDAP possui versões para servidores Windows.

**Flexibilidade**

A flexibilidade é um dos pontos mais fortes do LDAP. Seu poder de integração a várias plataformas, ambientes, aplicações e tecnologias é o que o faz um produto tão interessante. Uma qualidade como essa não se mantém sozinha e pode, se mal utilizada, se virar contra o administrador.

As árvores estruturadas nas bases LDAP e suas diferentes formas de implementação podem ser uma explosão de recursos, mas, por outro lado, pode ser propiciar uma desorganização nas informações. Vale dizer que é um recurso poderoso na mão de administradores bem intencionados e coerentes com a organização das bases.

**Resultados:**

A junção de portabilidade e flexibilidade da ferramenta revela um potencial de aplicabilidade bastante alto, além é claro do fator econômico que é

favorecido pelas caracterísitcas open source do produto.

Os resultados tendem a ser mais positivos conforme o planejamento e a implementação cuidadosas dessas características:

- maior integração entre diversas arquiteturas, recursos e aplicações da rede
- caminho para o "single logon" , ou seja uma base centralizada de autenticação dos usuários para quaisquer recursos disponíveis na rede;
- maior segurança nos acessos a sistemas e informações;
- padronização do processo de autenticação;
- padronização da administração de acessos;
- integração das bases de autenticação com as bases corporativas;
- flexibilidade de administrações diferenciadas para árvores ou ramos específicos da base de dados mantida no servidor LDAP;
- melhor administração;

### 3.2.6 - Distribuição e Comercialização

O OpenLDAP é uma das várias distribuições de LDAP disponíveis. A distribuição da OpenLDAP Foundation [OpenLDAP Foundation (2004)] , utilizado neste projeto é lcenciada sob a OpenLDAP Public License. Essa é uma licença permissiva de software livre, que não exige copyleft, exige apenas a inclusão dos termos da licença nas redistribuições [OpenLDAP Licence (2004)].

### 3.2.7 - Aplicabilidade no projeto

Neste projeto, o Open será utilizado como Serviço de Diretórios para autenticação da rede corporativa, ou seja, os dados relacionados às informações de autenticação dos funcionários da empresa estarão armazenadas em uma base OpenLDAP, obedecendo uma modelagem hierárquica.

## 3.3 - Utilização do PostgreSQL como base corporativa

Muitas são as razões que estão levando o mercado empresarial a buscar alternativas de softwares frente aos tradicionais lideres de mercado. Potências como Microsoft, IBM e ORACLE no campo dos gerenciadores de bancos de dados já sofrem alta concorrência de produtos de código aberto.

O PostgreSQL possui qualidades como robustez, portabilidade, confiabilidade e facilidade de uso que podem, se sombra de dúvida, fazer frente aos melhores produtos de mercado.

Já não são poucas as empresas que adotam o PostgreSQL. O Metrô de São Paulo, por exemplo, possui bases significativamente pesadas e altamente importantes para o seu funcionamento hospedadas em bases PostgreSQL nos seus diversos prédios administrativos.

No site oficial www.PostgreSQL.org, muitos outros exemplos de usuários podem ser vistos e suas declarações sobre resultados obtidos são bastante positivos.

## 3.3 - Utilização do LDAP como base de autenticação

Existem inúmeras alternativas para uma arquitetura de segurança e autenticação em uma rede corporativa. O fato mais relevante é que quanto mais organizada e bem administrada for essa arquitetura, menos as chances de falha

em consequentemente, maior a segurança para os dados e aplicações nesta rede.

É neste requisito que o LDAP se mostra uma excelente alternativa. Isso porque o seu alto grau de portabilidade e de integração permite que possa existir uma arquitetura de autenticação em base unificada. Ou seja, uma vasta gama de aplicativos de mercado e de linguagens de desenvolvimento já estão preparadas para a integração com uma base LDAP.

Na prática, essa estrutura integrada resulta na facilidade ao usuário final – que só é solicitado em autenticação uma única vez no logon na rede , e na facilidade do administrador – que através da definição de perfis administra todo o leque de recursos disponíveis.

# Capítulo 4

# Integração das bases de autenticação e corporativa

Definidos os produtos para autenticação e para o armazenamento dos dados corporativos, um esclarecimento sobre as vantagens da integração entre os mesmos se faz necessário.

## 4.1 - Finalidade

O interesse em integrar a base de autenticação à base corporativa surge no momento em que os dados da base de autenticação necessitam constante atualização. Isso ocorre devido ao dinamismo característicos das organizações, que contratam novos funcionários periodicamente, os promove em todas as direções organizacionais e, finalmente os afasta por diversos motivos.

Esse acompanhamento da "vida profissional" de cada funcionário tem que, de alguma forma, estar contemplado na base de autenticação. Um funcionário afastado não pode ter direito aos recursos da rede, um funcionário do departamento RH, movimentado para o FINANCEIRO deve ter o seu perfil

de acesso alterado, deixando de acessar determinadas aplicações específicas do departamento RH e passando a ser cliente das aplicações do FINANCEIRO.

Normalmente, as informações relativas aos funcionários de uma organização se encontram na base corporativa, são alimentadas e atualizadas por uma equipe de profissionais especializados em recursos humanos através de uma aplicação desenvolvida sob-medida ou de um pacote como um ERP.

A equipe responsável por manter essas informações raramente compreende que elas devem ser repassadas à equipe de segurança da rede. Não vêem nenhuma relação entre os departamentos de Recursos Humanos (RH) e de Tecnologia da Informação (TI).

Como conseqüência, não é raro haver divergência entre as informações armazenadas na base de autenticação e a realidade profissional do funcionário. Ele se loga na rede e tem acesso negado por estar afastado, por exemplo. Ou ainda, apesar de já trabalhar no departamento Administrativo há duas semanas, ainda tem acesso a aplicações de Recursos Humanos, sua antiga área.

Um dos piores casos são as demissões. Imagine que um funcionário demitido ainda continue com acesso aos recursos de informática da empresa. Os prejuízos podem ser irreparáveis.

Situações como essa são imperdoáveis para a imagem da empresa, da sua área de TI e, principalmente é uma grave falha de segurança. E para que sejam evitadas, nada mais indicado do que automatizar a atualização da base de autenticação.

Considerando que a origem dessas informações é a base de dados corporativa, a solução é integrar a base corporativa à base de autenticação, de modo que a cada atualização na base corporativa, a mesma alteração seja espelhada imediatamente para a base de autenticação. Com esse processo, a situação do funcionário disponível na sua base original seria respeitada também pela segurança da rede corporativa.

## 4.2 - Determinação da topologia de atualização

O processo de integração entre as bases corporativas e de autenticação pode ser feita de muitos modos. Algumas características do ambiente e da política empresarial em questão podem ser indicativos da melhor alterativa de implantação. Para que o problema e a solução sejam melhor definidos, é imprescindível o levantamento do ambiente, dos recursos disponíveis e do escopo da solução.

### 4.2.1 - Levantamento das necessidades

É o benefício esperado.

Quais serão as utilizações básicas da base de autenticação? Que informações serão replicadas? Existirão deleções de informações? Existirá atualização externa nessa base? Haverá histórico de atualizações?

No levantamento das necessidades, perguntas como essas devem ser detalhadamente respondidas. Esse escopo vai definir quantas e quais rotinas deverão ser previstas no processo de atualização.

### 4.2.2 - Periodicidade

A periodicidade define qual o nível de integração necessário. Muitas vezes, a própria origem da informação não é atualizada em tempo real. É possível que o departamento de Recursos Humanos adote uma rotina de

atualização das bases semanal ou mensal. Nestes casos é inútil uma transferência diária de dados para a base de autenticação.

A periodicidade vai influenciar de maneira determinante na escolha do modelo de transferência. Uma replicação com características de espelhamento deve prever gatilhos, disparados a cada atualização. Uma replicação em lote pode ser feita através de rotinas agendadas pelo sistema operacional, pelo próprio *software* do banco de dados ou por uma rotina externa.

### 4.2.3 - Confiabilidade

Esse requisito define o grau de confiabilidade esperado do processo em relação à qualidade do conteúdo da base. Ou seja, qual a conseqüência causada pela inconsistência das informações e qual o período de defasagem máximo suportado.

### 4.2.4 -Disponibilidade

Disponibilidade é o percentual de tempo em que o sistema fica disponível. Em outras palavras, qual a conseqüência da indisponibilidade da base LDAP atualizada e qual o período máximo de indisponibilidade suportado.

## 4.3 - Levantamento dos recursos disponíveis

Definido o scopo do projeto, é necessário o levantamento dos recursos que poderão ser utilizados para o cumprimento dos benefícios esperados. Esses recursos envolvem *hardware*, *software*, recursos humanos, prazos e recursos externos.

### 4.3.1 - Recursos Materiais

Os recursos materiais necessários para a implantação do projeto dependerá de quais produtos e recursos já se encontram disponíveis.

Considerando os produtos escolhidos, ou seja o OpenLDAP para a base de dados de autenticação e o PostgreSQL para a base corporativa, fazem parte da lista de ferramentas básicas necessárias:

***Hardware***:

Servidor Pentium III 512 RAM, HD de 30G para hospedagem da base corporativa;

Servidor Pentium III 512 RAM, HD de 30G para hospedagem da base de autenticação;

***Software* básico:**

SO LINUX Redhat 9.0 ou similar

SGBD PostgreSQL 7.3 ou superior

Interface de administração para PostgreSQL (PgAdmin ou similar)

OpenLDAP 2.2 ou superior

Interface de administração para bases LDAP (PhpLDAP ou similar)

### 4.3.2 - Recursos Humanos

Para a execução do projeto de integração será necessária a colaboração de diversos profissionais como:

- responsáveis pelas atividades de gestão das informações relativas a empregados na empresa (analistas de sistemas e/ou administradores de dados);
- responsáveis pela hospedagem e manutenção das informações na base de dados corporativa (DBA);
- responsáveis pela segurança da rede corporativa (analista de segurança);

## 4.4 - Levantamento do ambiente disponível

Por estarmos tratando de uma organização já estabelecida, muito provavelmente alguns recursos daqueles levantados como necessários já podem existir. Neste caso, é importante um levantamento junto aos administradores para um melhor aproveitamento do recurso ou para um redirecionamento das necessidades e das interfaces do projeto.

### 4.4.1 - Servidores de autenticação

Muito provavelmente já existam servidores de autenticação na rede corporativa. Pode ser um Primary Domain Control no caso de uma rede Microsoft, um servidor LINUX , *softwares* de terceiros ou até mesmo um servidor LDAP.

Nesses casos o projeto pode assumir proporções maiores. Será

necessária uma análise para implantação da base unificada LDAP ou administrar essa nova base em paralelo a atual.

Devido ao grande poder de integração que o LDAP possui, são raras as situações onde a base LDAP não seja viável, porém, o processo de migração pode não ser tão simples, dependendo do porte da rede e do domínio da organização sobre suas aplicações.

### 4.4.2 - Servidores das bases corporativas

Assim como os servidores de bases de autenticação, uma empresa já consolidada tende a já possuir um ou mais servidores de bancos de dados para a hospedagem dos dados corporativos. esses servidores podem estar baseados em *softwares* diferentes como SQLServer, ORACLE, DB2, MySQL e outros.

Nesses casos, o projeto deve se adaptar às bases já existentes, captando as informações onde quer que elas estejam. Isso porque se torna impraticável solicitar qualquer alteração em uma base de dados corporativa já em produção.

## 4.5 - Mapeamento das bases (autenticação e corporativas)

Em qualquer que seja o caso – bases já existentes ou bases criadas junto com o projeto – a disposição e a administração das informações é crucial para o sucesso do projeto pois será a partir da estrutura dos dados corporativos que o filtro para a replicação será montado.

### 4.5.1 - Administradores da informação

Os profissionais envolvidos na administração das informações corporativas devem ser envolvidos na implantação, para que as informações relativas aos funcionários possam ser disponibilizadas (direitos de acesso) e para que a dicionarização destes dados possa ser reconhecida e documentada.

A necessidade do profissional administrador das bases de dados vem do fato de que nem sempre é simples um mapeamento do conteúdo de cada tabela em uma base de dados. Os dados necessários podem estar distribuídos em várias tabelas, em várias instâncias do banco de dados e até em vários servidores.

Somente com o conhecimento do administrador de dados é possível identificar a localização exata das informações necessárias e trata-las de modo a facilitar o seu acesso.

### 4.5.2 - Disposição e organização das informações

Caso a organização não possua profissionais responsáveis pela administração das informações corporativas ou que dominem a arquitetura de dados existentes, seu conteúdo e organização, garimpar as informações necessárias na sua origem será uma tarefa mais complexa.

Em muitos casos é necessário recorrer ao fornecedor do aplicativo – no caso de pacotes, ou ao desenvolvedor – em caso de produção interna. De qualquer forma, antes de traçar os métodos de replicação, é essencial o mapeamento das informações nas bases corporativas. Esse mapeamento inclui basicamente:

- localização física das bases envolvidas;
- servidores, sistemas operacionais e SGBD envolvidos;

- integridade das informações;
- metodologia de atualização das informações.

## 4.6 - Levantamento das Técnicas possíveis

Com o mapeamento das informações em mãos já é possível partir para a determinação de técnicas possíveis baseadas na arquitetura e nos recursos disponíveis. Para uma discussão organizada, as funcionalidades oferecidas por cada produto participante devem ser levadas em conta.

Mais uma vez, os produtos avaliados são o Gerenciador de Banco de Dados PostgreSQL e a ferramenta de autenticação OpenLDAP.

### 4.6.1 - Utilização dos recursos do PostgreSQL

Assim como os melhores SGBD do mercado, o PostgreSQL oferece uma série de recursos internos para a execução de tarefas vinculadas a administração dos dados armazenados.

Este capítulo não tem a intenção de discutir todos os recursos oferecidos pelo produto, pois seria fugir do objetivo principal do projeto. O objetivo é apresentar funcionalidades que possam ser aplicadas ao objetivo da integração com outros produtos.

Uma verdadeira vastidão de funções já se encontra disponível no código padrão do banco e as que não estão embutidas podem ser encontradas na comunidade de desenvolvimento, na maioria das vezes [Álvaro Pereira Neto (2003)], [Álvaro Pereira Neto (2003b)].

Dentro da filosofia do código aberto, o PostgreSQL é extremamente

flexível quanto a novas necessidade e aceita a codificação de pequenos programas para a sua adaptação perfeita às necessidades de seus usuários.

Esses pequenos programas são as funções, *procedures* e *triggers*. Elas podem ser criadas conforme a necessidade da situação em diversas linguagens de programação como perl, C, java, etc.

A codificação de pequenos códigos internos a um SGBD é uma rotina relativamente freqüente para os seus administradores. A necessidade de implementar determinados comportamentos na base de dados os leva a traçar lógicas específicas, acompanhando a política da organização, modelos de integração, etc.

#### *4.6.1.1 - Triggers*

As *triggers* (ou gatilhos) são códigos executados mediante o comportamento dos objetos do banco, ou seja, são "disparados" quando determinado evento ocorre dentro da base [Denise Lemes Fernandes Neves (2002)].

Um exemplo clássico é a chamada "deleção em cascata", quando é definido que, para cada registro deletado em uma determinada tabela, um outro registro será automaticamente deletado em uma segunda tabela. A figura 4.1 mostra um exemplo de trigger escrita em psql.

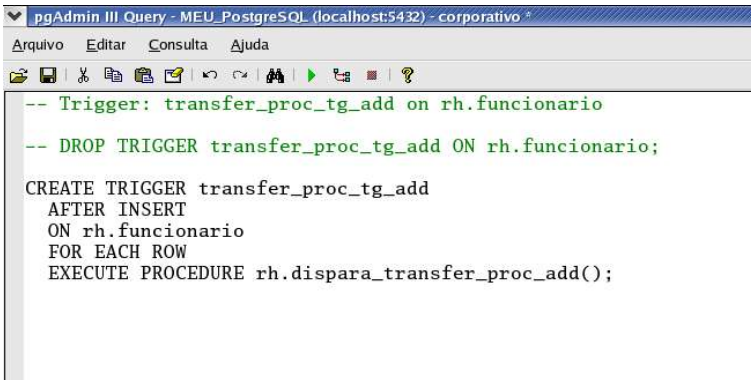


**Figura 4.1** – Trigger PostgreSQL em psql

A figura 4.2 mostra o resultado da utilização da trigger exemplo.


**Figura 4.2** – Resultado de delete disparando trigger

### *4.6.1.2 - Funções*

As funções são pequenos programas independentes que podem ser chamados a qualquer momento – inclusive por uma trigger. Normalmente são transformações complexas de dados, chamadas a comandos externos, manipulação de strings, etc. As funções trabalham com a passagem de parâmetros e podem ser acionadas independentemente ou como parte de um comando SQL [Denise Lemes Fernandes Neves (2002)].

As figuras 4.3 e 4.4 mostram exemplos de codificação de funções em linguagem plpsql e plperl no PostgreSQL.

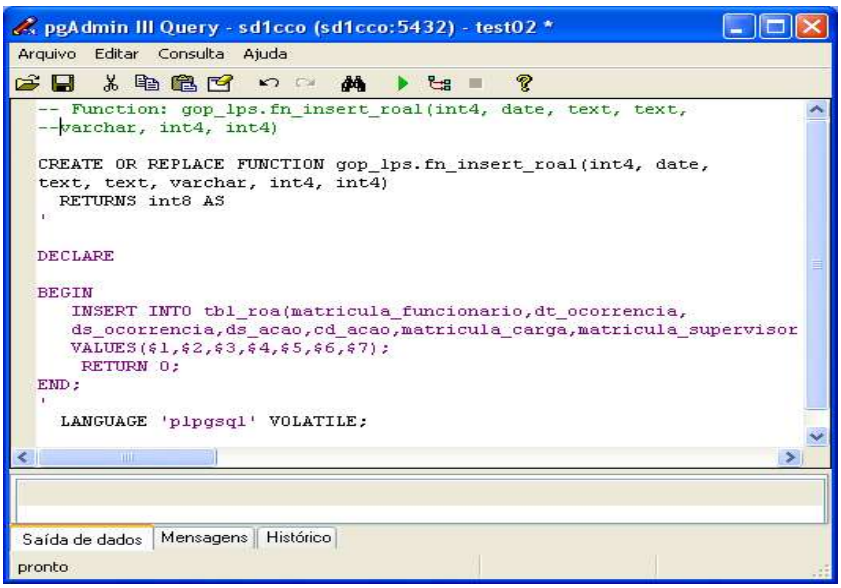


**Figura4.3** – Exemplo de função no PostgreSQL.

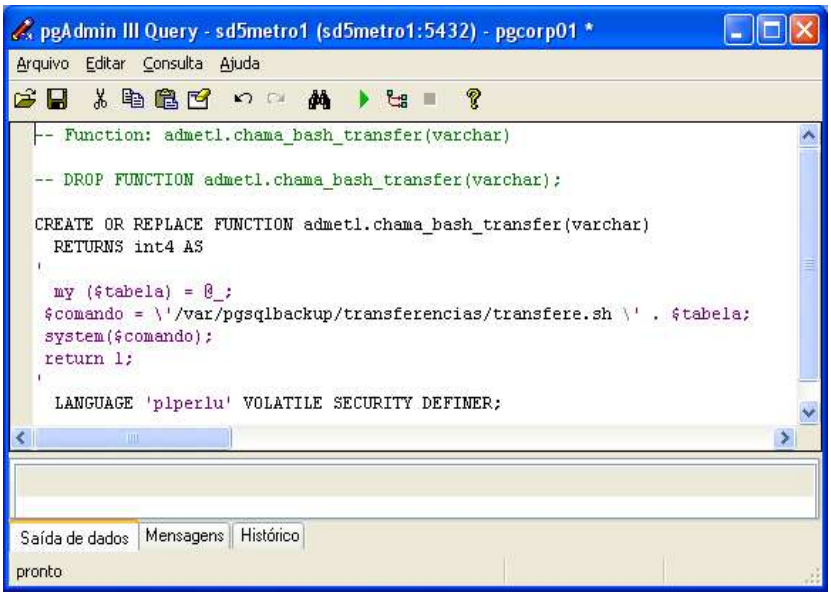


**Figura 4.4** – Exemplo de codificação de função em perl

Como característica própria, o PostgreSQL permite a codificação de funções em diversas linguagens de mercado. A linguagem nativa do banco é chamada PostgreSQL Structured Query Language (psql) e tem como característica ser extremamente adaptadas às necessidades de manipulação de dados.

Além do psql, o PostgreSQL admite programação em C, Java, Perl e Python [Álvaro Pereira Neto (2003b)]. Diversas bibliotecas estão sendo desenvolvidas para o suporte a outras linguagens de mercado [PostgreSQL Global Development Group (2003)].

## 4.6.2 - Utilização dos recursos do LDAP

Assim como exposto na análise de funcionalidades do PostgreSQL, esse capítulo não tem a intenção de discutir todos os recursos oferecidos pelo

OpenLDAP, mas sim apresentar funcionalidades que possam ser aplicadas ao objetivo da integração com produtos externos.

#### *4.6.2.1 - Recursos da ferramenta de repositório*

O LDAP permite a opção, na sua instalação, de qual mecanismo de armazenamento será utilizado. A configuração padrão prevê o armazenamento em uma arquivo local, mas é possível a utilização de bases de dados específicas, como o BerkleyDB, por exemplo.

Dependendo da arquitetura do produto escolhido, é possível o acesso direto a informação através do software de armazenamento, sem passar pela camada do LDAP [Sleepycat Software (2004)].

#### *4.6.2.2 - Comandos do LDAP Client*

Faz parte do pacote de instalação do LDAP uma interface cliente para o produto. Essa interface (existe para vários sistemas operacionais) permite as operações básicas na base LDAP: busca, inserção, deleção e edição de conteúdos [OpenLDAP Foundation (2004)].

#### *4.6.2.3 - Interface facilitada com linguagens*

O grande diferencial do LDAP é a enorme variedade de produtos e linguagens já adaptadas às rotinas de comunicação e manipulação de seu conteúdo. Essa característica pode ser decisiva na opção do modelo de integração.

Linguagens como Python, C, Java, Perl e PHP já possuem bibliotecas de funções específicas para o acesso ao LDAP. Através dessas funções é possível fazer a conexão, busca, inclusão, exclusão e alteração de registros armazenados na base LDAP, mesmo se a mesma estiver utilizando outras formas de armazenamento, como BerkleyDB, por exemplo [OpenLDAP Foundation (2004)].

#### *4.6.2.4 - Importação e expotação de arquivos formato LDIF*

Os arquivos *LDAP Data Import Format* (LDIF) são arquivos com uma formação específica que são portáveis entre produtos LDAP e entre ferramentas de administração do produto, como o PHPLDAPAdmin, o Webmim e outros. A utilização de arquivos em formato LDIF pode ser uma boa opção para o carregamento em *batch* da base LDAP.

Por sua formação ser perfeitamente reproduzível, é possível a criação de arquivos LDIF a partir de conteúdos de outras bases de dados. Para isso bastaria uma rotina para a conversão de formatos.

# Capítulo 5

# Plano de atuação

Uma vez analisados os recursos disponíveis, é hora de um plano de ação. Qual a combinação de tecnologias e recursos que melhor se adapta ao objetivo final?

## 5.1 - Modelo da integração

A macro estrutura da solução ou modelo de integração seguirá o fluxo descrito na figura 5.1. O fluxo da informação se comporta da seguinte forma:

- a informação é gerada através de uma aplicação utilizada no departamento de Recursos Humanos;
- a informação fica hospedada em uma base PostgreSQL na tabela "**funcionario***";*
- a cada atualização nesta tabela, existe uma propagação da informação atualizada para a base de autenticação LDAP.

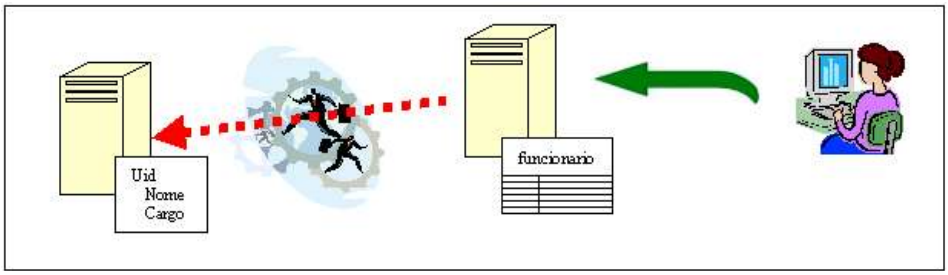


**Figura 5.1** – Modelo de integração

## 5.2 - Recursos e técnicas empregados

A opção pelos recursos empregados levou em conta fatores como:

- portabilidade;
- disponibilidade;
- consumo de recursos;
- facilidade de implantação;
- facilidade de administração.

Devido a necessidade de replicação instantânea, definida na fase de levantamento do projeto, serão utilizados os recursos de triggers, disponíveis no PostgreSQL. Com a utilização dessa funcionalidade, qualquer alteração na tabela de origem vai disparar o "evento de replicação" instantâneamente.

No passo seguinte, os recursos de programação de perl, disponíveis no PostgreSQL foram aliados à aderência dessa linguagem ao produto LDAP. Bibliotecas específicas para a comunicação com o LDAP foram instaladas no servidor LINUX onde está hospedado o SGBD PostgreSQL. Com elas, foram criadas rotinas externas em Perl, acionadas através das triggers.

Essas rotinas em Perl são responsáveis pela conexão e manipulação dos dados da base de autenticação LDAP.

Resumindo, foram utilizados os seguintes recursos:

- triggers (PostgreSQL),
- funções de trigger (PostgreSQL);
- funções (PostgreSQL);
- perl com biblioteca Net::LDAP (Perl);

# Capítulo 6

# Implantação

Para a reprodução do plano de implantação deste projeto, alguns pré-requisitos serão considerados:

- os produtos PostgreSQL e OpenLDAP já se encontram instalados e operantes;
- as informações de empregados são originadas em uma única base PostgreSQL, através de uma aplicação transacional já existente;
- a base de autenticação já foi criada e contém uma réplica fiel das informações corporativas;

## 6.1 - Instalação e configuração de produtos adicionais

### 6.1.1 - PgAdminIII

O PostgreSQL, como já dito, não inclui interface gráfica para administração. Em compensação, permite o acoplamento de uma infinidade de produtos disponíveis para esse mesmo fim.

A ferramenta mais conhecida entre os Administradores de Bancos de Dados que se utilizam do PostgreSQL é o PgAdmin [PostgreSQL Global Development Group (2003)]. É uma ferramenta licenciada pela Artistic License [FSF (2001)], que vem evoluindo a passos largos em suas implementações, tornando a tarefa de administrar uma base PostgreSQL cada vez mais fácil e eficiente.

A versão III está disponível para a instalação em GNU/Linux, FreeBSD e Windows 2000/XP. Além de portável, a ferramenta está disponível em 30 linguagens diferentes e contém documentação completa.

A instalação não requer nenhuma bilbioteca adicional. Basta fazer o download no site http://www.pgadmin.org/pgadmin3/download.php e seguir as instruções conforme o sistema operacional utilizado.

Neste projeto a instalação foi feita através dos pacotes RPM disponíves no site de download. Após a aquisição dos pacotes, basta executar o comando *rpm -Uvh pgadmin*.rpm*. Não existem configurações adicionais. A figura 6.1 mostra a tela inicial do PgAdminIII.

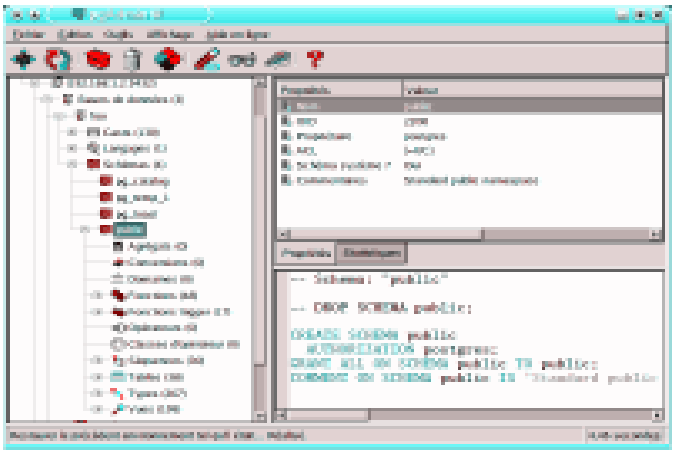

**Figura 6.1** – Tela inicial do PGAdminIII

### 6.1.2 - PHPLDAPAdmin

Assim como o PostgreSQL, o OpenLDAP não oferece em seu pacote original nenhuma interface gráfica para administração. Existem muitos projetos para ferramentas com essa finalidade. Uma delas é o PHPLDAPAdmin. Esse produto permite a execução das tarefas básicas de adminisgtração do LDAP como a visualização da árvore, visualização de esquemas, busca, criação, deleção, copia e edição de entradas. Além disso, a grande vantagem do PHPLDAPAdmin é estar em ambiente WEB, permitindo a administração da base de autenticação de qualquer estação conectada à rede.

O PHPLDAPAdmin é distribuído sob a General Public Licence (GNU) [FSF (2001)]e pode ser adquirido através de download no site do projeto: http://phpldapadmin.sourceforge.net/.

Sua instalação requer um servidor WEB em funcionamento e é bastante simples, uma vez que basta descompactar o conteúdo do arquivo disponibilizado no site para dentro da pasta de documentos utilizada pelo servidor WEB. A integração entre o PHP e o PostgreSQL requer, apenas, a configuração do servidor web com o módulo relativo ao PostgreSQL [Ewald Geschwinde, Hans-juergen Schoenig (2002)].

Após a instalação, é necessário configurar o arquivo *config.php* com os dados respectivos ao servidor LDAP. Um exemplo para essa configuração pode ser visto no anexo 1.

A figura 6.2 mostra a tela inicial do PHPLDAPAdmin. A figura 6.3 mostra o PHPLDAPAdmin no ambiente do projeto.


**Figura 6.2 –** Tela inicial do PHPLDAPAdmin

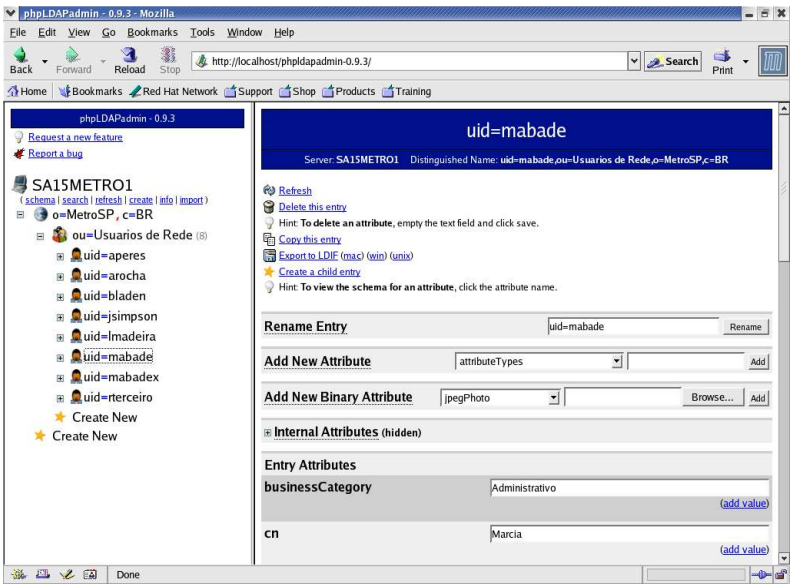


**Figura 6.3 –** Tela  do PHPLDAPAdmin no ambiente do projeto

### 6.1.3 - Perl

Perl é uma linguagm de programação com importantes características como estabilidade, portabilidade, extensibilidade e produtividade [Larry Wall, Tom Christiansen, Jon Orwant (2001)]. É muito utilizada para tarefas de administração de ambientes e para a progamação WEB. Perl é Open Source, licenciado sobre Artistic License, ou GNU General Public License [FSF (2001)] e está disponível para diversos sistemas operacionais como Unix systems, LINUX, Macintosh , Windows e outros.

A linguagem de programação perl é pre-requisito para este projeto. Ela deve estar instalada no servidor onde está o PostgreSQL (base de dados corporativa).

Para a sua instalação, basta arquirir os arquivos correspondentes ao sistema operacional do servidor no site http://www.perl.org/. Neste projeto, por se tratar de um servidor LINUX RedHat 9, optou-se pelos pacotes RPM ( rpm -Uvh ActivePerl*.rpm ).

### 6.1.4 - Módulos LDAP para Perl

Uma das caracterísiticas mais importantes do Perl é a sua extensibilidade. As funcionalidades contruídas através das mais variadas necessidades dos desenvolvedores funcionam como uma extensão da linguagem [Larry Wall, Tom Christiansen, Jon Orwant (2001)]. Ou seja, a reutilização de código ocorre naturalmente.

Este projeto utiliza essa característica do Perl. A possibilidade de uso de um módulo já pronto para o acesso e manipulação dos dados da base LDAP foi descoberta durante o levantamento de técnicas possíveis.

**Net::LDAP** é uma coleção de módulos que implementam interfaces a serviços LDAP para programas Perl [Paul Dwerryhouse (2003)]. O módulo pode ser utilizado para realizar buscas na árvode de diretórios ou para realizar funções de manutenção como adição, deleção ou alteração de entradas na base LDAP.

A instalação do módulo referentes à comunicação com servidore LDAP é um exemplo de extensão da linguagem Perl. Através da utilização dessa bilbioteca, o tempo de desenvolvimento do projeto foi drasticamente reduzido.

A instalação da bilbioteca "*Net::LDAP - Lightweight Directory Access Protocol"* pode ser feita através do download em formato RPM de: http://atrpms.net/dist/rh9/perl-ldap/perl-ldap-0.31-2.rh9.at.noarch.rpm.html.

### 6.1.5 - Linguagem Perl no PostgreSQL

A linguagem Perl não é instalada por *default* no pacote básico do PostgreSQL . É necessária a adição do pacote *plperl* na instalação [PostgreSQL Global Development Group (2003)].

O software pode ser adquirido no site www.PostgreSQL.org

### 6.1.6 - Código para teste

Para a averiguação do cumprimento dos objetivos do projeto, está disponível no anexo 2 uma pequena aplicação, desenvolvida em PHP.

A aplicação é composta por três arquivos (ufla1.html, ufla1.php e ufla2.php) e imagens que compõem suas telas.

A sua instalação consiste na cópia dos arquivos para o diretório de documentos utilizados pelo servidor WEB. Sua execução consistem na chamada do ufla1.html no browser cliente.

Esta aplicação realiza um "*logon*" através da autenticação do utilizador em uma base LDAP. Caso o usuário não seja autenticado, um erro é exibido. Em caso de sucesso, uma tela com dados do usuário é exibida, simulando um portal corporativo.

A figura 6.4 mostra a tela de logon da aplicação. A figura 6.5 mostra a tela que simula o portal corporativo acessado com sucesso.

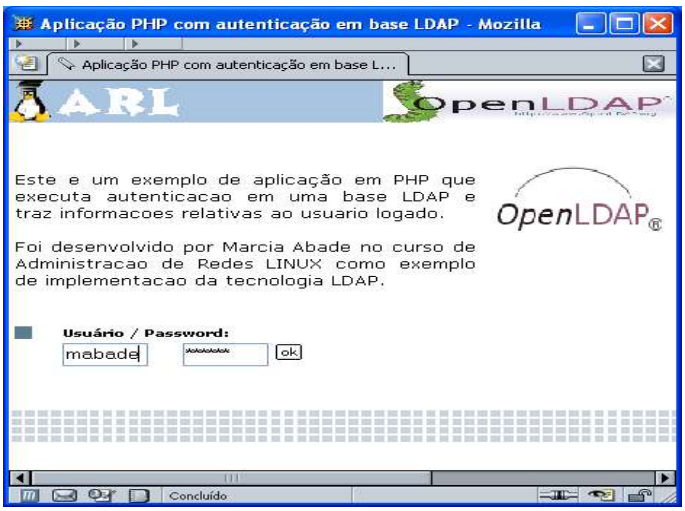


**Figura 6.4** - Tela de logon da aplicação exemplo

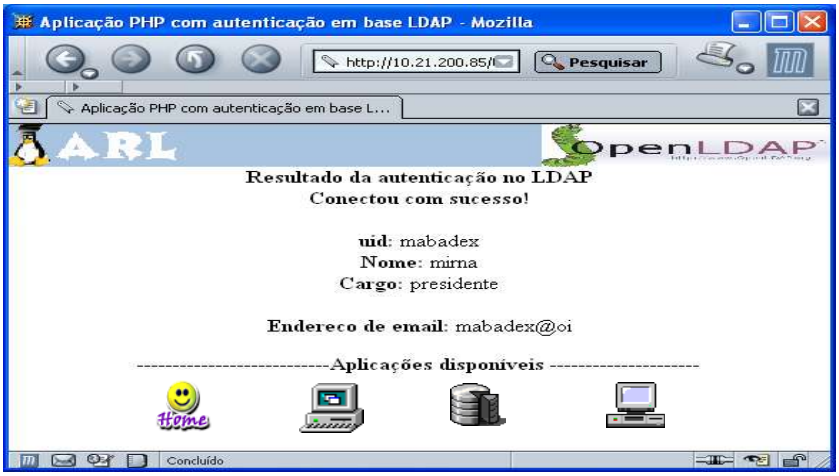


**Figura 6.5** - Tela de logon com sucesso na aplicação exemplo

# 6.2 - Codificação dos programas Perl

Pela facilidade proporcionada pelo SGBD PostgreSQL de aceitar funções internas codificadas em Perl, a programação da lógica de transferência diretamente na base de dados se torna uma alternativa possível.

Porém, considerando-se o fator portabilidade, a alternativa da codificação de programas externos ao PostgreSQL foi adotada. Dessa forma, o projeto de atualização fica disponível para qualquer SGBD que suporte chamadas ao sistema openacional – e essa é uma característica bastante comum entre os produtos de mercado.

O códico é bastante simples e está apresentado com comentários para o seu completo entendimento. O único pré-requisito é a instalação do móduto Net::LDAP, visto no capítulo 6.1.4.

Foram desenvolvidos códigos para duas rotinas: a inclusão de novas entradas no LDAP e a alteração de dados existentes.

Código para rotina de inserção de dados no LDAP.

```
use Net::LDAP;
# usa o modulo Net::LDAP

$ldap = Net::LDAP->new( "sa15metro1" ) or die "$@";
# cria string de conexão ao LDAP

$mesg = $ldap->bind("cn=root,o=MetroSP,c=BR", password=>"secret");
# faz o bind e armazena o retorno da função  na variavel  mesg

if ($mesg->code)
# checa erro no bind
{
                print "------------erro no bind------------\n";
                LDAPerror ( "Binding", $mesg );
                # chama rotina de tratamento de erros
}

$result = $ldap->add("uid=".$ARGV[0].",ou=Usuarios de Rede,o=MetroSP,c=BR",
              attr => [ 'uid' => $ARGV[1],
                        'cn' => $ARGV[2],
                        'sn' => $ARGV[3],
                        'userPassword' => $ARGV[4],
                        'employeeType' => $ARGV[5],
                        'employeeNumber' => $ARGV[6],
                        'mail' => $ARGV[7],
                        'businessCategory' => $ARGV[8],
                       'departmentNumber' =>$ARGV[9],
                        'ou' => [$ARGV[10],$ARGV[11]],
                       'physicalDeliveryOfficeName' => $ARGV[12],
                       title' => $ARGV[13],
                       'uniqueMember' => [$ARGV[14],$ARGV[15]],
                        'objectclass' => [ 'person','inetOrgPerson','groupOfUniqueNames']
                       ]
         );
# faz a insercao com os argumentos enviados na linha de comando e armazena resultado na variavel result


if ($result->code)
# checa erros na insercao
{
                print "------------erro no add------------\n";
                LDAPerror ( "Binding", $result );
                # chama rotina de tratamento de erros
}


 sub LDAPerror
 {
   my ($from, $mesg) = @_;
   print "Return code: ", $mesg->code;
   print "\tMessage: ", $mesg->error_name;
   print " :",          $mesg->error_text;
   print "MessageID: ", $mesg->mesg_id;
   print "\tDN: ", $mesg->dn;
 }
```

Código para rotina de alteração de dados no LDAP.

```
use Net::LDAP;
# utiliza modulo Net::LDAP

$ldap = Net::LDAP->new( "sa15metro1" ) or die "$@";
# testa conexao com servidos LDAP e armazena resultados na variavel ldap

$mesg = $ldap->bind("cn=root,o=MetroSP,c=BR", password=>"secret");
# faz o bind e armazena o resultado na variavel mesg

if ($mesg->code)
# testa erro no bind
{
                print "------------erro no bind-------------\n";
                LDAPerror ( "Binding", $mesg );
                # chama rotina de tratamento de erros
}

$dn = "uid=".$ARGV[0].",ou=Usuarios de Rede,o=MetroSP,c=BR";
# define dn a ser alterado, considera sempre o mesmo ou, o e c, recebe  uid po parametro

$mesg = $ldap->modify($dn, replace => { $ARGV[1] => $ARGV[2] } );
# faz a alteracao no dn definido, considerando o aributo e valor passados por argumentos

 sub LDAPerror
#  rotina de tratamento de erros
 {
   my ($from, $mesg) = @_;
   print "Return code: ", $mesg->code;
   print "\tMessage: ", $mesg->error_name;
   print " :",          $mesg->error_text;
   print "MessageID: ", $mesg->mesg_id;
   print "\tDN: ", $mesg->dn;
 }

 $ldap->unbind;
```

# 6.3 - Codificação das funções

Para que os programas externos, codigicados em Perl, sejam acionados por rotinas internas do SGBD PostgreSQL, as chamadas devem ser feita através de uma função. Esses pequenos códigos podem ser desenvolvidos em diversas linguagens, como visto no capítulo 3.1.4. Para manter um padrão de pregramação, optou-se pela linguagem Perl.

Como a lógica da transferencia foi codificada nos programas Perl, as funções desenvolvidas no PostgreSQL apenas recebem os valores enviados pelas funções de trigger e os repassa como parâmetros para os programas externos.

Foram desenvolvidas duas funções, a função ldap_add para os eventos de inserção e a função ldap_modify para os eventos de update. A deleção de dados não foi considerada devido à caracteríricas da base de dados corporativa de funcionários, onde nunca existe deleção de registros.

```
CREATE OR REPLACE FUNCTION postgres.ldap_add(varchar, varchar, varchar, varchar, varchar, varchar, varchar,
varchar, varchar, varchar, varchar, varchar, varchar, varchar, varchar, varchar, varchar)
  RETURNS varchar AS
'
 my ($uid) = $_[0];
 my ($par_uid) = $_[1];
 my ($par_cn) = $_[2];
 my ($par_sn) = $_[3];
 my ($par_passwd) = $_[4];
 my ($par_empType) = $_[5];
 my ($par_empNumber) = $_[6];
 my ($par_mail) = $_[7];
 my ($par_businessCateg) = $_[8];
 my ($par_depNember) = $_[9];
 my ($par_ou1) = $_[10];
 my ($par_ou2) = $_[11];
 my ($par_physicalName) = $_[12];
 my ($par_title) = $_[13];
 my ($par_uniqueMember1) = $_[14];
 my ($par_uniqueMember2) = $_[15];
 $comando = \'perl /home/r188763/ldap/perl_add.pl \' .
                 $uid . \' \'.
                 $par_uid . \' \'.
                 $par_cn . \' \'.
                 $par_sn . \' \'.
                 $par_passwd . \' \'.
                 $par_empType . \' \'.
                 $par_empNumber . \' \'.
                 $par_mail . \' \'.
                 $par_businessCateg . \' \'.
                 $par_depNember . \' \'.
                 $par_ou1 . \' \'.
                 $par_ou2 . \' \'.
                 $par_physicalName . \' \'.
                 $par_title . \' \'.
                 $par_uniqueMember1 . \' \'.
                 $par_uniqueMember2 . \' \';
  my $result = system($comando);
 return $comando;
'
  LANGUAGE 'plperlu' VOLATILE SECURITY DEFINER;
```

```
CREATE OR REPLACE FUNCTION postgres.ldap_modify(varchar, varchar, varchar)
  RETURNS varchar AS
'
 my ($uid) = $_[0];
 my ($parametro) = $_[1];
 my ($valor) = $_[2];
 $comando = \'perl /home/r188763/ldap/perl_modify.pl \' . $uid . \' \'. $parametro . \' \' . $valor ;
 my $result = system($comando);
 return $comando;
'
  LANGUAGE 'plperlu' VOLATILE SECURITY DEFINER;
```

# 6.4 - Codificação das funções de trigger

Para que as funções ldap_add e ldap_modify, codigicadas em Perl, sejam acionados a cada modificação na tabela de funcionários, são necessárias as chamadas funções de trigger e as triggers. Esses pequenos códigos devem ser desenvolvidos em psql.

As funções de trigger são responsáveis por manipular valores de registros que estão sendo acessados em determinada tabela. Através dessas funçõe é possível resgatar, por exemplo, o antigo valor de um campo recém alterado, assim como o seu novo valor.

Foram codificadas duas funções de trigger, uma para os eventos de inserção de dados e outra para as rotinas de alteração de dados.

```
CREATE OR REPLACE FUNCTION rh.dispara_transfer_proc_add()
  RETURNS trigger AS
'
 BEGIN

  PERFORM(postgres.ldap_add(NEW.nome,cargo,area,gerencia,cod_salario,email,fone1,fone2,local,rs,ccusto,login));

  RETURN NEW;
 END;

'
  LANGUAGE 'plpgsql' VOLATILE;
```

```
CREATE OR REPLACE FUNCTION rh.dispara_transfer_proc_modify()
 RETURNS trigger AS
'
 BEGIN

 PERFORM(postgres.ldap_modify(NEW.nome,2,3));

 RETURN NEW;
END;

'
 LANGUAGE 'plpgsql' VOLATILE;
```

# 6.4 - Codificação das triggers

Finalmente, a tabela de funcionários deve ser contemplada com triggers. As triggers são gatilhos, responsaveis por orientar o SGBD sobre o que fazer em determinados eventos de manipulação de dados na tabela. Neste projeto, apenas as inserções e os updates estão sendo monitorados. As triggers devem ser escritas em psql e devem conter basicamente: qual o evento a ser monitorado e o que fazer no caso do evento ocorrer.

```
CREATE TRIGGER funcionario_insert
 AFTER INSERT
 ON rh.funcionario
 FOR EACH ROW
 EXECUTE PROCEDURE rh.dispara_transfer_proc_add();
```

```
CREATE TRIGGER funcionario_update
 AFTER UPDATE
 ON rh.funcionario
 FOR EACH ROW
 EXECUTE PROCEDURE rh.dispara_transfer_proc_modify();
```

## 6.5 - Representação do funcionamento

Com todos os objetos já codificados, testados e implantados, o processo de atualização automática já está em funcionamento. As figuras 6.6 e 6.7 mostram uma representação do processo em funcionamento.

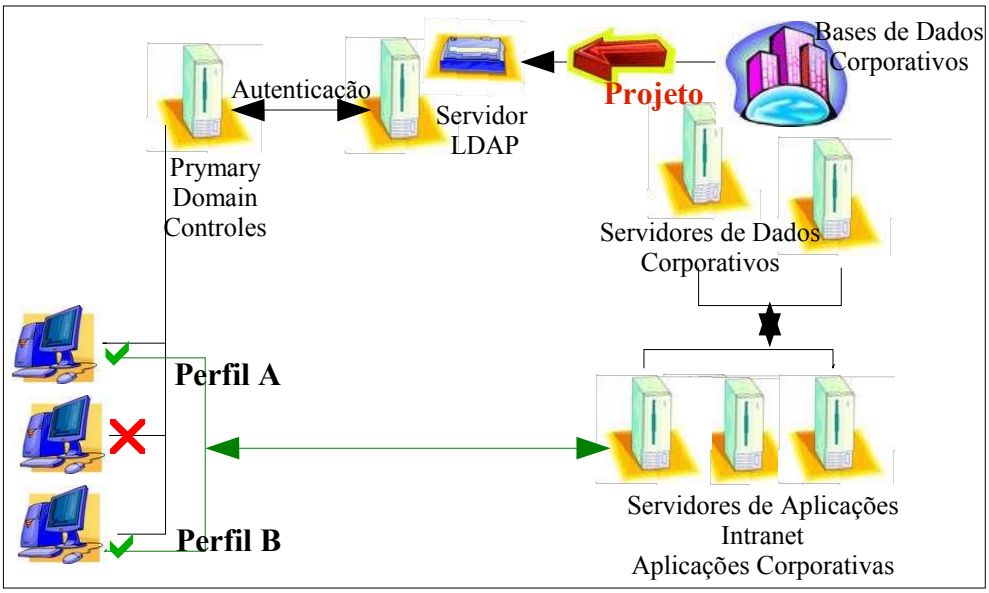


**Figura 6.6 -** Representação do funcionamento geral do projeto

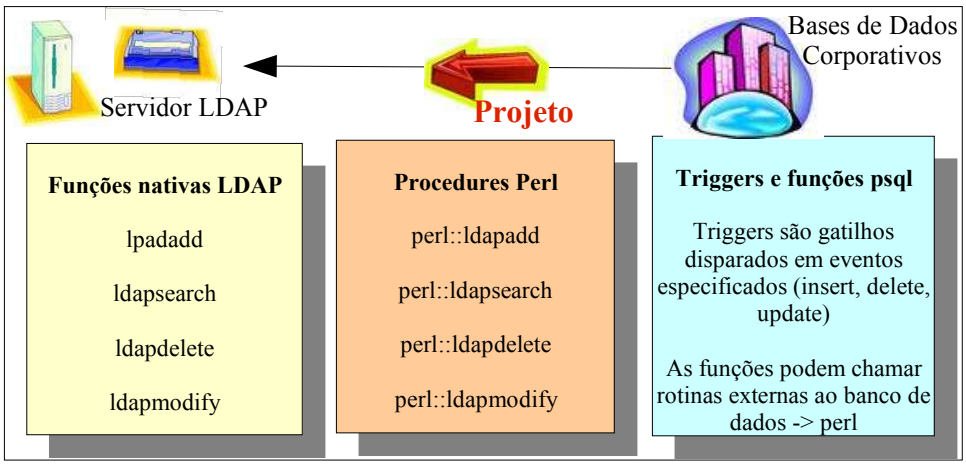


**Figura 6.7 -** Representação das etapas internas do projeto

## 6.6 - Resultados Obtidos

Todos os procedimentos descritos neste projeto foram implantados em ambiente de testes na Cia do Metropolitano de São Paulo, produzindo os resultados esperados em cada etapa de implantação.

Como resultado final, pode-se destacar avanços na administração da rede corporativa tais como:

- considerável diminuição no tempo destinado à manutenção da base LDAP;
- aumento da confiabilidade das informações armazenadas no LDAP;
- disponibilidade de um fácil desenvolvimento das interfaces de dados corporativas como intranet e portais;
- organização geral dos perfis de acesso às aplicações corporativas.

Também pode-se destacar avanços nos processos geridos pelo RH:

- diminuição de burocracia na comunicação entre o RH e a área de TI;
- disponibilização imediata dos recursos computacionais aos funcionários recém admitidos na empresa;
- segurança e proteção dos recursos computacionais de funcionários demitidos ou afastados.

A tarefa de administrar os procedimentos implantados (replicação de dados) passou a fazer parte das rotinas geridas pelo Administrador de Banco de Dados, profissional dedicado à administração das bases de dados da empresa. Este fato também pode ser visto como uma melhor organização das tarefas, uma

vez que dispensa o profissional administrador da rede da exaustiva tarefa de replicar manualmente informações já registradas em sistemas informacionais.

De forma geral, os resultados ultrapassaram os benefícios esperados inicialmente.

Atualmente, a Cia do Metrô analisa a implantação do processo em ambiente de produção, o que envolve a administração de mais de 800 máquinas em rede e de 8000 funcionários espalhados em 15 gerências.

Uma vez que a parte técnica-operacional já está desenvolvida e testada, o passo mais complexo se dá na direção da organização funcional administrativa dos dados envolvidos.Essa tarefa implica na definição dos profissionais envolvidos, na documentação das diretrizes do projeto, oficialização dos perfis de segurança e outras atividades inseridas nas implantação de um projeto desse vulto.

Uma vez finalizada essa tarefa, o prazo de implantação do projeto técnico é de 2 meses, incluindo o período de testes das rotinas, em paralelo com o procedimento atualmente utilizado.

# Capítulo 7

# Conclusão

A complexidade de uma solução geralmente reside na sua concepção e não na sua implantação. Este projeto é um exemplo disso. A organização das idéias, aliada ao conhecimento das ferramentas e técnicas disponíveis, torna possível a solução da grande maioria dos problemas propostos. A necessidade é o estopim, o raciocínio e o conhecimento os combustíveis. Não há como não chegar ao fogo.

Neste projeto, identificou-se a necessidade de melhoria do processo de atualização dos dados de usuários da rede corporativa.

Com análises mais profundas, a definição do alvo foi ampliada e clarificada. O escopo do projeto ficou definido como:

- manter a segurança de acesso aos recursos da rede corporativa;
- utilizar perfis de acesso baseados na situação atual dos funcionários;
- automatizar a atualização dos perfis de acesso.

Buscando esses resultados, conhecimentos foram adquiridos através de análises de produtos, pequisa de ferramentas, de linguagens de programação, de técnicas já disponíveis e de técnicas possíveis de serem desenvolvidas.

Com o escopo do projeto aliado aos conhecimentos necessários sobre as possibilidades reais, uma estratégia de implantação foi traçada. A estratégia de implantação deve ser flexível, reavaliada e remodelada durante a fase de testes. Isso ocorre devido a existência de possíveis enganos nas avaliações iniciais ou

erros de interpretação de funcionalidades, dificuldades não previstas, etc.

Na fase de testes, os conhecimentos adquiridos são “colocados à prova” e consolidados mediante o sucesso de cada módulo da implantação.

O resultado do projeto é, então, comparado ao escopo inicial e, se todos os requisitos foram contemplados, o projeto é considerado terminado com sucesso. Nessa ocasião, o status do projeto é elevado a solução, e pode ser compartilhado com outras pessoas e situações que possuam necessidades semelhantes.

O processo de análise e solução de problemas é um dom exclusivo do ser humano e é daí que ocorre a evolução da nossa espécie. Foi assim desde as necessidades essenciais, como o fogo e a roda, passsando pelo desenvolvimento das ciências como a matemática e a física e chegando às grandes revoluções tecnológicas dos dias de hoje.

A filosofia surgida com o movimento do software livre revive esta evolução a cada novo desenvolvimento. O conhecimento adquirido será melhor empregado se for de propriedade de todos. Assim, ninguém precisa reinventar a roda para produzir um automóvel ou reinventar a fórmula de Báskara para projetar um edifício.

Os softwares nada mais são que invenções, descobertas e organizações de processos manuais. São extensões do intelecto humano em evolução, contribuindo com passos para renovações e aperfeiçoamento de ideias. O compartilhamento e a integração de conhecimentos é essencial para a evolução dos processos.

Este projeto expõe e disponibiliza uma solução barata e eficiente para a solução de um problema de integração específico, entre as bases corporativas e as bases de autenticação de uma empresa. A expectativa, além dos benefícios implícitos à solução, é colaborar no crescimento e na evolução dos processos de informações baseados em softwares livres.

# Anexo 1

```
<?php

/*
 *                The phpLDAPadmin config file
 *
 *   This is where you customize phpLDAPadmin. The most important
 *   part is immediately below: The "LDAP Servers" section.
 *   You must specify at least one LDAP server there. You may add
 *   as many as you like. You can also specify your language, and
 *   many other options.
 *
 */

// Your LDAP servers
$i=0;
$servers = array();
$servers[$i]['name'] = 'SA15METRO1';    /*  A convenient name that will appear in
                                        the tree viewer */
$servers[$i]['host'] = 'sa15metro1';  /*  Examples: 'ldap.example.com',
                                        'ldaps://ldap.example.com/'
                                        Note: Leave blank to remove it from the list
                                        of servers in the tree viewer*/
$servers[$i]['base'] = 'o=MetroSP,c=BR'; /*  The base DN of your LDAP server. Leave this
                                        blank to have phpLDAPadmin
                                        auto-detect it for you. */
$servers[$i]['port'] = 389;                /*  The port your LDAP server listens on
                                        (no quotes) */
$servers[$i]['auth_type'] = 'config';       /*  2 options: 'form': you will be prompted, and
                                        a cookie stored with your login dn and
                                        password. 'config': specify your login dn
                                        and password here. In both cases, use caution! */
$servers[$i]['login_dn'] = 'cn=root,o=MetroSP,c=BR';
                                    /*  For anonymous binds, leave the
                                        login_dn and login_pass blank */
$servers[$i]['login_pass'] = 'secret';      /*  Your password (only if you specified 'config'
                                        for 'auth_type' */
$servers[$i]['tls'] = false;               /*  Use TLS to connect. Requires PHP 4.2 or newer */
$servers[$i]['default_hash'] = 'crypt';     /*  Default password hashing algorith;
                                        One of md5, ssha, sha, md5crpyt, smd5, blowfish or
                                        leave blank for now default algorithm. */
$servers[$i]['login_attr'] = 'dn';          /*  If you specified 'form' as the auth_type above,
                                        you can optionally specify here an attribute
                                        to use when logging in. If you enter 'uid',
                                        then login as 'dsmith', phpLDAPadmin will
                                        search for uid=dsmith and log in as such. Leave
                                        blank or specify 'dn' to use full DN for
                                        logging in .*/
$servers[$i]['read_only'] = false;          /*  Specify true If you want phpLDAPadmin to not
                                        display or permit any modification to the
                                        LDAP server. */
$servers[$i]['enable_auto_uid_numbers'] = false;
                                    /*  This feature allows phpLDAPadmin to
                                        automatically determine the next
                                        available uidNumber for a new entry. */
$servers[$i]['auto_uid_number_mechanism'] = 'search';
                        /* The mechanism to use when finding the next available uidNumber.
                           Two possible values: 'uidpool' or 'search'. The 'uidpool'
                           mechanism uses an existing uidPool entry in your LDAP server
                           to blindly lookup the next available uidNumber. The 'search'
                           mechanism searches for entries with a uidNumber value and finds
                            the first available uidNumber (slower). */
```

```
$servers[$i]['auto_uid_number_search_base'] = 'o=MetroSP,c=BR';
                        /* The DN of the search base when the 'search'
                           mechanism is used above. */
$servers[$i]['auto_uid_number_min'] = 1000;
                        /* The minimum number to use when searching for the next
                           available UID number (only when 'search' is used for
                           auto_uid_number_mechanism' */
$servers[$i]['auto_uid_number_uid_pool_dn'] = 'o=MetroSP,c=BR';
                        /* The DN of the uidPool entry when 'uidpool'
                           mechanism is used above. */


// If you want to configure additional LDAP servers, do so below.
/*
$i++;
$servers[$i]['name'] = 'sa15metro1';
$servers[$i]['host'] = 'sa15metro1';
$servers[$i]['base'] = 'o=MetroSP,c=BR';
$servers[$i]['port'] = 389;
$servers[$i]['auth_type'] = 'config';
$servers[$i]['login_dn'] = 'cn=root,o=MetroSP,c=BR';
$servers[$i]['login_pass'] = 'secret';
$servers[$i]['tls'] = false;
$servers[$i]['default_hash'] = 'crypt';
$servers[$i]['login_attr'] = 'dn';
$servers[$i]['read_only'] = false;
$servers[$i]['enable_auto_uid_numbers'] = false;
$servers[$i]['auto_uid_number_mechanism'] = 'search';
$servers[$i]['auto_uid_number_search_base'] = 'o=MetroSP,c=BR';
$servers[$i]['auto_uid_number_min'] = 1000;
$servers[$i]['auto_uid_number_uid_pool_dn'] = 'o=MetroSP,c=BR';
*/
// If you want to configure more LDAP servers, copy and paste the above (including the "$i++;")

// The temporary storage directory where we will put jpegPhoto data
// This directory must be readable and writable by your web server
$jpeg_temp_dir = "/tmp";       // Example for Unix systems
//$jpeg_temp_dir = "c:\\temp"; // Example for Windows systems

/**                         **/
/**   Appearance and Behavior  **/
/**                         **/

// The language setting. If you set this to 'auto', phpLDAPadmin will
// attempt to determine your language automatically. Otherwise, available
// lanaguages are: 'ct', 'de', 'en', 'es', 'fr', 'it', 'nl', and 'ru'
// Localization is not complete yet, but most strings have been translated.
// Please help by writing language files. See lang/en.php for an example.
$language = 'auto';

// Set to true if you want to draw a checkbox next to each entry in the tree viewer
// to be able to delete multiple entries at once
$enable_mass_delete = false;

// Set to true if you want LDAP data to be displayed read-only (without input fields)
// when a user logs in to a server anonymously
$anonymous_bind_implies_read_only = true;

// If you used auth_type 'form' in the servers list, you can adjust how long the cookie will last
// (default is 0 seconds, which expires when you close the browser)
$cookie_time = 0; // seconds

// How many pixels wide do you want your left frame view (for the tree browser)
$tree_width = 320; // pixels
```

```
// How long to keep jpegPhoto temporary files in the jpeg_temp_dir directory (in seconds)
$jpeg_tmp_keep_time = 120; // seconds

// Would you like to see helpful hint text occacsionally?
$show_hints = true; // set to false to disable hints

// When using the search page, limit result size to this many entries
$search_result_size_limit = 50;

/**                          **/
/** Simple Search Form Config **/
/**                          **/

// Which attributes to include in the drop-down menu of the simple search form (comma-separated)
// Change this to suit your needs for convenient searching. Be sure to change the correlating
// list below ($search_attributes_display)
$search_attributes = "uid, cn, gidNumber, objectClass, telephoneNumber, mail, street";

// This list correlates to the list directly above. If you want to present more readable names
// for your search attributes, do so here. Both lists must have the same number of entries.
$search_attributes_display = "User Name, Common Name, Group ID, Object Class, Phone Number, Email, Address";

// The list of attributes to display in each search result entry summary
$search_result_attributes = "dn, cn";

/**                                  **/
/** User-friendly attribute translation     **/
/**                                  **/

$friendly_attrs = array();

// Use this array to map attribute names to user friendly names. For example, if you
// don't want to see "facsimileTelephoneNumber" but rather "Fax".

$friendly_attrs[ 'facsimileTelephoneNumber' ] =        'Fax';
$friendly_attrs[ 'telephoneNumber' ]  =                'Phone';

/**                                  **/
/** Hidden attributes                      **/
/**                                  **/

// You may want to hide certain attributes from being displayed in the editor screen
// Do this by adding the desired attributes to this list (and uncomment it). This
// only affects the editor screen. Attributes will still be visible in the schema
// browser and elsewhere. An example is provided below:

//$hidden_attrs = array( 'jpegPhoto', 'objectClass' );

?>
```

# Anexo 2

ufla1.html

```
<!--
ufla1.html
função: frame superior da tela - contém logos e figuras relativas ao portal
-->

<html>
<head>
<title>UFLA - PHP&LDAP</title>
<meta name="author" content="Administracao de Redes LINUX">
</head>
<body bgcolor="#FFFFFF" text="#000000" leftmargin="0" topmargin="0" marginwidth="0" marginheight="0">
<table width="100%" border="0" cellspacing="0" cellpadding="0" height="43" align="center">
  <tr bgcolor="#a9c4e0">
     <td colspan="2" height="44" bgcolor="#a9c4e0" valign="top"><img src="teleduc_logo.gif" width="120" height="40"
align="top"></td>
    <td height="44" valign="top" bgcolor="#a9c4e0">
     </td>
    <td width="180" height="44" valign="top">
      <div align="right"><img src="ldapworm.gif" width="202" height="40" alt="LDAP"></div>
      <div align="right"></div>
    </td>
  </tr>
</table>
</body>
</html>
```

## ufa1.php

```
<!--
ufla1.php
função: formata a tela em tabela, organizando o espaço para a
exibibição de caixas de texto para entrada dos parâmetros "usuário" e "senha".
Após o fornecimento dos parâmetros, os mesmos são repassados para o form ufla2.php
-->

<html>
<head>
<title>index_UFLA</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<script language="JavaScript">
</script>
</head>

<body bgcolor="#FFFFFF" text="#000000" topmargin="0">
<br>
<table width="98%" border="0" cellspacing="0" cellpadding="0" align="center">
  <tr>
    <td colspan="3" height="10"></td>
  </tr>
  <tr>
    <td width="629" height="98" valign="top">
      <div align="center">
        <p align="justify"><font face="Verdana, Arial, Helvetica, sans-serif" size="2">Este e um exemplo de aplicação em
PHP
        que executa autenticacao em uma base LDAP e traz informacoes relativas ao usuario logado.</font></p>
        <p align="justify"><font face="Verdana, Arial, Helvetica, sans-serif" size="2">
          Foi desenvolvido por Marcia Abade no curso de Administracao de Redes LINUX
          como exemplo de implementacao da tecnologia LDAP.</font></p>
      </div>
    </td>
    <td width="24" height="98" valign="top"> </td>
    <td width="95" height="98" valign="top"><img src="ldaplogo.gif" width="120" height="84"></td>
  </tr>
</table>
<br>
<div align="left">

  <table width="98%" border="0" cellpadding="0" cellspacing="0" align="center">

    <tr>
          <td width="1%" height="16" valign="top"><font face="Verdana, Arial, Helvetica, sans-serif" size="1"><img
src="mw_quadrado.gif" width="12" height="12"></font></td>
      <td width="32%" height="16" valign="top"><font face="Verdana, Arial, Helvetica, sans-serif" size="1"><b>Usuário /
Password: </b></font></td>
    </tr>
    <tr>
      <td width="1%" height="10"> </td>
      <td width="32%" height="10" valign="top">
        <form action="ufla2.php" method="post">
          <input type="text" name="uid" maxlength="10" size="6">
               <font face="Verdana, Arial, Helvetica, sans-serif" size="2">
                                    <input type="password" name="userpass" maxlength="10" size="6">
          <input type=image src="ok_mm.gif" value="OK" name="imgc" width="18" height="15">
          </font><br>
        </form>
      </td>
      <td height="10" width="1%"> </td>

      <td width="1%" height="10"> </td>
```

```
    </tr>
  </table>
  <br>

  <br>
  <table width="98%" border="0" cellpadding="0" cellspacing="0" align="center">
    <tr>
      <td height="7"><img src="mw_base.gif" width="697" height="34"></td>
    </tr>
  </table>

</div>
</body>
</html>
```

ufla2.php

```
<!--
ufla2.php
função: recebe os parâmetros "usuário" e "senha" e, com este valores processa a validação
do usuário na base LDAP
Neste exemplo utilizamos como exemplos os nomes de servidores/usuários utilizados como
teste no desenvolvimento. Eles devem ser alterados conforme os parâmetros da sua
implementação. Siga os comentário para adaptá-los.
-->


<?
//indique aqui o nome ou endereço IP do seu servidor LDAP
$HOST="sa15metro1";

//checa se o servidor está respondendo e armazena conexão em $ds
$ds=ldap_connect($HOST) or die ("Nao foi Possivel conectar ao Servidor");

//concatena parâmetros recebidos na string padrão de DN e armazena em $ldapdn
$uid = $_POST["uid"];
$userpass = $_POST["userpass"];
$ldaprdn  = "uid=$uid,ou=Usuarios de Rede,o=MetroSP,c=BR";
$ldappass = "$userpass";

//Efetua o bind com os parâmetros recebidos
ldap_bind($ds, $ldaprdn, $ldappass) or die("<b><center>Erro na Conexao com o
Servidor...</center><br><center>Contatar o Administrador do Sistema.</center></b>");

//Checa se a autenticação ocorreu com sucesso e exibe resultado
if($ds){
echo "<b><center>Resultado da autenticação no LDAP </center></b>";
echo "<b><center>Conectou com sucesso!</center></b>";
echo "<b><center></b>";
echo "<br>";

//Realiza busca de atributos - feito em bloco separado pois pode ser
//um ramo do LDAP administrado independentemente do primeiro
//Define árvore que será varrida
$dn = "ou=Usuarios de Rede,o=MetroSP,c=BR";

//define filtro - no caso busca por uid
$filter="(uid=$uid)";

//define valores de quais atributos será retornado pela busca
//foi utilizado o atributo member para definir as aplicacoes as quais o uid possui
//acesso. Para cada aplicacao, um valor em "member"
$justthese = array("uid","cn","sn","mail","title","employeeType","member");

//realiza busca
$sr=ldap_search($ds, $dn, $filter, $justthese);

//checa resultados
$info = ldap_get_entries($ds, $sr);

//exibe atributos encontrados na busca
//aqui, por motivo de exemplificação são consideradas 10 aplicações possíveis
//para cada uid. O array é infinito. É possível codificar um loop para um
//numero indefinido de aplicacoes

print "<b>uid: </b>" .      $info[0]["uid"][0] . "<br>";
print "<b>Nome: </b>" .                $info[0]["cn"][0] . "<br>";
print "<b>Cargo: </b>" .    $info[0]["title"][0] . "<br>";
print "<b>Situacao: </b>" . $info[0]["employeeType"][0] . "<br>";
```

```
print "<b>Endereco de email: </b>" .                    $info[0]["mail"][0] . "<br>";
print "<p><b>--------------------------Aplicações disponíveis ---------------------</b><p>";
if ($info[0]["member"][0] <> '')
                {print substr($info[0]["member"][0],4) . "<br>";}
if ($info[0]["member"][1] <> '')
                {print substr($info[0]["member"][1],4) . "<br>";}
if ($info[0]["member"][2] <> '')
                {print substr($info[0]["member"][2],4) . "<br>";}
if ($info[0]["member"][3] <> '')
                {print substr($info[0]["member"][3],4) . "<br>";}
if ($info[0]["member"][0] <> '')
                {print substr($info[0]["member"][4],4) . "<br>";}
if ($info[0]["member"][1] <> '')
                {print substr($info[0]["member"][5],4) . "<br>";}
if ($info[0]["member"][2] <> '')
                {print substr($info[0]["member"][6],4) . "<br>";}
if ($info[0]["member"][3] <> '')
                {print substr($info[0]["member"][7],4) . "<br>";}
if ($info[0]["member"][0] <> '')
                {print substr($info[0]["member"][8],4) . "<br>";}
if ($info[0]["member"][1] <> '')
                {print substr($info[0]["member"][9],4) . "<br>";}

echo "</center></b>";

}

//fim
?>
```






# Referências Bibliográficas

[Fernando Jose Barbin Laurindo (2002)] FERNANDO JOSE BARBIN LAURINDO. **Tecnologia da Informação: Eficácia nas Organizações.** 1ª ed. São Paulo: Editora Futura, 2002.

[Paulo L.Geus e Emilio T. Nakamura (2003)] PAULO L.GEUS E EMILIO T. NAKAMURA. **Seguranca de Redes - Edicao revista e ampliada.** 1ª ed. São Paulo: Editora Futura, 2003.

[Jose Carlos Cordeiro Martins (2003)] JOSE CARLOS CORDEIRO MARTINS. **Gestao de Projetos de Seguranca da Informacao.** 1ª ed. São Paulo: Editora BRASPORT , 2003.

[Bernard H. Boar (2003)] BERNARD H. BOAR. **Tecnologia da Informação.** 2ª ed. São Paulo: Editora Futura, 2003.

[David M. Kroenke (1999)] DAVID M. KROENKE. **Banco de Dados - Fundamentos, Projeto e Implementação.** 6ª ed. São Paulo: Editora LTC , 1999.

[Felipe Nery R. Machado (1996)] FELIPE NERY R. MACHADO. **Projeto de Banco de Dados - Uma Visão Prática.** 10ª ed. São Paulo: Editora Érica, 1996.

[William Pereira Alves (2004)] WILLIAM PEREIRA ALVES. **Fundamentos de Bancos de Dados.** 1ª ed. São Paulo: Editora Érica , 2004.

[C. J. Date (2004)] C. J. Date.**Introdução a Sistemas de Banco de Dados.** 1ª ed. Rio de Janeiro:Editora Campus, 2004.

[Denise Lemes Fernandes Neves (2002)] DENISE LEMES FERNANDES NEVES. **PostgreSQL: Conceitos e Aplicações** 1ª ed. São Paulo: Editora Érica , 2002.

[Álvaro Pereira Neto (2003b)] ÁLVARO PEREIRA NETO. **PostgreSQL: Técnicas Avançadas - Versões Open Source 7.x** 2ª ed. São Paulo: Editora Érica , 2003.

[Larry Wall, Tom Christiansen, Jon Orwant (2001)] LARRY WALL, TOM CHRISTIANSEN & JON ORWANT. **Programação Perl.** Tradução da 3. ed. Rio de Janeiro:Editora Campus, 2001.

[Rosângela Hickson(2002)] ROSÂNGELA HICKSON. **Aprenda a programar em C, C++ e C#**. Rio de Janeiro:Editora Campus,2002.

[Brian Arkills (2003)]BRIAN ARKILLS. **LDAP Directories Explained**. New York: Editora Addison Wesley, 2003

[Ewald Geschwinde, Hans-juergen Schoenig (2002)] EWALD GESCHWINDE, HANS-JUERGEN SCHOENIG. **PHP and PostgreSQL Advanced Web Programming.** Massashusets: Editora Sams, 2002

[Álvaro Pereira Neto (2003)] ÁLVARO PEREIRA NETO. **PostgreSQL - Técnicas Avançadas.** São Paulo: Editora Érica, 2003

[Harvey H. Deitel, Paul J. Deitel, Ben Wiedermann, Jonathan P. Liperi (2002)] HARVEY M. DEITEL, PAUL J. DEITEL, BEN WIEDERMANN, JONATHAN P. LIPERI. **Python How to Program**. Toronto: Editora Prentice Hall ,2002

[The Mozilla Organization (2004)] THE MOZILLA ORGANIZATION em SITE OFICIAL MOZILA. **PerLDAP: Source Code Release.** Disponível em: http://www.mozilla.org/directory/perldap.html Acesso em maio/2004

[PostgreSQL Global Development Group (2003)] POSTGRESQL GLOBAL DEVELOPMENT GROUP em SITE OFICIAL POSTGRESQL. **Official Documentation.** Disponível em: http://www.postgresql.org Acesso em maio/2004

[OpenLDAP Foundation (2004)] OPENLDAP FOUNDATION em SITE OFICIAL OPENLDAP. **Manual Pages.** Disponível em: http://www.openldap.org/software/man.cgi Acesso em maio/2004

[Graham Barr (2003)] GRAHAM BARR em SITE OFICIAL PERL LDAP. **Documentation.** Disponível em: http://ldap.perl.org/ Acesso em maio/2004

[Paul Dwerryhouse (2003)] PAUL DWERRYHOUSE em LINUX JOURNAL. **HOWTO: An Introduction to perl-ldap.** Disponível em: http://www.linuxjournal.com/article.php?sid=7086 Acesso em maio/2004

[Graham Barr (2004)] GRAHAM BARR em CPAN. **Documentation.** Disponível em: http://search.cpan.org/~gbarr/perl-ldap/ Acesso em maio/2004

[Sleepycat Software (2004)] Sleepycat Software em Página Oficial. **Documentation.** Disponível em: http://www.sleepycat.com/ Acesso em maio/2004

[FSF (2001)] FREE SOFTWARE FOUNDATION. **Licenças de Software Livre.** Disponível em http://www.gnu.org/licenses/licenses.pt.html. Atualizado em 15/09/2001. Acesso em janeiro de 2004.

[OpenLDAP Licence (2004)] OPENLDAP FOUNDATION em SITE OFICIAL OPENLDAP. **Licence Pages.** Disponível em: http://www.openldap.org/software/release/license.html Acesso em maio/2004