Dov Bigio
Rony Martins Sakuragui
Sérgio Zular Zveibil

Método de Programação Para Aplicações para Dispositivos Portáteis dotados do Palm OS.

Projeto de Formatura da disciplina PCS 0502 - Laboratório de Projeto de Formatura II.
Departamento de Engenharia de Computação e Sistemas Digitais da Escola Politécnica da Universidade de São Paulo.

São Paulo
2001

Dov Bigio
Rony Martins Sakuragui
Sérgio Zular Zveibil

Método de Programação Para Aplicações para Dispositivos Portáteis dotados do Palm OS.

Projeto de Formatura da disciplina PCS 0502 - Laboratório de Projeto de Formatura II.
Departamento de Engenharia de Computação e Sistemas Digitais da Escola Politécnica da Universidade de São Paulo.

Curso de Engenharia de Eletricidade - opção Computação

Orientador: Prof. Wilson Ruggiero

São Paulo
2001

Bigio, Dov; Sakuragui, Rony Martins; Zveibil, Sérgio Zular

Método de Programação Para Aplicações para Dispositivos Portáteis
dotados do Palm OS.
100p.

Projeto de Formatura da disciplina PCS 0502 - Laboratório de Projeto de Formatura II. Departamento de Engenharia de Computação e Sistemas Digitais da Escola Politécnica da Universidade de São Paulo.

1. Computação 2. PDA

A todos aqueles que nos ajudaram a desenvolver este projeto. A todos aqueles que se interessam por desenvolvimento de aplicações para a Internet, Redes de Computadores e PDAs. Esperamos que este trabalho seja útil.

AGRADECIMENTOS

Gostaríamos de agradecer ao Professor Wilson Ruggiero, que, apesar do pouco contato que tivemos ao longo do desenvolvimento projeto, foi nosso orientador, e nos ajudou a ter a idéia de desenvolver este projeto.

Agradecemos também a todos os nossos professores do Depto. de Engenharia da Computação da Escola Politécnica da Universidade de São Paulo, que nos ensinaram conceitos que serviram de base para a resolução dos diversos obstáculos que encontramos no decorrer deste projeto.

Agradecemos também ao pessoal do LARC (Laboratório de Arquitetura e Redes de Computadores) da Escola Politécnica, visto que nos disponibilizaram o servidor e os softwares necessários para este projeto.

Além disso, cabem especiais agradecimentos às nossas famílias e amigos, que acompanharam de perto as noites e fins de semana que passamos em claro buscando soluções para otimizar o desenvolvimento deste projeto.

RESUMO

A rede de dados, em sua concepção, tem a função básica de transportar os dados dos usuários para o servidor, e vice-versa, como numa aplicação tradicional de cliente-servidor. Os dados contém as informações necessárias às aplicações como, por exemplo, nome de usuário e senha. Podemos separar o sistema de comunicação em duas partes principais: a interface com web e a interface wireless.

Fluxo de Dados

A interface com a web consiste basicamente da conexão do servidor com a Internet. Ela faz parte do desenvolvimento do Servidor em si e não é o objeto de discussão neste ítem, uma vez que o "Web Server" utilizado no projeto tem todas as funcionalidades para conexão do usuário ao servidor pela Internet.

A interface wireless é responsável pela comunicação do usuário com o Servidor através de sua Palm Pilot. A proposta do grupo foi criar uma rede totalmente *wireless* (sem cabos). Como não há nenhuma tecnologia "de facto" para implementar esta rede, fez-se necessário o seu desenvolvimento nos mais diversos níveis, desde a decisão sobre a tecnologia de enlace até criação de um protocolo de comunicação. Esta rede será o objeto de discussão do projeto.

SUMÁRIO

ÍNDICE DE FIGURAS

INTRODUÇÃO

Este Projeto de Formatura é nosso Projeto de Conclusão de Curso de Engenharia Elétrica com Ênfase em Computação na Escola Politécnica.

Seu principal objetivo é consolidar conceitos estudados ao longo destes cinco (ou mais) anos de faculdade, aplicando-os na elaboração de um sistema de computação.

Este projeto é, antes de tudo, um projeto de integração de sistemas. Para implementar o sistema proposto o grupo separou o desenvolvimento do sistema em três módulos principais: a Palm, a Rede e o Servidor. Uma vez estando estes módulos funcionando, o trabalho final foi colocá-los juntos, resolvendo os eventuais problemas de interface.

Na prática, vários desafios foram encontrados para realizar esta integração. O primeiro desafio é "começar certo"; isto é, para ser possível desenvolver módulos do sistema em separado é necessário, antes de começar o trabalho, especificar exatamente qual a função de cada um deles e, principalmente, como será a interface de cada um com os demais módulos do sistema.

Acontece que, como previsto pelo grupo, e como se provou durante os trabalhos, é inevitável uma revisão desta especificação conforme os problemas e novas necessidades vão sendo encontrados. Surge então um segundo desafio: como desenvolver um sistema de forma que, sendo necessárias mudanças, estas não ocasionem um total retrabalho no que já foi feito?

A resposta é simples: modularização. Não uma modularização em nível de sistemas, como mencionado antes, mas sim modularização nos componentes de baixo nível, onde cada pequeno módulo funcional deve ser independente do outro, permitindo mudanças na sua estrutura interna sem que os demais sejam afetados. Isto é especialmente importante na fase final do projeto, pois como cada módulo do sistema foi desenvolvido totalmente em separado, é inevitável que pequenos ajustes sejam necessários durante sua integração.

Essencial também para o andamento do projeto foi a definição de uma metodologia de testes que permitisse que cada módulo estivesse funcionando corretamente antes de 'conectá-lo' aos demais. Assim, foi possível que os desenvolvimentos do sistema no gateway, na Palm e no servidor (como será explicado posteriormente) pudessem ser totalmente independentes até a maior parte do projeto.

Apesar de recursos escassos, e de tempo não integral de dedicação ao projeto, acreditamos que foi possível desenvolver um projeto completo de qualidade. Ainda assim, visando um desenvolvimento posterior deste projeto, incluímos neste documento uma seção intitulada Próximos Passos, que direciona os caminhos que devem ser seguidos para que este projeto possa ser utilizado comercial ou até mesmo academicamente.

MEMBROS DA EQUIPE

- *Dov Bigio (dovb@terra.com.br)*, atualmente trabalhando como estagiário na Accenture, uma empresa de consultoria na área de tecnologia. Formatura prevista para Dezembro de 2001.

- *Rony Martins Sakuragui (rsakuragui@yahoo.com.br)*. Formatura prevista para Junho de 2002.

- Sérgio Zular Zveibil (sergioz@prolan.com.br), atualmente trabalhando como estagiário na ProLan, uma empresa que trabalha com integração de redes de computadores. Formatura prevista para Dezembro de 2001.

OBJETIVO

Encontrar uma metodologia de trabalho simples para o uso das tecnologias para a o desenvolvimento de aplicações de troca de dados entre computadores desktop e dispositivos de computação pessoal portáteis (PDA).

O objetivo do projeto em si não foi o desenvolvimento de aplicações aplicando esta tecnologia (por isso foi desenvolvido apenas um sistema simples de troca de mensagens), mas sim o estudo de como utilizá-la de maneira eficiente.

APRESENTAÇÃO

O projeto consiste no desenvolvimento de aplicações que se utilizem dos recursos de comunicação a distância disponíveis nos dispositivos *handheld* baseados no sistema operacional *PalmOS*, como os modelos *Palm Pilot*, da Palm Inc. e *Visor*, da Handspring Inc. As aplicações estarão voltadas para os dispositivos compatíveis com os modelo de Palm que aceitem adaptadores para comunicação wireless. (Inicialmente, desejava-se utilizar o padrão 802.11, para redes wireless; no entanto, devido a dificuldade de se encontrar adaptadores que permitissem conectar uma interface deste padrão no aparelho Palm V – não dotado de entrada para placas de expansão – prefiriu-se adotar a interface infra-vermelha já disponível no PDA).

As aplicações consistirão na troca de dados entre os dispositivos portáteis e um servidor de dados ligado à Internet, permitindo a o acesso a uma base dados e a troca de informações. Além disso, o trabalho será complementado com uma interface web para atualização e consulta aos dados, permitindo que os mesmos dados sejam manipulados das duas maneiras.

Para exemplificar o problema que desejou-se resolver ao ser proposto este projeto, seguem dois casos de uso possíveis.

Exemplo 1:

Considerando um usuário de PDA que seja também usuário de um sistema de notícias pela Palm, atualmente, ele teria o noticiário atualizado em sua Palm toda vez que fizesse um HotSync[1], através de um módulo instalado em seu PC, que atualizaria as notícias da Internet.

Utilizando a tecnologia proposta, a agência provedora das notícias poderia disponibilizar terminais em pontos estratégicos (aeroportos, shoppings, etc), onde os usuários poderiam, através da interface infra-vermelha, atualizar o módulo de notícias de sua Palm, sem a necessidade de estar em seu próprio computador.

[1] HotSync: Sistema de sincronização de dados entre a Palm Pilot e o PC.

Exemplo 2:

Um site que possua um sistema de e-mail poderia oferecer a seus usuários a possibilidade de escrever e enviar mensagens através de suas Palm Pilots, e fazê-lo a partir de terminais espalhados em quaisquer pontos estratégicos da cidade, fazendo com que a Palm envie os e-mails diretamente, sem passar pelo programa de e-mail do usuário, como é feito com o processo de HotSync padrão.

A equipe optou por desenvolver uma aplicação-teste do uso da tecnologia baseada no segundo exemplo apresentado acima.

Observação:

Neste documento, são usadas as nomenclaturas: Palm, Palm Pilot, dispositivo handheld e PDA para referir-se aos dispositivos portáteis baseados no sistema operacional Palm OS, versão 3.0 ou superior.

FASES DO DESENVOLVIMENTO

Foram identificadas algumas fases necessárias para análise e implementação do sistema. As fases não necessitam ser e nem foram desenvolvidas na ordem que se segue, tendo sido desenvolvidas em paralelo na maioria dos casos (com excessão da integração). Subseqüêntemente, estas fases serão melhor descritas e aprofundadas. São elas:

- *Elaboração da estrutura da base dados*: detalhamento dos dados relevantes para o sistema, desenvolvimento do diagrama de entidade-relacionamento, estruturação em forma de tabelas, escolha do sistema gerenciados de banco de dados e implementação;

- *Desenvolvimento de componentes*: esta fase consiste no desenvolvimento de bibliotecas de componentes úteis para o projeto. Os componentes desenvolvidos nesta fase serão úteis na fase do desenvolvimento do sistema de comunicação, através da arquitetura de objetos distribuídos da Microsoft.

- *Desenvolvimento do site*: planejamento do site que oferecerá as interfaces necessárias para todas as aplicações através da web, design, desenvolvimento da lógica utilizando-se de *server-side scripts* para acesso à base de dados.

- *Interface Palm*: planejamento e desenvolvimento das interfaces necessárias no dispositivo handheld, utilizando-se da linguagem de programação adequada e do desenvolvimento de pequenos bancos de dados no mesmo.

- *Comunicação e formatação de dados*: planejamento e desenvolvimento de software capaz de interpretar requisições vindas da Palm através do dispositivo de Rx/Tx, fazer as operações necessárias no banco de dados, e retornar o resultado para a Palm, através deste dispositivo.

- *Projeto da Arquitetura da Rede*: planejamento da arquitetura da rede, suas especificações, protocolos de comunicação e obtenção de equipamentos.

ARQUITETURA DE REDE

Inicialmente, a idéia original era a utilização de dispositivos de rádio para realizar a comunicação das Palms com o Servidor. De fato, muito esforço foi feito no primeiro semestre no sentido de achar um hardware que permitisse esta comunicação.

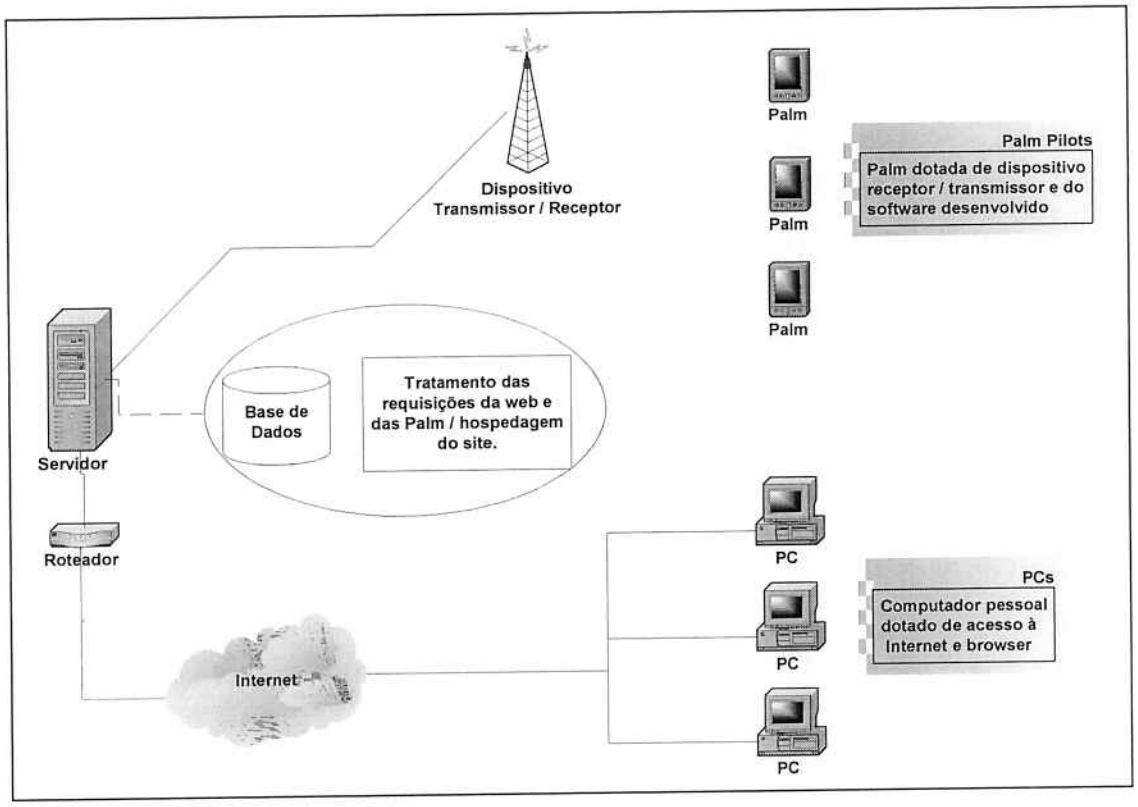


Figura 1: Arquitetura Geral do Sistema

Após análises sobre a viabilidade do projeto, facilidade de desenvolvimento e obtenção de equipamentos necessários (não foi encontrado um dispositivo que fizesse o modelo Palm V, utilizado para o desenvolvimento, transmitir e receber via rádio), foi adotada a seguinte arquitetura (ver Figura 2) para o sistema em questão.

Decidiu-se que a comunicação com os PDAs se daria através de uma interface infra-vermelha, conectada a cada um dos gateways (no caso prático deste projeto, apenas um), que estarão, por sua vez, conectados ao servidor através de uma rede wireless padrão 802.11.

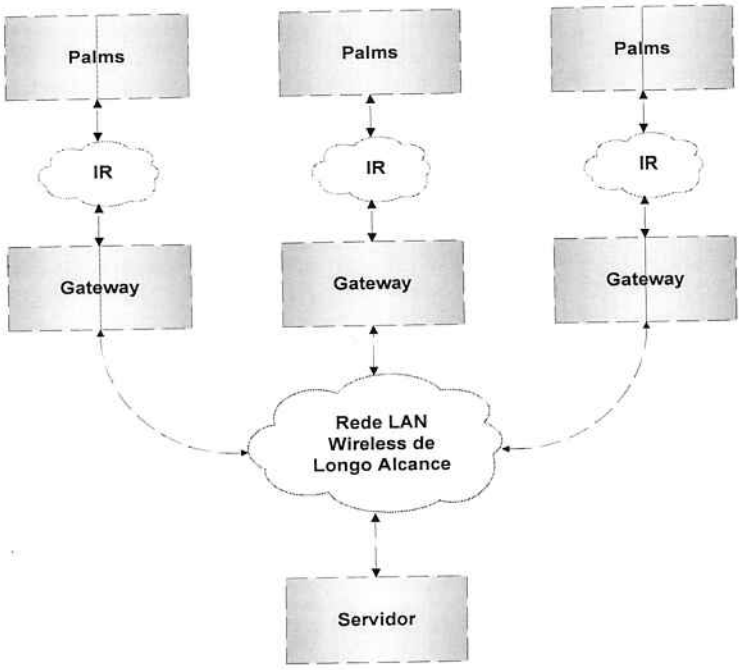


Figura 2: Arquitetura do Sistema

A nova arquitetura, utlizando o infravermelho, é baseada na utilização de "Gateways", ou portões, capazes de se comunicar tanto com as Palms quanto com o servidor central. Cada Gateway deve ser capaz de se comunicar através de uma interface infravermelha com as Palms, e se comunicar através de padrões de rede com o Servidor. Esta nova arquitetura, além de viabilizar o projeto, apresentou várias vantagens em relação à primeira proposta:

- A comunicação de rádio com o servidor pode ser feita através do padrão de rede Ethernet Wireless no padrão 802.11b.
- A comunicação infravermelho utiliza um padrão de mercado bem conhecido, o IrDA, com os drivers e interfaces padrão para a comunicação.
- Modularização da rede: cada Gateway é um elemento de rede autônomo e independente. Além disso, pode-se utilizar vários Gateways para aumentar a capilaridade da rede, ou seja, basta adicionar um Gateway para que um novo ponto de acesso ao servidor seja disponibilizado no site do cliente.

- Transparência: o Gateway funciona efetivamente como uma "bridge", sendo transparente tanto para a Palm quanto para o Servidor. Isto permite que as aplicações da Palm e do Servidor sejam desenvolvidas em paralelo com o desenvolvimento da rede, conforme a filosofia de integração de sistemas mencionada.

Tecnologia de Enlace – O Alicerce da Rede

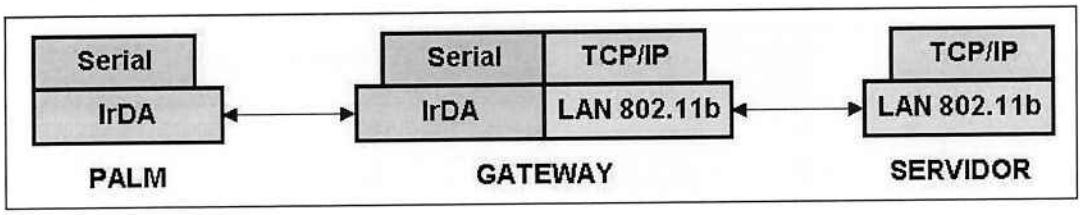


Figura 3: Stack de Protocolos

Tendo definido a arquitetura da rede, o próximo passo foi definir quais tecnologias de enlace iriam ser utilizadas como base da rede de comunicação. Por enlace entende-se a comunicação entre dois elementos vizinhos na rede. No caso, teremos um enlace entre cada Palm e seu respectivo Gateway, e um enlace entre cada Gateway e o Servidor.

Enlace Infravermelho (Palm-Gateway)

Para efetiva comunicação entre as Palms e o Gateway, como mencionado anteriormente, utilizamos a tecnologia de comunicação por infravermelho, baseada no padrão IrDA.

A IrDA é uma organização internacional que cria e promove padrões para equipamentos de transmissão infravermelho, geralmente para transmissão ponto-a-ponto (http://www.irda.org/)

Do ponto de vista da implementação, o IrDA é um conjunto de protocolos criados para suportar conexões a curtas distâncias com taxas de transmissão de 155Kbps até 4 Mbps. Alguns sistemas operacionais já possuem suporte ao IrDA, entre eles a

família MS Windows, que fornece APIs para programação e desenvolvimento de aplicativos.

Serial IrDa Physical Layer (SIR)

O SIR é um protocolo que define a transmissão serial via infravermelho (curta distância) com um start bit, oito data bits e um stop bit. A taxa máxima neste modo é de 115,2 Kbps em half duplex.

O hardware utilizado no projeto utiliza este protocolo para transmissão serial.

IrLAP (Camada de Enlace)

O protocolo de enlace utilizado na comunicação infravermelho é o IrLAP, que é baseado no protocolo HDLC (frame-relay) amplamente utilizado. O IrLAP oferece uma única e confiável conexão entre dois equipamentos. Se por qualquer razão a conexão for quebrada, uma mensagem de erro é enviada para as aplicações.

As velocidades de transmissão também são negociadas e modificadas pelo o IrLAP durante a conexão e transmissão dos dados.

IrLMP e TinyTP

Estes dois protocolos implementam a multiplexação de dados, permitindo que vários dispositivos se conectem simultaneamente utilizando o IrLAP, sem que uma transmissão interfira na outra.

Os protocolos também adicionam um controle de fluxo por cada conexão IrLAP, o que permite que as aplicações transmitam grande blocos de dados sem se preocupar com o controle de fluxo dos mesmos.

IrCOMM (Virtualização da Serial)

IrCOMM é uma família de protocolos que roda em cima do stack IrDA. Estes protocolos foram criados para permitir a transmissão de dados pelo IrDA como se o dispositivo fosse uma interface serial comum virtual.

Este tipo de virtualização foi utilizada na implementação do projeto, pois permite a transmissão de dados através de rotinas de transmissão serial padrão, tanto na Palm quanto no Gateway. Em outras palavras, todo o protocolo IrDA torna-se transparente, não sendo necessário o tratamento de dados no nível do infravermelho.

Foi encontrado um driver sob licença GNU (software gratuito) que implementa o IrCOMM no Windows 2000, que é a plataforma principal de desenvolvimento das aplicações de rede. Mais detalhes no tópico "Integração com a Palm".

De qualquer maneira, cabe aqui mencionar um pouco do funcionamento interno do IrDA. Os métodos e propriedades dos drivers IrDA assemelham-se muito aos de uma conexão TCP/IP, no sentido que é necessário um estabelecimento e negociação inicial da conexão.

Os dispositivos infravermelhos checam constantemente a presença de outros dispositivos na área de alcance do infravermelho. Ao verificar a presença de um dispositivo, ambos carregam em sua base de dados (IAS) informação contendo a identidade de cada dispositivo, e iniciam a transmissão de dados.

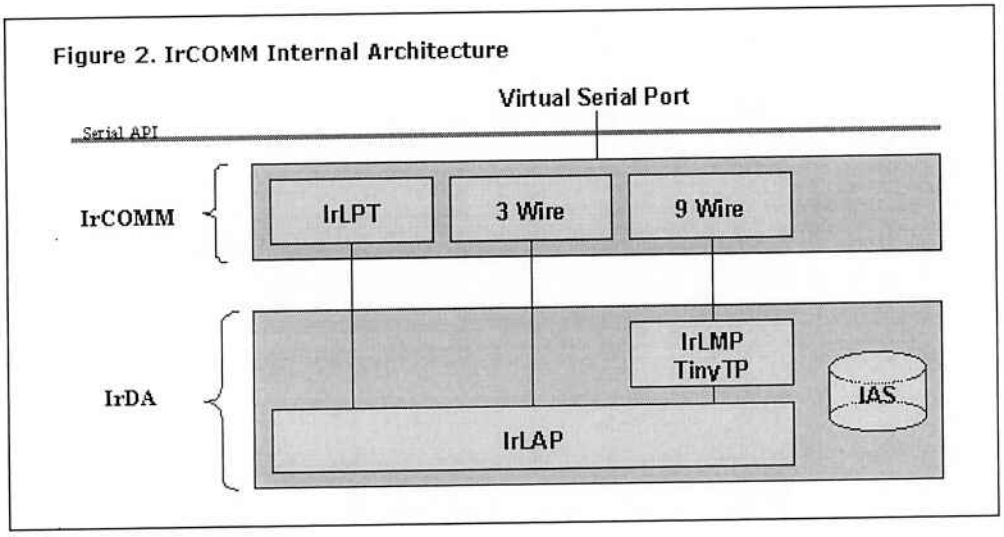


Figura 4: Arquitetura interna doIrCOMM

Do ponto de vista do IrCOMM, toda esta negociação é absolutamente transparente, não havendo necessidade de tratamento dos eventos. A transmissão de

dados é feita normalmente pelos métodos send() e receive(), que são traduzidos para o formato do protocolo TinyTP.

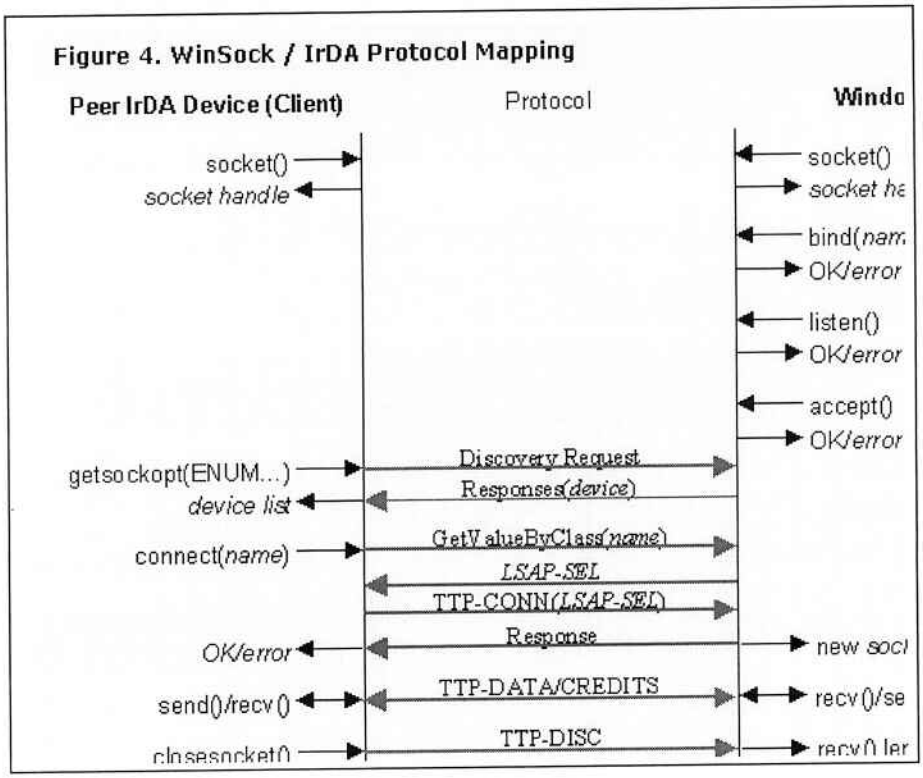


Figura 5: Mapeamento IrDA - WinSock

O equipamento

A interface infravermelho encontrada que satisfizesse toda a descrição feita anteriormente e utilizada no Gateway foi a XTNDAccess IrDA PC Adapter, conectada diretamente na interface serial do computador.

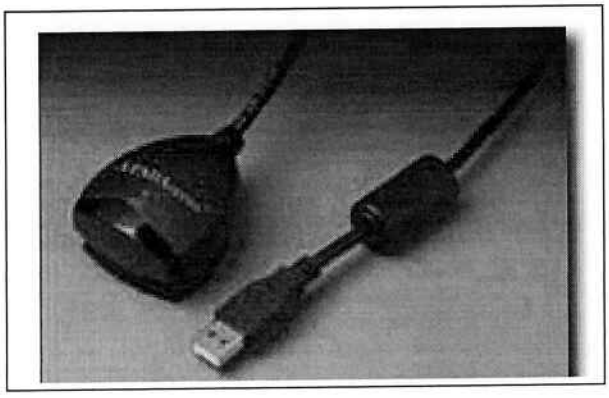


Figura 6: Extended Systems XTNDAccess IrDA PC Adapter

Especificações

- IrDA V1.0 Compliant
- Suporte para Windows 95/98/2000 e NT
- Taxa de transmissão de 115.200 baud com variação automática da banda conforme taxa de erros da transmissão
- Alcance de 1 metro
- Interface RS-232C de 9 pinos
- US$60,00

Enlace LAN Wireless – TCP/IP

A comunicação entre o Gateway e o Servidor é baseada no protocolo TCP/IP, que opera em nível de camada de transporte e rede. Para implementar o enlace em rádio, o grupo optou por utilizar uma variante da tecnologia Ethernet, a rede LAN Wireless, no padrão 802.11b.

Do ponto de vista do TCP/IP, a rede LAN Wireless é absolutamente transparente, ou seja, é como se o sistema rodasse em uma rede local tradicional. Claro que, o protocolo 802.11b em si difere do Ethernet tradicional, principalmente no que diz respeito ao meio físico de transmissão e ao método de detecção de colisão.

Velocidade (Mbps)	11	5.5	2	1
Alcance (Aberto)	160	270	400	550
Alcance(Semi Aberto)	50	70	90	115
Alcance(Fechado)	25	35	40	50

Figura 7: Velocidade X Alcance: Placa Orinoco Silver Avaya

Como método de transmissão, a rede LAN Wireless utiliza a tecnologia de Direct Sequence Spread Spectrum (CCK, DQPSK, DBPSK), a 2.4Ghz. O método de colisão utilizado é o CSMA/CD com ACK (Carrier Sense Media Access with Collision Detection), com mensagem de acknowledgement.

Na tabela acima temos um resumo da característica de Velocidade x Alcance da placa. Importante frisar que o próprio equipamento vai ajustando a velocidade de transmissão em função da força do sinal. Se o sinal estiver fraco, a velocidade é diminuída de forma a aumentar o alcance de transmissão, e vice-versa.

Tem-se um alcance máximo de 160m a 11Mbps, que é a velocidade máxima; e um alcance de 550 a 1Mbps, que é a taxa mínima de transmissão.

O equipamento

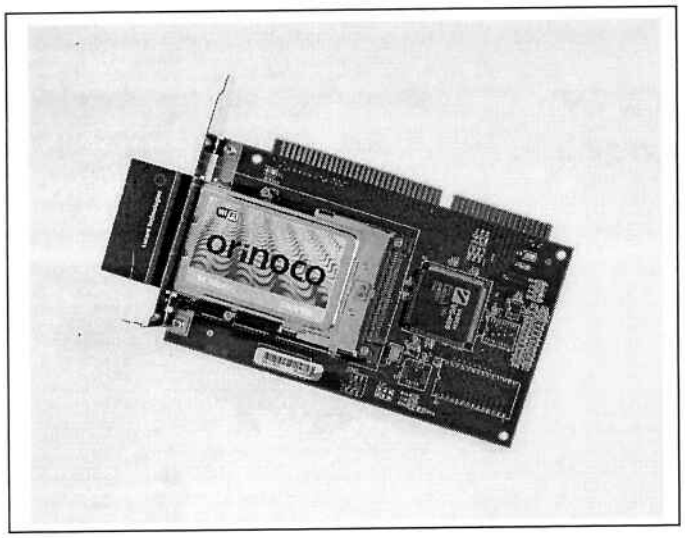


Figura 8: Placa LAN Wireless Modelo Orinoco Silver. Fonte: Lucent

O equipamento utilizado no projeto foi gentilmente emprestado pela Lucent, e é a placa Orinoco PCI versão Silver. A placa é totalmente compatível com o padrão LAN Wireless e possui todos os drivers necessários para que o TCP/IP tenha um acesso transparente à rede.

Este equipamento permite vários tipos de arquitetura rede. Entre elas, a mais tradicional, está ilustrada abaixo, e se baseia numa arquitetura de células. Em cada célula, que é basicamente definida pelo alcance máximo de transmissão, todos os equipamentos podem falar entre si, num esquema ponto-a-ponto. Cada célula também elege um concentrador, ou gateway de acesso, que pode ser equipamento específico ou uma máquina com interface comum programada como tal.

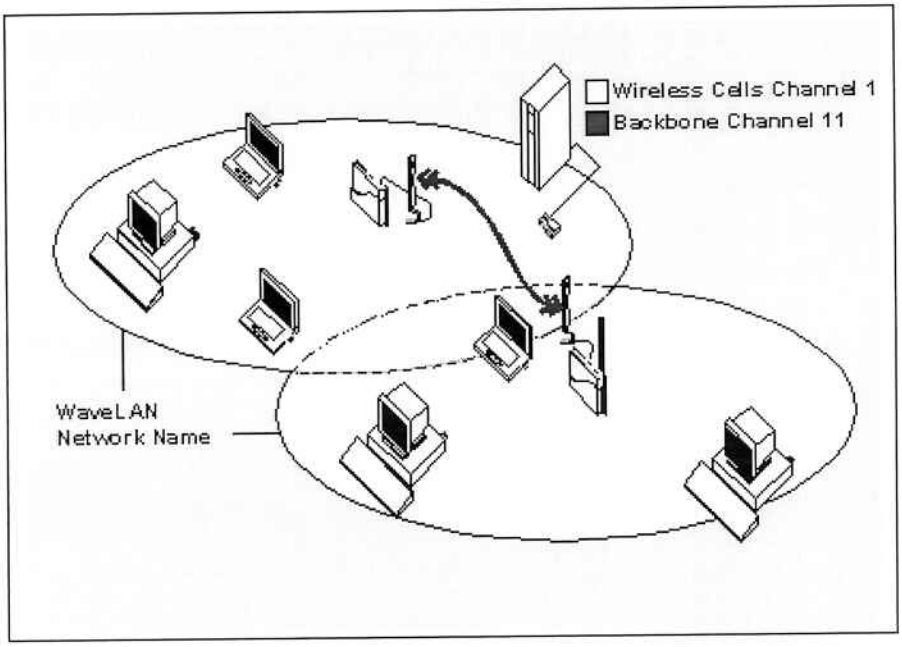


Figura 9: Arquitetura LAN Wireless. Fonte: Lucent

Caso um dos equipamentos tente acessar outro que está fora da célula, ele envia a mensagem para o gateway, que vai encaminha-la para os gateways das demais células. Desta forma, tem-se efetivamente um backbone lógico na rede, que passa por todas as células. Isto adiciona uma flexibilidade enorme à rede, permitindo redes sem fio de, em teoria, alcance ilimitado (claro que existem problemas de atraso em excesso em redes muito grandes, mas não é o o que ocorre nos casos práticos encontrados. Se for necessário, recomenda-se um arquitetura mista de wireless e redes tradicionais).

Importante notar a equivalência desta topologia com a topologia de rede proposta pelo grupo. Por esta, e por todas as características apresentadas anteriormente, a rede LAN Wireless foi a escolha natural para a comunicação dos gateways com o servidor.

Segurança

A tecnologia Spread Spectrum, por si só, já é bastante segura, uma vez que a sequência de bits é "espalhada" por diversas frequências diferentes, evitando que "sniffers" simples interceptem os dados. Porém, como existe a possibilidade de escuta

através de equipamento especializado, a Lucent adicionou nas suas placas ferramenta criptográfica para transmissão dos dados, bastando apenas habilitá-la na transmissão.

Esta criptografia está baseada numa chave de 64Bits (WEP - Wired Equivalent Privacy). A versão gold possui suporte para uma chave RC4 de 128Bits, porém não foi disponibilizada pela Lucent para este projeto.

A possibilidade de criptografar os dados a nível de enlace, e em hardware, permite uma grande simplificação nos protocolos de alto nível no que diz respeito a segurança (se desejado, porém, pode-se adicionar mais níveis de criptografia). Este foi mais um diferencial que levou à escolha da rede LAN Wireless no projeto.

ELABORAÇÃO DA ESTRUTURA DA BASE DE DADOS

Para armazenar os dados tratados pelo sistema, devem ser considerados dois formatos de bancos de dados distintos.

- MS Acess 2000: Devido ao curto prazo de tempo que o grupo teve para instalar e aprender o uso do MS SQL-Server, optou-se por uma solução mais simples, utilizando o MS-Acess. A principal diferença, é que não puderam ser implementados mecanismos de trigger, constraints e store procedures que poderiam facilitar a programação e a segurança dos dados. Estes mecanismos tiveram que ser implementados via software, numa camada superior à base de dados. No entanto, o MS Access também é um banco de dados relacional e de fácil utilização

- Banco de dados na Palm (formato PDB), que armazenará quantidades pequenas de dados, (no caso, o PalmOS usa o formato PDB de banco de dados). Na Palm, a estrutura de dados será bastante simplificada, armazenando apenas os dados relativos às mensagens do usuário.

Baseando-se nos requisitos definidos pela especificação do software, foi elaborado o seguinte diagrama Entidade-Relacionamento, de modo a obter a maior eficiência possível nas consultas de dados normalizados.

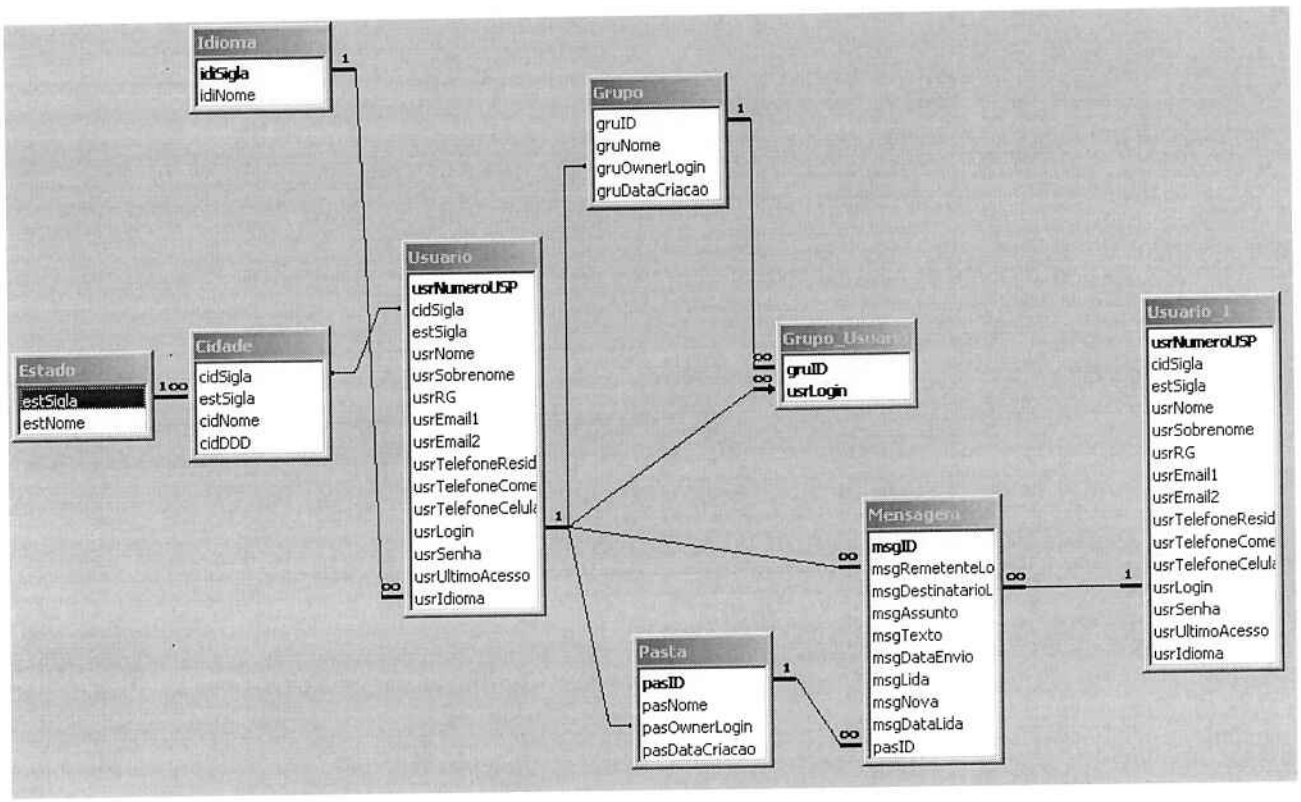


Figura 10: Diagrama ER - Estrutura do Banco de Dados

Entidades / Tabelas:

- **Estado**: {estSigla, estNome) – Esta tabela conterá todos os nomes e siglas de estados brasileiros. Esta tabela constará do sistema apenas por formalidade, visto que todos os alunos da Universidade, pelo menos enquanto cursam a graduação, residem no estado de São Paulo.

- **Cidade**: {cidSigla, estSigla (FK), cidNome, cidDDD) – Tabela que conterá as informações sobre as cidades brasileiras (principalmente as de São Paulo, próximas à capital), contendo a sigla da cidade, seu nome e DDD, além de uma referência ao estado em que ela se situa.

- **Idioma**: {idSigla, idNome) – Tabela que conterá todos os idiomas disponíveis no sistema, visto que este será multi-língüe.

- **Grupo**: {gruID, gruNome, gruOwnerLogin, gruDataCriacao} – Os usuários do sistema poderão ser divididos em grupos, de modos que o envio de mensagens e eventos poderão ser associados a grupos, e não apenas a usuários únicos. Cada grupo possui um proprietário, assim, cada usuário poderá ter os seus grupos.

- **Usuário**: {usrLogin, cidSigla (FK), estSigla (FK), usrNome, usrSobrenome, usrNumeroUSP, usrRG, usrEmail1, usrEmail2, usrTelefoneResidencial, usrTelefoneComercial, usrTelefoneCelular, usrLogin, usrSenha} – Conterá as informações pessoais de cada usuário.

- **Grupo_Usuário**: {gruID (FK), usrLogin (FK) – tabela que relaciona os usuários e a que grupos estes pertencem. Um usuário pode estar em mais que um grupo.

- **Mensagem**: {msgID, msgRemetenteID(FK), msgDestinatarioID(FK), msgAssunto, msgTexto, msgDataEnvio, msgLida, msgNova, pasID (FK)} – conterá informações relativas às mensagens enviadas pelo sistema para cada usuário.

- **Pasta**: {pasID, pasNome, pasOwnerLogin (FK), pasDataCriacao} – cada usuário poderá agrupar suas mensagens em pastas diferentes.

DESENVOLVIMENTO DE COMPONENTES

Grande parte das aplicacações desenvolvidas hoje dia utiliza-se da arquitetura de duas camadas, ou seja, o servidor de banco de dados, e a aplicação propriamente dita. Esta arquitetura possui uma série de desvantagens, como por exemplo: código relativo à lógica de negócio misturado com código relativo à interface, cada aplicação cliente roda num processo separado, não podendo compartilhar recursos, cada cliente precisa ter os drivers de acesso ao banco de dados e fazer sua própria conexão ao servidor, fica difícil construir aplicações que se baseiam em servidores em LANs distintas, a aplicação fica muito dependende da plataforma, e outras.

Dividindo-se a aplicação em camadas (multi-tier), foi possível satisfazer aos seguintes fatores:

- Separar lógica de acesso a dados da lógica de negócio e da de apresentação, trazendo facilidade na análise e geração de código, e modularização, facilitando a manutenção, por isolar cada partícula de software em partes distintas.
- Compartilhamento de recursos: compartilhamento de memória, distribuição do processamento entre diversas máquinas de uma rede, compartilhamento de conexões ao banco de dados, etc;
- Interoperabilidade entre diferentes plataformas e sistemas operacionais (desde - que estes, e os componentes desenvolvidos, respeitem a padrões pré-estabelecidos pelas organizações de padronização).
- Código mais inteligível, e portanto de maior manutenibilidade.

Através do MS Windows 2000 e de seu Gerenciador de Componentes COM+, é possível que bibliotecas desenvolvidas em diversas linguagens (Visual Basic, Visual C, etc) sejam registradas no sistema. Desta maneira, os componentes destas DLLs podem ser utilizados em qualquer outra aplicação, escrita em qualquer linguagem, inclusive linguagens de script, como Active Server Pages (ASP), rodando no servidor IIS.

No caso deste projeto, serão utilizados componentes escritos em Visual Basic, que serão acessados pelas páginas ASP rodando no servidor HTTP. Da mesma forma, estes componentes estarão sendo utilizados pelo servidor responsável pela retirada de dados da base e comunicação com o gateway.

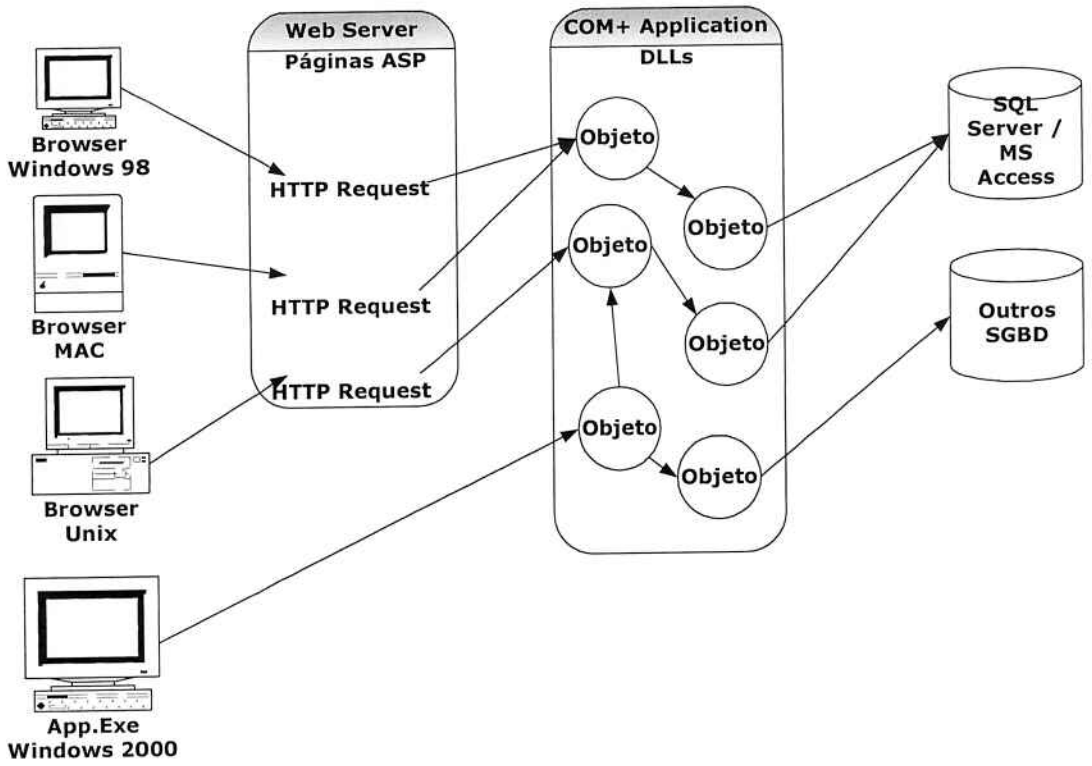


Figura 11: Arquitetura COM+

Para programação utilizando esta tecnologia, será usado o paradigma da orientação a objetos, ou, pelo menos o suporte que o Visual Basic oferece para esta técnica de Engenharia de Software. Por exemplo, o Visual Basic não suporta herança da maneira como este conceito é especificado pelos padrões da Orientação a Objetos.

Seguem todas as bibliotecas de classes criadas e suas interfaces:

- **Formatura_DB.dll**

Esta biblioteca faz a interface entre a biblioteca de lógica de negócios Formatura_BL.dll e o banco de dados propriamente dito. Ela não deve ser utilizada diretamente por nenhuma outra DLL, e muito menos por nenhuma outra parte da aplicação.

Nenhuma outra classe ou biblioteca acessa o banco de dados diretamente, e, portanto, todas as queries SLQ são compostas e executadas pelos métodos desta classe, que, na maioria dos casos, retornam vetores de Strings carregados com as informações do banco de dados.

- **Classe:** cDB – classe que oferece métodos para acessar a base de dados.
- **Propriedades:**
  -
- **Métodos:**
  - *abreParaVerificacao(strSQL As String) as Boolean*
    recebe como parametro uma query SQL e retorna True caso esta query tenha como retorno um recordset não nulo.
  - *Apagar(ByVal strWhere As String, ByVal strTabela As String) As Boolean*
    recebe como parametro uma string contendo o nome da tabela que terá registros apagados, e uma string contendo a clausúla where do comando SQL que fara a exclusão. Retorna True se a o comando for executado com sucesso (mesmo que a exclusão não tenha sido feita, por n
  - *Atualiza(strRegistros() As String, strValores() As String, ByVal strTabela As String, ByVal strRegistro As String, ByVal strValor As String, Optional blnString as Boolean) As Boolean*
    Recebe como parametro um vetor de registros a serem atualizados, um vetor com os valores a serem atualizados, uma string com o nome

da tabela no banco de dados na qual será feita a atualização, uma string contendo o registro, e outra o seu valor, para compor a clausula where do update que será feito na tabela. blnString determina que tratamento de strValor deve ser de string (entre aspas na query SQL). Retorna true caso o comando seja executado com sucesso.

- *Carregar(strRegistros() As String, ByVal strTabela As String, ByRef strRetorno() As String, strWhere As String) As Boolean*
  Coloca no vetor strRetorno os registros do vetor strRegistros carregados da tabela strTabela, de acordo com o critédio estabelecido pela clausula where. Retorna True se houver pelo menos um registro carregado.
- *Conecta(blnGravacao As Boolean) As ADODB.Connection*
  Realiza a conexão com o banco de dados através do driver ODBC, recebendo como parâmetro um booleano que indica se o banco de dados deve ser aberto para escrita (True), ou somente para leitura (False). Este método só deverá ser chamado pelos demais métodos desta mesma classe.
- *Contar(ByVal strTabela As String, strWhere As String, Optional strJoin As String) As Long*
  Retorna um número longo com a quantidade de registros da tabela strTabela que satisfazem as condições da clausula where em strWhere, e, o parâmetro opcional strJoin permite que seja feito um JOIN na query, enriquecendo sua sintaxe e sua performance.
- *Popular(strRegistros() As String, ByVal strTabela As String, ByRef strRetorno() As String, Optional strWhere As String, Optional strJoin As String, Optional strOrder As String) As Long*
  Carrega no vetor strRetorno os registros do vetor strRegistros da tabela strTabela, que satisfaçam os critérios da clausula where em strWhere (se for vazia, todos), incluindo um JOIN de tabelas, caso strJoin seja não nulo, ordenados de acordo com o critério estabelecido por strOrder, caso este seja não nulo.
- *Function Salvar(strRegistros() As String, strValores() As String, ByVal strTabela As String) As Boolean*

Grava no banco de dados nos registros de strRegistros, os valores de strValores, na tabela strTabela, retornando True caso a gravação seja realizada com sucesso.

- *Verifica(ByVal strTabela As String, ByVal strCampo As String, ByVal strValor As String, Optional ByVal strOther) As Boolean*
  Retorna True se conseguir abrir a tabela strTabela, onde o campo strCampo seja igual ao valor em strValor, e outros complementos para a clausula where, caso strOther seja não nulo.

- **Formatura_BL.dll**

Esta biblioteca implementa a lógica de negócios, contendo a definição das classes úteis à aplicação. Ela faz referência direta a Formatura_DB.dll, e deve ser referenciada tanto pela biblioteca de apresentação Formatura_VI.dll, quanto pelas interfaces da aplicação com o usuário (programa desenvolvido em VB e página ASP).

Esta biblioteca trata as entidades da aplicação em alto nivel, utilizando classes que representam a realidade, conforme os conceitos de Orientação a Objeto, que foram, na medida do possível, aplicados neste projeto.

- **Classe:** cCidade – classe que representa uma cidade no sistema
- **Propriedades:**
  - Nome as String, contém nome da cidade
  - Sigla as String, contém sigla da cidade
  - DDD as String, contém DDD da cidade
- **Métodos:**
  - *Carregar(Optional strSigla As String) As Boolean*
    Carrega o objeto cidade dada uma sigla em strSigla ou caso a propriedade sigla do objeto esteja preenchida. Retorna True se objeto for carregado com sucesso.

- **Classe:** cCidades – classe que representa um conjunto de cidades
- **Propriedades:**

- **Métodos:**
    - *Add(objCidade As cCidade, Optional sKey As String)*
      Adiciona um objeto do tipo cCidade na coleção de objetos, e, caso seja dado um índice, o objeto é adicionado na posição correspondente.
    - *Item(vntIndexKey As Variant) As cCidade*
      Retorna o objeto do tipo cCidade que está no índice solicitado da coleção.
    - *Remove(vntIndexKey As Variant)*
      Remove o objeto do tipo cCidade que está no índice solicitado da coleção
    - *Count() As Long*
      Retorna um inteiro longo com o número de objetos do tipo cCidade na coleção.
    - *Popular(Optional ByVal estSigla As String) As Boolean*
      Popula a coleção cCidades com objetos do tipo cCidade, de acordo com a sigla de um estado caso esta for fornecida (ou todas as cidades), e retorna True caso tenha sucesso na operação.

- **Classe:** cErro – classe que representa uma mensagem de erro para o usuário final

- **Propriedades:**
    - Descricao as String, contém descrição do erro

- **Métodos:**

- **Classe:** cErros – classe que representa um conjunto de mensagens de erro para o usuário final.

- **Propriedades:**
    - Erro as Boolean, indica se objetos cErro na coleção

- **Métodos:**

- *Add(Descricao As String, Optional sKey As String)*
  Adiciona um objeto do tipo cErro com descrição dada pelo parâmetro na coleção de objetos, e, caso seja dado um índice, o objeto é adicionado na posição correspondente.
- *Item(vntIndexKey As Variant) As cErro*
  Retorna o objeto do tipo cCidade que está no índice solicitado da coleção.
- *Remove(vntIndexKey As Variant)*
  Remove o objeto do tipo cErro que está no índice solicitado da coleção
- *Count() As Long*
  Retorna um inteiro longo com o número de objetos do tipo cErro na coleção.

- **Classe:** cEstado – classe que representa um Estado no sistema

- **Propriedades:**
  - Nome as String, contém nome do Estado
  - Sigla as String, contém sigla do Estado

- **Métodos:**
  - *Carregar(Optional strSigla As String) As Boolean*
    Carrega o objeto estado dada uma sigla em strSigla ou caso a propriedade sigla do objeto esteja preenchida. Retorna True se objeto for carregado com sucesso.

- **Classe:** cEstados – classe que representa um conjunto de Estados

- **Propriedades:**
  -

- **Métodos:**
  - *Add(objEstado As cEstado, Optional sKey As String)*
    Adiciona um objeto do tipo cEstado na coleção de objetos, e, caso seja dado um índice, o objeto é adicionado na posição correspondente.

- *Item(vntIndexKey As Variant) As cEstado*
  Retorna o objeto do tipo cEstado que está no índice solicitado da coleção.
- *Remove(vntIndexKey As Variant)*
  Remove o objeto do tipo cEstado que está no índice solicitado da coleção
- *Count() As Long*
  Retorna um inteiro longo com o número de objetos do tipo cEstado na coleção.
- *Popular() As Boolean*
  Popula a coleção cEstados com objetos do tipo cEstado, representando todos os estados cadastrados no sistema, e retorna True caso tenha sucesso na operação.

- Classe: cGrupo – classe que representa um grupo de usuários

- Propriedades:

- Nome as String, contém nome do grupo
- OwnerLogin as String, contém login do criador do grupo
- ID as Long, contém ID do grupo
- DataCriacao as Date, contém data de criação do grupo
- Membros as cUsuarios, contém objeto cUsuarios populador com membros do grupo

- Métodos:

- *AdicionarUsuario(strUsuarioLogin As String) As cErros*
  Adiciona um usuário a um grupo de usuários, dado seu login. O grupo.ID deve estar definido para esta operação ser realizada.
  Retorna uma coleção cErros com os cErro obtidos.
- *Carregar(Optional ByVal lngGrupoID As Long, Optional ByVal blnUsuarios As Boolean = True) As Boolean*
  Retorna True caso consiga carregar um objeto grupo, dado seu ID.
  Carrega dados dos usuários do grupo caso blnUsuarios seja True.
- *AdicionarUsuario(strUsuarioLogin As String) As cErros*
  Exclui um usuário de um grupo de usuários, dado seu login. O

grupo.ID deve estar definido para esta operação ser realizada.

Retorna uma coleção cErros com os cErro obtidos.

- *Excluir() As cErros*

  Remove todos os usuários de um grupo, e exclui grupo (grupo.ID deve estar preenchido). Retorna coleção de erros cErros.

- *Salvar() As cErros*

  Salva dados de grupo com propriedades preenchidas no banco de dados e retorna cErros com erros da operação

- *VerificaUsuarioGrupo(strUsuarioLogin As String) As Boolean*

  Retorna True se usuário de login strLogin pertencer ao grupo.

- Classe: cGrupos – classe que representa um conjunto de Grupos

- Propriedades:

- Métodos:

- *Add(objGrupo As cGrupo, Optional sKey As String)*

  Adiciona um objeto do tipo cGrupo na coleção de objetos, e, caso seja dado um índice, o objeto é adicionado na posição correspondente.

- *Item(vntIndexKey As Variant) As cGrupo*

  Retorna o objeto do tipo cGrupo que está no índice solicitado da coleção.

- *Remove(vntIndexKey As Variant)*

  Remove o objeto do tipo cGrupo que está no índice solicitado da coleção

- *Count() As Long*

  Retorna um inteiro longo com o número de objetos do tipo cGrupo na coleção.

- *Popular(Optional gruOwnerLogin As String, Optional strGrupoNome As String, Optional blnUsuarios as Boolean) As Boolean*

  Popula a coleção cGrupos com objetos do tipo cGrupo, dado um proprietário de grupo, ou um nome de grupo (para pesquisa), se não, todos os grupos, dependendo de blnUsuarios para carregar usuarios

de cada grupo ou não. cadastrados no sistema, e retorna True caso tenha sucesso na operação.

- **Classe:** cMensagem – classe que representa uma mensagem
- **Propriedades:**
  - RemetenteLogin as String, contém login do remetente da mensagem
  - DestinatarioLogin as String, contém login do destinatário da mensagem
  - ID as Long, contém ID da mensagem
  - Assunto As String, contém assunto da mensagem
  - Texto As String, contém texto da mensagem
  - DataEnvio As Date, contém data de envio da mensagem
  - DataLida As Date, contém data de leitura da mensagem
  - Lida As Boolean, indica se mensagem já foi lida pelo destinatário
  - Nova As Boolean, indica se mensagem é nova
- **Métodos:**
  - *Carregar(Optional ByVal lngMsgID As Long, Optional ByVal msgRemetente As String, Optional ByVal msgDestinatario As String, Optional ByVal blnEnviada As Boolean) As Boolean*
    Retorna True caso consiga carregar um objeto de mensagem, dado seu ID, seu remetente, seu destinatário o se foi enviada.
  - *Salvar() As cErros*
    Retorna objeto cErros informando se objeto mensagem preenchido pode ser salvo no banco de dados. Texto ou Assunto, Remetente e Destinatario são campos obrigatórios.
  - *marcarComoLida() as Boolean*
    Retorna True caso consiga atualizar no Banco de Dados que mensagem atual foi lida.
  - *MudarPasta(ByVal lngNovaPastaID as Long)*
    Retorna True se mudou mensagem para nova pasta
  - *Excluir as cErros*
    Excluir mensagens e retorna cErros contendo erros se houver.

- **Classe:** cMensagens – classe que representa um conjunto de Mensagens
- **Propriedades:**
  -
- **Métodos:**
  - *Add(objMensagem As cMensagem, Optional sKey As String)*
    Adiciona um objeto do tipo cMensagem na coleção de objetos, e, caso seja dado um índice, o objeto é adicionado na posição correspondente.
  - *Item(vntIndexKey As Variant) As cMensagem*
    Retorna o objeto do tipo cMensagem que está no índice solicitado da coleção.
  - *Remove(vntIndexKey As Variant)*
    Remove o objeto do tipo cMensagem que está no índice solicitado da coleção
  - *Count() As Long*
    Retorna um inteiro longo com o número de objetos do tipo cMensagem na coleção.
  - *Popular(Optional ByVal strRemetenteLogin As String, Optional ByVal strDestinatarioLogin As String, Optional ByVal blnLida As Boolean, Optional ByVal blnNaoLida As Boolean, Optional ByVal blnNova As Boolean, Optional ByVal blnNaoNova As Boolean, Optional lngPastaID as Long) As Boolean*
    Popula a coleção cMensagens com objetos do tipo cMensagem, dado um RemetenteLogin, e/ou DestinatarioLogin, e/ou se foi lida ou não, se é nova ou não, e/ou uma Pasta e retorna True caso tenha sucesso na operação.
  - *RetornaUmaNaoLida (ByVal strDestinatarioLogin As String) As cMensagem)*
    Retorna uma cMensagem entre as não lidas da coleção cMensagens. É necessária para o protocolo de mensagem escolher uma mensagem para transmitir para a Palm.

- **Classe:** cUsuario – classe que representa uma mensagem

- **Propriedades:**
    - Idioma as String, contém idioma de preferência do usuário
    - UltimoAcesso as Date, contém data de último acesso do usuário
    - Login as String, contém login do usuário
    - TelefoneCelular As String, contém celular do usuário
    - TelefoneComercial as String, contém telefone comercial do usuário
    - TelefoneResidencial as String, contém telefone residencial do usuário
    - Email1 as String, contém email do usuário
    - Email2 as String, contém outro email do usuário
    - RG as String, contém RG do usuário
    - Senha as String, contém senha do usuário
    - NumeroUSP as Long, contém Numero USP do usuário
    - Nome as String, contém nome do usuário
    - Sobrenome as String, contém sobrenome do usuário
- **Métodos:**
    - *VerificaSePodeLogar() As Boolean*

      Retorna True caso este usuário pode ser logado no sistema, preenchido número USP do usuário ou Login e Senha.
    - *Salvar() As cErros*

      Retorna objeto cErros informando se objeto usuário preenchido pode ser salvo no banco de dados. Nome, Sobrenome, NumeroUSP, Idioma, Cidade, Estado, Email1 ou Email2 obrigatórios.
    - *Carregar(Optional ByVal strLogin As String, Optional ByVal lngNumeroUSP As Long) As Boolean*

      Retorna True caso carregar dados do usuário, dado seu login, ou seu número USP, ou preenchido seu login.

- **Classe:** cUsuarios – classe que representa um conjunto de Mensagens
- **Propriedades:**
    -
- **Métodos:**
    - *Add(objUsuario As cUsuario, Optional sKey As String)*

      Adiciona um objeto do tipo cUsuario na coleção de objetos, e, caso

seja dado um índice, o objeto é adicionado na posição correspondente.

- *Item(vntIndexKey As Variant) As cUsuario*
  Retorna o objeto do tipo cMensagem que está no índice solicitado da coleção.
- *Remove(vntIndexKey As Variant)*
  Remove o objeto do tipo cUsuario que está no índice solicitado da coleção
- *Count() As Long*
  Retorna um inteiro longo com o número de objetos do tipo cUsuario na coleção.
- *Popular(Optional ByVal strNome As String = "", Optional ByVal strSobrenome As String = "", Optional ByVal strLogin As String = "", Optional ByVal lngGrupoID As Long = 0) As Boolean*
  Popula a coleção cUsuarios com objetos do tipo cUsuario, dado nome ou login ou sobrenome, ou grupo a que pertence, e retorna True caso tenha sucesso na operação.

- Classe: cPasta – classe que representa uma Pastas

- Propriedades:

- Nome As String
- OwnerLogin As String
- ID As Long
- DataCriacao As Date
- Membros As cMensagens

- Métodos:

- *Carregar(Optional ByVal strOwnerLogin As String, Optional ByVal lngPastaID As Long, Optional ByVal blnMensagens As Boolean = True) As Boolean*
  Carrega dados de uma pasta dado seu ID, carregando suas mensagens (ou nao) e retorna true se for bem sucedido.

- *ExcluirMensagem(lngMsgID As Long) As cErros*
  Exclui mensagem da pasta dado seu ID e retorna objeto do tipo cErros contendo erros se houver.
- *Excluir() As cErros*
  Exclui pasta atual e retorna objeto do tipo cErros contendo erros se houver.
- *Salvar() As cErros*
  Salva pasta desde que tenha owner e nome
- *VerificaSePastaExisteParaUsuario() As Boolean*
  Verifica se ja existe pasta com este me.Nome para me.OwnerLogin
- *VerificaMensagemPasta(lngMsgID As Long) As Boolean*
  Verifica se mensagem pertence a pasta

- Classe: cPastas– classe que representa um conjunto de Pastas

- Propriedades:

-

- Métodos:

- *Add(objPasta As cPasta, Optional sKey As String)*
  Adiciona um objeto do tipo cPasta na coleção de objetos, e, caso seja dado um índice, o objeto é adicionado na posição correspondente.
- *Item(vntIndexKey As Variant) As* cPasta
  Retorna o objeto do tipo cPasta que está no índice solicitado da coleção.
- *Remove(vntIndexKey As Variant)*
  Remove o objeto do tipo cPasta que está no índice solicitado da coleção
- *Count() As Long*
  Retorna um inteiro longo com o número de objetos do tipo cIdioma na coleção.
- *Popular(Optional pasOwnerLogin As String, Optional strPastaNome As String, Optional blnMensagens As Boolean) As Boolean*
  Popula a coleção cPastas com objetos do tipo cPasta dado o

proprietário, o nome da pasta, e retorna True caso tenha sucesso na operação (carregando mensagens ou nã0)

- **Classe:** cIdiomas – classe que representa um conjunto de Idiomas
- **Propriedades:**
  -
- **Métodos:**
  - *Add(objIdioma As cIdioma, Optional sKey As String)*
    Adiciona um objeto do tipo cIdioma na coleção de objetos, e, caso seja dado um índice, o objeto é adicionado na posição correspondente.
  - *Item(vntIndexKey As Variant) As cIdioma*
    Retorna o objeto do tipo cIdioma que está no índice solicitado da coleção.
  - *Remove(vntIndexKey As Variant)*
    Remove o objeto do tipo cIdioma que está no índice solicitado da coleção
  - *Count() As Long*
    Retorna um inteiro longo com o número de objetos do tipo cIdioma na coleção.
  - *Popular() As Boolean*
    Popula a coleção cIdiomas com todos os objetos do tipo cIdioma existentes, e retorna True caso tenha sucesso na operação

- **Classe:** cIdioma – classe que representa um Idioma
- **Propriedades:**
  - Nome As String
  - Sigla As String
- **Métodos:**
  - *Carregar(Optional strSigla As String) As Boolean*

  Retorna True se conseguir carregar os dados do idioma

- **Formatura_VI.dll**

Esta biblioteca faz a interface entre a Camada de Apresentação e a Lógica de negócios. Ela deve ser utilizada somente pelas páginas ASP da interface web do sistema, e sua responsabilidade é gerar código HTML que satisfaça à identidade visual do site.

- **Classe:** cPO – classe que oferece métodos para acessar os objetos de apresentação.

- Propriedades:

- Idioma as String, contém idioma que textos deverão ser apresentados

- Métodos:

- *montaBarra(strCorPortugues As String, intLargura As Integer, strTexto As String, Optional strCorTexto As String = "#ffffff", Optional strAlign As String = "left") As String*
  Retorna uma string contendo código HTML para gerar uma barra cor strCorPortugues, de largura intLargura, com o texto strTexto de cor strCorTexto (RGB), alinhado segundo strAlign.
- *montaCabecalho() As String*
  Retorna uma string contendo código HTML para gerar cabeçalho de cada página ASP.
- *montaHTMLHead(Optional ByVal strTitle As String, Optional ByVal blnCSS* As Boolean) As String
  Retorna uma string contendo código HTML para gerar código do Header de cada página HTML, colocando um titulo em <title></title> e incluindo o CSS caso necessário
- *montaNavBar(Optional ByVal strNome As String) As String.*
  Retorna uma string contendo código HTML para gerar código do menu lateral das páginas do site, de acordo com nome do usuário, caso este esteja logado, ou permitindo que login seja feito.

- *montaSelectCidadesPorEstado(ByVal strNome As String, Optional ByVal* estSigla As String, Optional ByVal cidSigla As String) As String

  Retorna uma string contendo código HTML para gerar combo box de nome strNome de cidades por estSigla (ou todas as cidades), com a cidade de sigla cidSigla já selecionada por default.
- *montaSelectEstados(ByVal strNome As String, Optional estSigla As String,* Optional strOther As String) As String

  Retorna uma string contendo código HTML para gerar combo box de Estados, de nome strNome, com o estado de estSigla selecionado por default, e um campo strOther que pode conter informações sobre chamadas a javascript ou estilos.
- *montaSubmit(ByVal strNome As String, ByVal strValue As String) As String*

  Retorna uma string contendo código HTML para gerar submit button com nome strNome e valor strValor.
- *montaTextField(ByVal strNome As String, ByVal strValue As String, ByVal* intSize As Integer, Optional ByVal intMaxSize As Integer, Optional ByVal blnHidden As Boolean, Optional strOther As String) As String

  Retorna uma string contendo código HTML para gerar text box de nome strNome, valor default strValue, tamanho intSize, tamanho máximo intMaxSize, hidden (se blnHidden = True), com estilos e javascript definidos em strOther.
- *mostraErro(objErros As cErros) As String*

  Retorna uma string contendo código HTML para gerar lista de descrições de cErro contida em cErros.
- *pegaTexto(strTexto As String) As String*

  Retorna uma string contendo o texto carregado da chave strTexto do arquivo texto do idioma carregado no objeto cPO.
- *montaIdiomas(ByVal strNome As String, ByVal lngIdiomaSelID As Long, Optional ByVal strScript As String) As String*

  Retorna uma string contendo código HTML para gerar combo box de

Idiomas, de nome strNome, com o idioma de ID lngIdiomaID selecionado por default, e um campo strScript que pode conter informações sobre chamadas a javascript.

- *pegaTexto(strTexto As String) As String*
  Retorna uma valor de uma dada chave do arquivo textos.txt correspondente ao idioma setado na propriedade Idioma (Ver detalhes no parágrafo sobre internacionalização).

Uso dos Componentes

O modelo de uso destes componentes pode ser acompanhado pelo seguinte diagrama de seqüência.

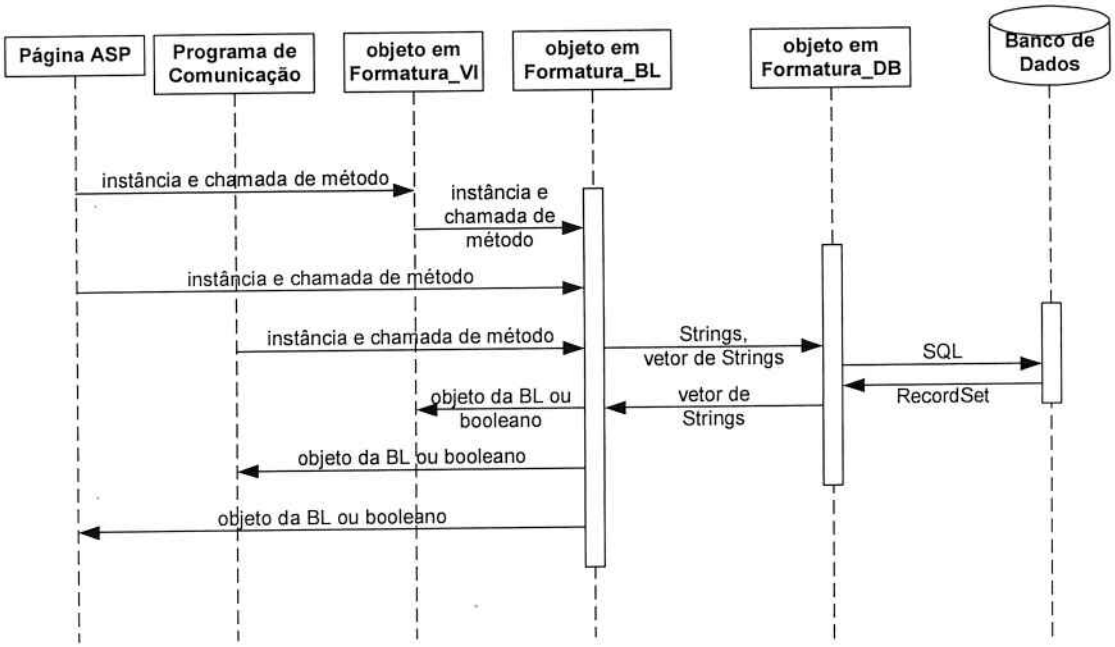


Figura 12: Diagrama de Seqüência - Uso dos Componentes

As páginas ASP, que fazem parte da camada de apresentação, podem utilizar-se de componentes da lógica de apresentação e da lógica de negócios.

O programa de comunicação, que, neste nível de abstração pode ser considerado como um agente da camada de apresentação 'paralelo' às páginas ASP, pode somente instanciar componentes da lógica de negócios, visto que a lógica de apresentação oferece recursos apenas para clientes que interpretam HTML.

A lógica de apresentação, tem acesso exclusivamente a camada de lógica de negócio, que, por sua vez, tem acesso exclusivo à biblioteca de acesso ao banco de dados.

A biblioteca de acesso ao banco de dados, por sua vez, tem exclusividade sobre o acesso ao banco de dados propriamente dito (na verdade, ela tem acesso ao componente Activex Data Object (ADO) da Microsoft, que acessa, via ODBC o banco de dados – isto foi excluído da representação do diagrama, visto que não fez parte do desenvolvimento do projeto, sendo parte integrante da plataforma Microsoft adotada).

Desta forma, a modularização, o controle de acessos, e o compartilhamento de recursos foram conseguidos sem grandes dificuldades, bastando para isso registrar os componentes no Container disponibilizado pela Microsoft no sistema operacional Windows 2000 (Gerenciador de Componentes COM+).

Este container é responsável por gerenciar o pooling de conexões ao banco de dados, e outros recursos, como segurança, que não foram necessários nesta aplicação.

Metodologia de Testes

À medida que foram desenvolvidos os componentes, eles puderam ser testados antes do desenvolvimento das páginas ASP ou de sua integração com os módulos de comunicação.

Para tanto, foi desenvolvido um pequeno projeto em VB que fez o papel de 'fat client', referenciando às bibliotecas a serem testadas (ora Formatura_DB.dll, ora Formatura_BL.dll e ora Formatura_VI.dll), utilizando-se controles básicos da interface gráfica do MS Windows, como botões, labels, text boxes e message boxes.

Desta maneira, à medida que novos componentes foram sendo criados (ou antigos foram sendo corrigidos), sempre foi possível testar seu funcionamento imediatamente, através deste projeto criado especialmente para testes.

Isso agilizou em muito o desenvolvimento, e permitiu que os componentes estivessem praticamente prontos e sem erros ao serem integrados com as páginas ASP e com os módulos de comunicação.

As próximas figuras exemplificam o uso deste projeto de testes e um código exemplo para testar o uso de objPasta de dado objUsuario. Desta maneira, testa-se o funcionamento dos métodos carregar de objUsuario, Popular e Count de objPastas.

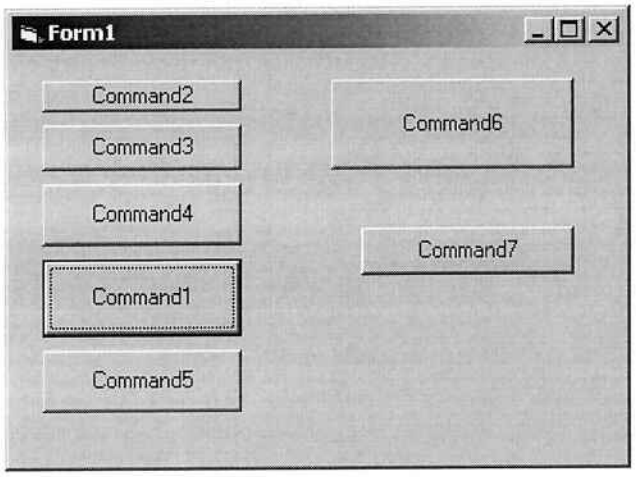


Figura 13: Exemplo da Interface de Testes dos Componentes

```
Private Sub Command6_Click()
    Dim objUsuario As New cUsuario
    Dim objPastas As New cPastas
    Dim objPasta As cPasta
    objUsuario.Carregar ("dovb")
    MsgBox objPastas.Popular(objUsuario.Login, , True)
    For Each objPasta In objPastas
        MsgBox objPasta.Nome & " - " & objPasta.Membros.Count
    Next

    Set objUsuario = Nothing
```

Figura 14: Exemplo de código associado a um botão

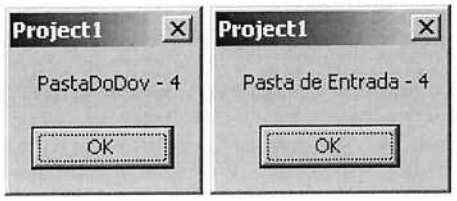


Figura 15: Resultado do Teste

Desta maneira, os testes podem ser feitos rapidamente, de acordo com as necessidades do programador.

A medida que novos componentes foram sendo criados, os testes foram levando em conta a integração entre eles. No caso do exemplo acima, primeiro testou-se a classe cUsuario, depois a cPasta, e depois a cPastas. Posteriormente, usou-se o exemplo propriamente dito, testando a integração entre os três. Deve-se levar em conta que testar uma classe significa utilizar exaustivamente seus métodos, passando todas as combinações possíveis de parâmetros (considerando inclusive variações nos opcionais), e verificar o resultado do método.

É muito útil o modo de debug oferecido pelo ambiente de desenvolvimento do Visual Basic, que vai executando o programa linha a linha, e mostrando os valores das variáveis, assim, cada instrução programada pode ser acompanhada e seus efeitos colaterais analisados, sem a necessidade de compilação, visto que o interpretador do Visual Basic funciona bastante bem para este fim.

DESENVOLVIMENTO DO SITE

ASP (Active Server Pages) é o nome dado a Microsoft à sua tecnologia de páginas dinâmicas. O desenvolvimento se deu utilizando a linguagem VB Script, por sua facilidade e sua integração com os componentes registrados no Gerenciador de Componentes do Windows 2000, e pela interface do ambiente de desenvolvimento Interdev 6.0.

Considerando que a maior parte da lógica necessária para a aplicação é realizada pelos componentes descritos no item anterior, às páginas ASP cabem duas tarefas principais:

- Gerenciar a geração de código HTML;
- Gerenciar a navegação pelo site;
- Tratar as requisições GET e POST dos browsers, fazendo chamada aos componentes (seus métodos e propriedades) requisitados;
- Chamar os componentes (seus métodos e propriedades) conforme suas necessidades;

O desenvolvimento do site se deu em etapas. Primeiramente decidiu-se quais as funcionaliades que ele deveria oferecer. Posteriormente, foram esboçadas todas as páginas necessárias para oferecer tais recursos. De posse destes dados, iniciou-se o desenvolvimento do site.

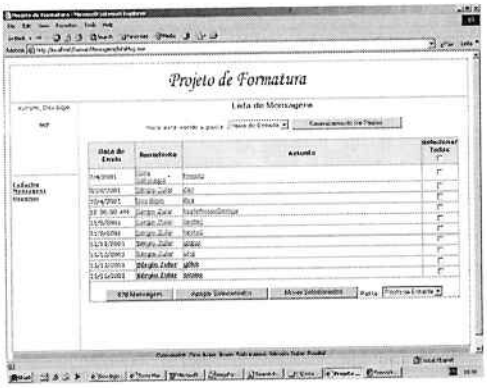
Pode-se considerar que esta fase do desenvolvimento do projeto foi relativamente curta e simples, uma vez que ela é apenas usuária dos componentes e não implementa muita lógica.

Não foi dada importância ao design do site, que é pobre em cores e em diagramação. Num projeto real, esta parte deveria ser implementada por especialistas em artes gráficas, que agregariam valor ao projeto dando-lhe uma aparência amigável e agradável aos olhos do usuário.

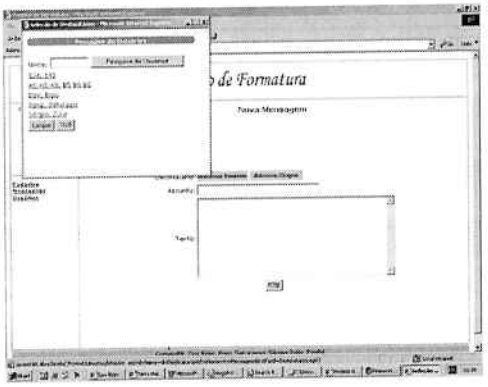
Como recurso adicional, e somente para demonstrar o potencial uso do sistema, foi implementado um mecanismo de internacionalização do site. Utilizando-se o método FormaturaBL.cPO.pegaTexto(chave), pode-se pegar o valor associado à chave solicitada de um dos arquivos textos.txt existentes na estrutura de diretórios, sob o diretório /rb/IDIOMA. Desta forma, cade usuário pode ver o site no idioma que lhe for mais conveniente, podendo, inclusive, graças ao suporte dos browsers modernos ao padrão UNICODE, selecionar idiomas cujos carácteres não são latinos.

Seguem algumas telas comentadas do site:

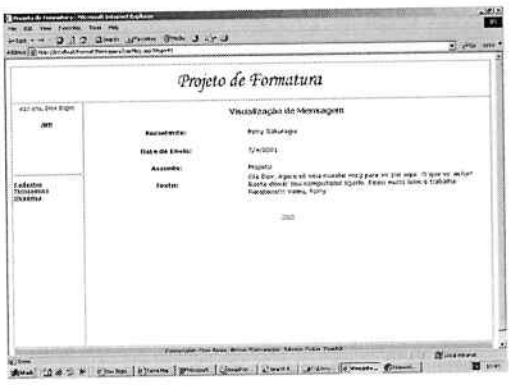
Tela 1: Apresentação do site. O usuário tem acesso ao menu principal, e, sempre que estiver logado, o sistema o identificará por seu nome. O idioma default do site é o portugüês, mas, uma vez que o usuário está logado, os textos aparecerão em seu idioma preferido.



Tela 2: Cada usuário pode manter diversas caixas postais, porém todos possuem necessariamente uma caixa de entrada. Nesta caixa são listadas em negrito suas novas mensagens. O usuário pode realizar todas as tarefas comuns de um programa de mensagens, como criar pastas, mover mensagens entre pastas e eliminar pastas e mensagens



Tela 3: Ao enviar uma nova mensagem, o usuário poderá escolher os destinatários, sejam estes grupos ou pessoas, através de uma listagem (com pesquisa). Assim, enquanto estiver no site, o usuário não precisa conhecer o login de seus destinatários. Ao clicar em Enviar, a mensagem é imediatamente gravada nas caixas de entrada dos destinatários.



Tela 4: Visualização de mensagens (marca mensagens como lidas)

FORMATAÇÃO DOS PACOTES / PROTOCOLOS DE COMUNICAÇÃO

Requisitos

Uma vez definidos os protocolos e equipamentos de rede, o grupo deu início ao levantamento de requisitos que a rede deveria possuir para transportar os dados das aplicações. Alguns aspectos gerais foram definidos:

- A rede só transmitiria dados no formato de cadeia de caracteres (strings)
- As aplicações usariam informações em formato de campos, com cada campo contendo uma string com a informação necessária. Por exemplo:
  . *Campo Usuário*: string com o nome
  . *Campo Mensagem*: string com a mensagem
  . E assim por diante...

- Para tráfego na rede, estes campos deveriam ser agrupados, ou concatenados, de forma a se transmitir uma string única, contendo todos os campos necessários para as aplicações. Daí a necessidade de utilização de um delimitador que indicasse o início e o fim de cada campo na string.
- O delimitador é uma string de tamanho arbitrário. Deve-se ter a flexibilidade de, durante o projeto, modificar este delimitador conforme o grupo achar necessário, de forma simples e rápida.
- Da mesma maneira, deve-se ser capaz de adicionar ou subtrair o número de campos transmitidos, de forma muito simples, sem que a modificação resulte em grandes mudanças na estrutura da rede. Também deve ser fácil a troca de ordem dos campos na string transmitida.

A string deve ser transmitida por interface serial entre a Palm e o Gateway, e por conexão TCP/IP entre o Gateway e o Servidor (em cada caso usando as interfaces adequadas).

Módulo de tradução

Com estes requisitos acima, foi dado início ao desenvolvimento de um programa relativamente simples, que deveria ser capaz de receber uma string e separar os campos de forma adequada e, no caminho reverso, receber os campos e montar a string corretamente. Este programa foi desenvolvido para, futuramente, servir de módulo básico de montagem/desmontagem de campos para as aplicações de rede mais complexas. Mostra-se abaixo o funcionamento lógico do programa.

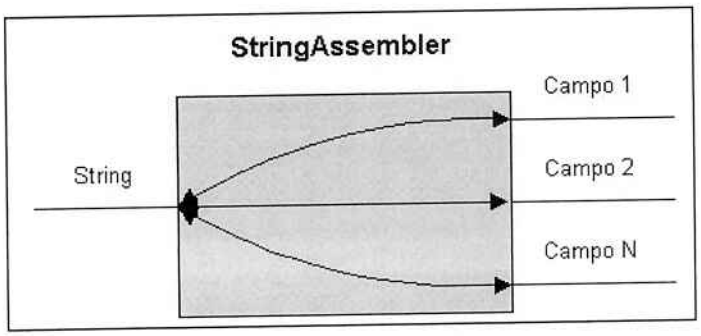


Figura 16: Funcionamento Lógico do Módulo

Abaixo está um exemplo do teste para uma string de 4 campos, sendo o separador a string "*;*". Observe que nas janelas de baixo o programa mostra os campos em separado, e na janela de cima a string a ser transmitida na rede (o programa funciona nos dois sentidos, conforme o botão selecionado).

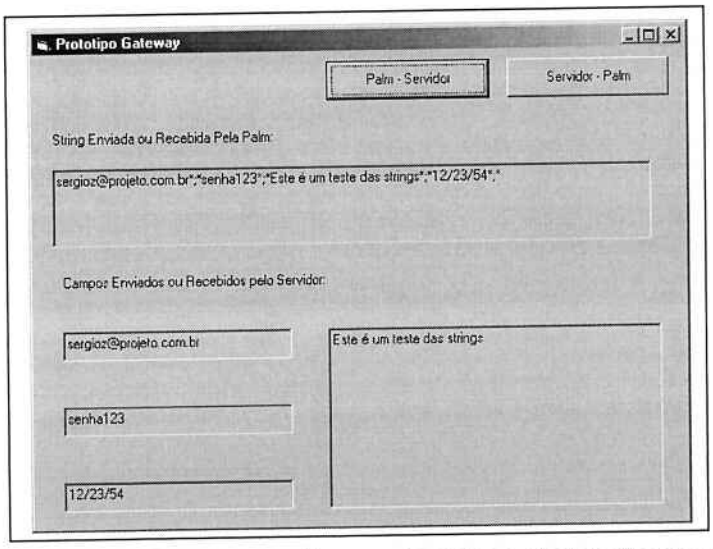


Figura 17: Programa de montagem e desmontagem de strings

Formato genérico da string

Com o módulo desenvolvido e testado, definiu-se o seguinte formato para o pacote:

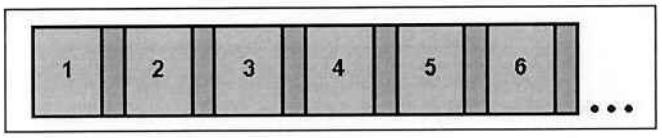


Figura 18: Formato genérico da string

Observe que cada campo é separado pelo mesmo delimitador. A string é finalizada também pelo delimitador, logo após o último campo.

Métodos do módulo

Para realizar a transformação de formatos, foi criado um módulo chamado **StringAssembler** que possui dois métodos:

- *MontaMensagemParaServidor*: pega uma string no formato correto e separa os campos.
- *MontaMensagemParaPalm:* pega os vários campos e monta a string colocando os separadores nos locais corretos.

Também são utilizados os seguintes parâmetros (constantes):

- *Delimitador*: especifica a string que delimita os campos
- *NúmeroCampos*: especifica quantos campos são transmitidos na string

Para modificar o número de campos, ordem dos campos e o delimitador basta modificar internamente o módulo StringAssembler, sem necessidade de nenhuma modificação adicional em qualquer outra parte do sistema.

Protocolo de Alto Nível – Interface com as Aplicações

Uma vez definido o funcionamento da rede em nível de transporte de dados, restou somente padronizar o meio como a rede vai interagir com as aplicações de alto

nível. Um outro modo de enxergar o problema é o seguinte: a rede presta serviço para as aplicações transportando os dados de uma ponta a outra, porém é necessário definir o modo que as aplicações acionam este serviço da rede e, adicionalmente, definir qual tipo de controle as aplicações podem ter no funcionamento da rede.

Do ponto de vista de implementação, o que está em discussão é o desenvolvimento dos módulos das aplicações que são responsáveis pela interface com a rede.

Utilizando a rede para controle das aplicações

Neste ponto do desenvolvimento foi necessária a definição dos controles necessários que as aplicações deveriam ter sobre a rede, e como as aplicações conversariam entre si. Inicialmente, ficou evidente a necessidade de alguns controles específicos:

Controles de Rede:

. *Controle de sinalização*: a aplicação deve sinalizar a rede quando pedir conexão e desconexão. A rede, conforme pedido, deve estabelecer ou fechar a conexão

. *Controle de conexão*: a rede deve informar às aplicações o estado da rede após o estabelecimento da comunicação. Se a rede cair, ela deve ser capaz de sinalizar a perda de conexão para as aplicações.

. *Controle de formato*: a rede tem autonomia de verificar se o formato dos dados estão corretos, e limitar o tamanho dos mesmos a um máximo pré-estabelecido, mesmo que isso acarrete em um erro em alto nível.

Controles de Aplicação:

. *Controle de fluxo*: é responsabilidade da aplicação controlar o envio das mensagens certas nas horas certas, sendo que ela pode utilizar suporte da rede para efetuar esta sincronização.

. *Controle de Autenticação*: caso seja necessário, a aplicação deve controlar a autenticação do usuário, podendo utilizar suporte da rede para sinalizar o sucesso ou falha de autenticação.

. *Controle de Erro*: a aplicação é responsável pela detecção de erros na comunicação. Sempre que houver erro ela deve informar qual tipo de erro encontrado, e fechar a conexão em seguida.

Para implementar tais controles, o optou-se por adicionar campos de controle junto aos dados enviados. Ao receber ou enviar dados, as aplicações devem manipular e preencher estes campos de forma a realizar os controles necessários. O funcionamento destes controles ficará bastante evidente conforme o projeto da rede for explicado.

Importante notar que, em princípio, qualquer tipo de informação pode ser utilizado nestes campos de controle, ou seja, qualquer controle de alto nível pode ser feito através deles. A desvantagem é a adição de um "overhead" na transmissão dos dados, já que os campos não transportam nenhum dado considerado útil do ponto de vista do usuário.

Definição de Transação

No projeto, a rede é utilizada principalmente para a transmissão de mensagens entre usuários cadastrados no sistema. As mensagens são transmitidas da Palm para o servidor e vice-versa.

Para especificar o protocolo de transmissão, não é necessário conhecer detalhes de como as mensagens são tratadas na Palm e no servidor, mesmo porque, sendo as partes desenvolvidas separadamente, e estando as interfaces entre elas definidas, isto não interessa.

É preciso, entretanto, ter uma visão macro dos processos envolvidos:

- Mensagens na Palm: a Palm possui um banco de dados de mensagens a serem enviadas ao servidor, e outro para receber as mensagens do mesmo.
- Mensagens no Servidor: o servidor possui um banco de dados de mensagens a serem enviadas à Palm, e outro para receber as mensagens da mesma.

- Transação: o grupo define como uma transação a troca de mensagens entre a Palm e o servidor. Ao final de uma transação, não deve haver nenhuma mensagem "a ser enviada" por nenhuma das partes.
- Sequência de transmissão: Numa transação, a Palm deve enviar todas as mensagens do seu banco de dados, enviando ao servidor mensagem específica quando não houver mais mensagens a enviar. Ao receber esta mensagem, o servidor deve enviar todas as mensagens do seu banco de dados à Palm, sinalizando a esta quando não houver mais mensagens a enviar. A transação deve então ser finalizada.
- Erro de transação: em qualquer fase da transação, se houver erro de qualquer natureza, a mesma deve ser interrompida. Caso ainda haja mensagens a serem enviadas, as mesmas devem ser transmitidas numa nova transação.
- Confiabilidade: Para toda mensagem recebida, o receptor deve enviar uma resposta de confirmação ao transmissor. Desta forma, em caso de erro, o transmissor saberá quais mensagens foram enviadas com sucesso, no caso de uma nova transação.

Especificando o Protocolo - Implementando uma Máquina de Estados

Como visto no item anterior, além de permitir o transporte dos dados em si, é necessário criar ferramentas para que tanto a Palm quanto o servidor sejam capazes de realizar uma transação. Enfim, definir em detalhes a interface da rede e das aplicações.

O melhor meio de abordar o problema, em termos de implementação, é criar uma Máquina de Estados de Transmissão, que trata todos os processos macro mencionados acima, porém já com ênfase na implementação. Vide abaixo:

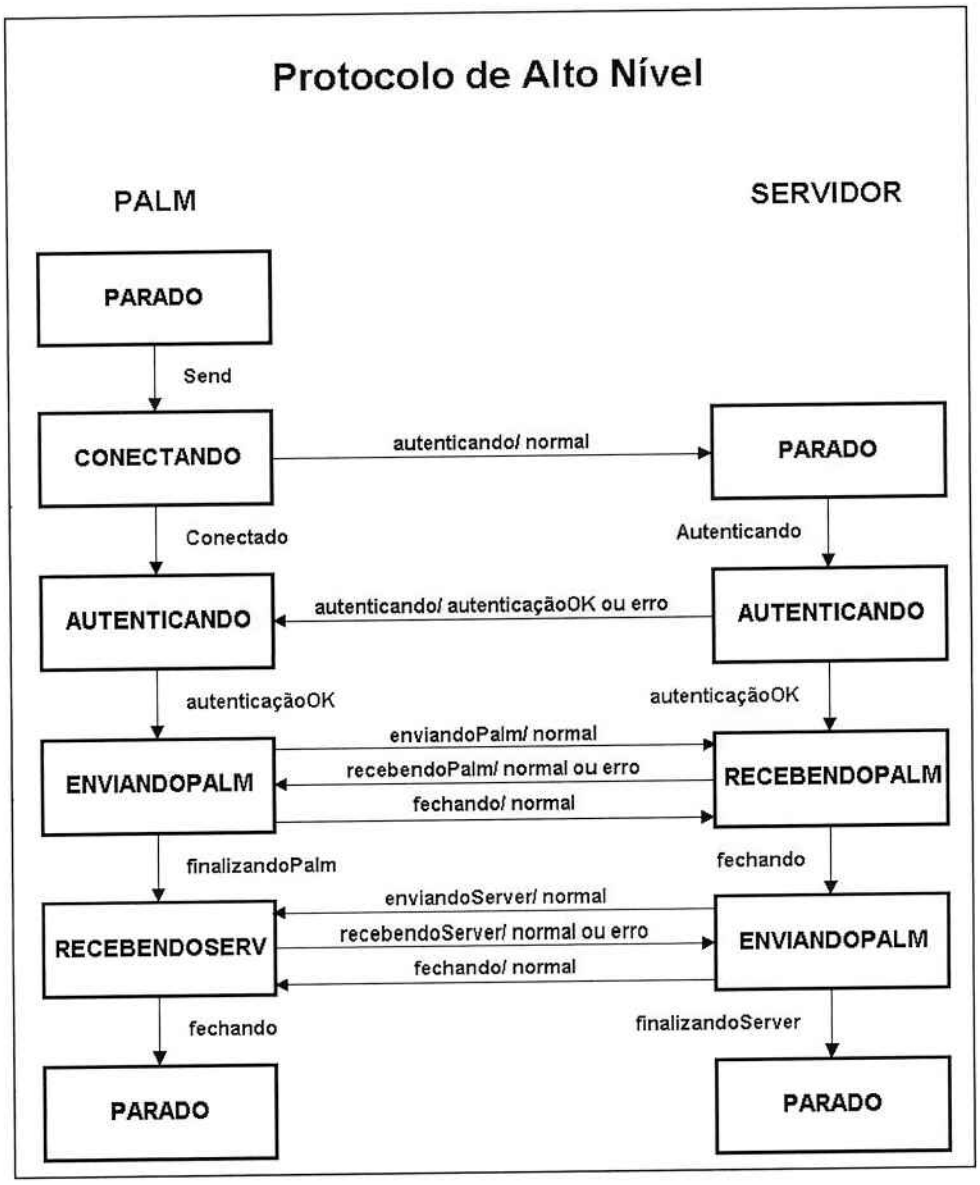


Figura 19: Máquina de Estados

A interpretação desta máquina é bem simples. Nos quadrados temos os estados dos equipamentos durante o processo de transação; no lado esquerdo os estados do sistema que se comunica com a Palm, no lado direito os estados do servidor.

A transição de estados só ocorre na presença de eventos, que são as flechas indicadas em baixo de cada quadrado. Um evento é, em geral, disparado pelo recebimento de um comando da aplicação, ou pelo recebimento de um controle específico pela rede.

Durante a fase de comunicação entre a Palm e o servidor, existe troca de mensagens de controle. Estas mensagens de controle, como mencionado anteriormente, são colocadas em campos específicos na string transmitida pela rede. No caso deste diagrama, são necessários dois campos de controle na comunicação entre a Palm e o servidor, que serão explicados em detalhe mais adiante.

Os Estados e Eventos da Comunicação do lado da Palm (lado direito do diagrama)

estPARADO: neste estado a Palm não está conectada à rede, portanto não realizando nenhum tipo de comunicação. O início da transação é causada por um evento interno SEND que aciona o pedido de conexão ao gateway, e envio da informação de autenticação ao Servidor.

estCONECTANDO: a Palm fica neste estado até o gateway confirmar a conexão com o servidor. Faz-se aqui uma ressalva ao diagrama acima, que para simplificar o desenho, não deixa claro esta confirmação de conexão por parte do gateway. Caso o gateway não confirme a conexão num tempo pré-determinado, a conexão pe fechada e vai para estPARADO.

estAUTENTICANDO: a Palm aguarda a confirmação de autenticação do servidor. Se a autenticação foi aceita, ela passa ao estENVIANDOPALM.

estENVIANDOPALM: neste estado, a Palm envia as mensagens do seu banco de dados para o servidor. A cada mensagem enviada, ela aguarda aviso do servidor da recepção da mesma. Ao receber a confirmação, se houver mais mensagens, manda a próxima mensagem. Se não houver mais mensagens, sinaliza ao servidor que finalizou a transmissão, e passa ao estRECEBENDOSERV

estRECEBENDOSERV: neste estado, a Palm recebe as mensagens do Servidor uma a uma, mandando aviso de confirmação a cada mensagem recebida. Ao receber sinalização do servidor que não há mais mensagens, vai para estPARADO.

Os Estados e Eventos do Servidor (lado esquerdo)

estPARADO: neste estado o servidor simplesmente "escuta" os gateways, porém não realiza nenhum tipo de comunicação. O início da transação é causada pelo pedido de conexão por parte do gateway, que vem junto com a informação de autenticação da Palm. Ao estabelecer conexão com o gateway, vai para estAUTENTICANDO.

estAUTENTICANDO: o servidor processa a informação de autenticação da Palm, validando ou não a mesma. Se não for válida, avisa a Palm do erro e vai para estFECHADO. Se for válida, avisa a Palm e vai para estRECEBENDOPALM.

estRECEBENDOPALM: neste estado, o servidor recebe as mensagens da Palm uma a uma, mandando aviso de confirmação a cada mensagem recebida. Ao receber sinalização da Palm que não há mais mensagens, vai para estENVIANDOPALM.

estENVIANDOPALM: neste estado, o servidor envia as mensagens do seu banco de dados. A cada mensagem enviada, ela aguarda a aviso da Palm da recepção da mesma. Ao receber a confirmação, se houver mais mensagens, manda a próxima mensagem. Se não houver mais mensagens, sinaliza à Palm que finalizou a transmissão, e passa ao estPARADO.

Transição de estados em caso de ERRO

Se houver qualquer erro em qualquer um dos estados acima, seja na Palm ou no servidor, ocorre uma transição para estPARADO. Estes erros podem ser causados tanto por perda de conexão, erros nos dados recebidos, ou erros de autenticação.

Sempre que houver transição para estPARADO, os equipamentos, além de enviar a mensagem de erro, irão automaticamente fechar a conexão, forçando o início de uma nova transação.

No caso de perda de conexão

Um fato que também não está evidenciado no diagrama, porém de grande importância, é o gerenciamento da conexão de rede em si, que é executado por todos os elementos envolvidos: gateway, servidor e Palm.

Periodicamente, a conexão é checada, seja pela verificação do Socket TCP/IP (no caso da ligação gateway-servidor), ou pelo envio de KeepAlives (no caso da ligação Palm-gateway). Se um dos elementos perceber que a conexão foi perdida, automaticamente vai para estPARADO, encerrando a transação.

Desta forma, a perda de conexão é detectada em qualquer parte da rede, e em qualquer sentido possível de comunicação.

As mensagens de controle

Para completar a especificação, falta apenas determinar o modo como as mensagens de controle, trocadas entre a Palm e o servidor, são enviadas.

As mensagens de controle são enviadas e recebidas em dois campos de controle específicos, enviados junto com os dados na string que passa pela rede. As mensagens de controle são obtidas através da recepção e interpretação destes dois campos, que seguem descritos abaixo:

- Controle de Comunicação: o controle de comunicação é um código que informa a aplicação qual o estado em que se encontra o outro equipamento:
    - Autenticando: informa a fase de autenticação. É mandando junto com a informação de autenticação (Palm), ou junto com a autorização (servidor);
    - enviandoPalm: Palm está enviando uma mensagem ;
    - recebendoPalm: servidor informa Palm que recebeu aquela mensagem;
    - fechando: Palm ou servidor sinalizam após enviar todas as mensagens;
    - enviandoServer: servidor está enviando uma mensagem;
    - recebendoServer: Palm informa servidor que recebeu aquela mensagem.

- Controle de Erro: indica se houver algum erro no processo de comunicação. O controle de erro é prioritário ao controle de comunicação, no sentido que ele é verificado primeiro e, em caso de erro, aborta toda a transação.

    - Normal: comunicação normal;

    - Erro: indica que houve algum erro no processo de comunicação. Na fase de autenticação, indica erro de usuário ou erro de senha.

    - AutenticaçãoOK: indica que a autenticação no servidor foi bem sucedida

Observe que, no diagrama, estes campos são indicados da seguinte maneira: Controle de Comunicação / Controle de Erro

Colocando tudo junto – Formato e Construção do "Campo"

Com o formato dos campos definidos, a forma de concatenação dos mesmos determinado, e o protocolo de alto nível especificado, temos praticamente todo o necessário para iniciar a implementação da rede. Falta, é claro, definir quais campos são necessários para as aplicações do projeto. São eles:

- Campos de Dados:
    1. Nome Usuário (String – máx 50 caracteres)
    2. Senha (String – máx 50 caracteres)
    3. Destinatário/Remetente (String – máx 100 caracteres)
    4. Data (String – máx 10 caracteres)
    5. Assunto (String – máx 100 caracteres)
    6. Mensagem (String – máx 10.000 caracteres)

- Campos de Controle:
    7. Controle de Comunicação (String – 3 caracteres)
    8. Controle de Erro (String – 3 caracteres)

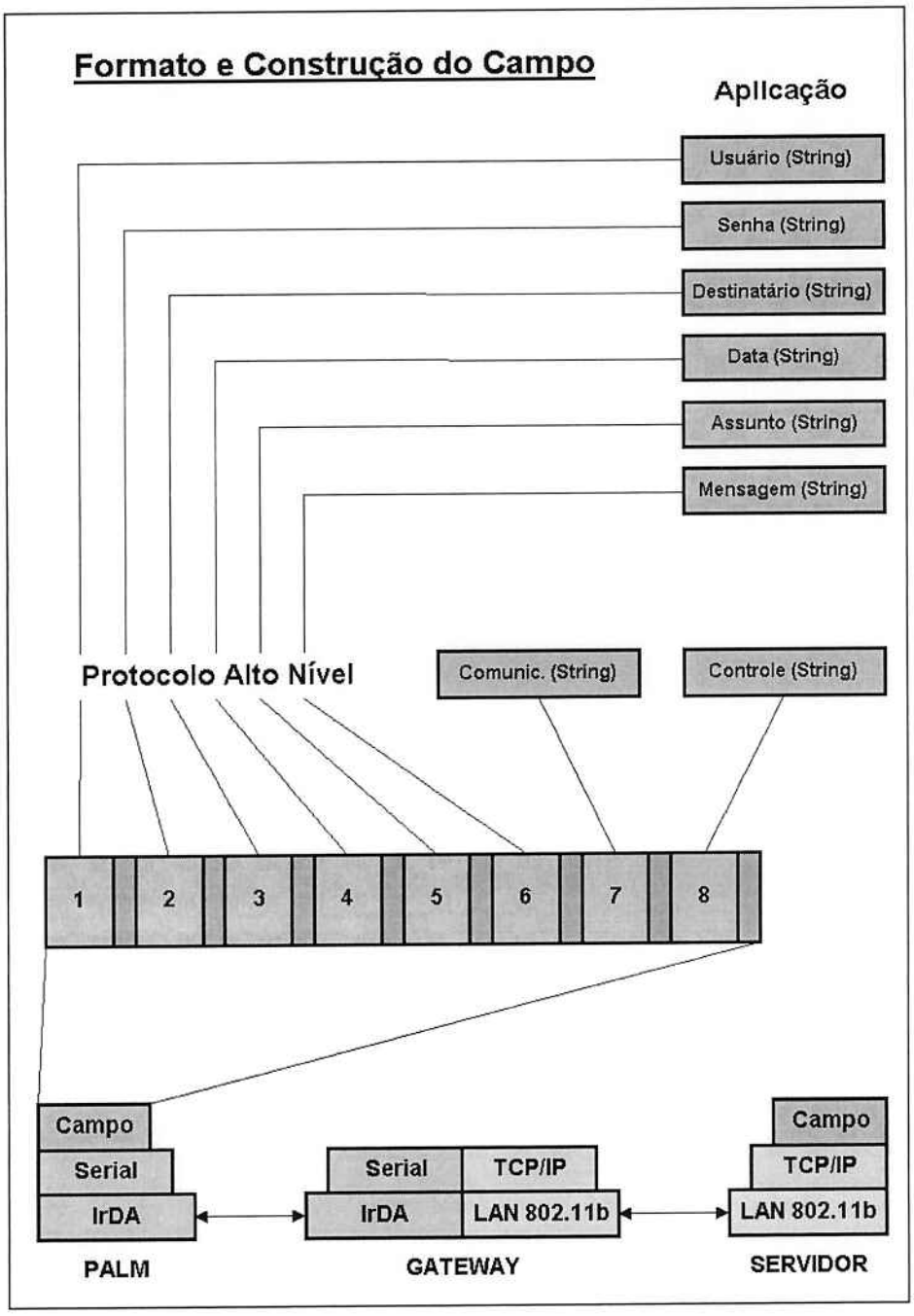


Figura 20: Formato e Construção do Campo

Na figura acima está ilustrado a organização interna dos campos que vão trafegar na rede. Estes campos são separados ou concatenados através de funções do módulo StringAssembler, conforme segmentação do delimitador.

A informação de cada campo, claro, depende do estado em que o equipamento se encontra, e dos dados e informações de controle recebidas, seguindo o diagrama de estados especificado.

A partir de agora, por motivo de clareza, chamaremos de "campo" a string composta pela concatenação dos campos de dados e de controle, e que vai ser transmitida pela rede (vide figura).

Início da Implementação – Metodologia de construção

Com a rede completamente especificada, seguiu-se a fase de implementação da rede. Nesta fase, os componentes da lógica de negócio do lado do servidor já encontrava-se desenvolvido, incluindo os métodos de acesso ao banco de dados, necessários para o envio e recepção de mensagens.

De qualquer maneira, como a ponta da Palm não estava desenvolvida, e como o protocolo não havia sido sequer sido testado, seria precipitado implementar a rede já utilizando o servidor em uma das pontas.. Obviamente, existia ainda todo um trabalho de refinamento e teste do protocolo antes dele entrar em fase de produção. Surgiu então a questão: como testar e desenvolver a rede independentemente do uso da Palm e do servidor?

A metodologia escolhida, como mencionado na introdução deste capítulo, consistiu em simplesmente criar módulos que simulassem os equipamentos sem, de fato, utilizar os sistemas. Foram então desenvolvidos emuladores, desenvolvidos com a idéia de módulos funcionais, preparados para a futura migração para os sistemas de produção. Eles seriam desenvolvidos, portanto, de modo que a integração com as outras partes do projeto fosse facilitada.

Um outro aspecto muito importante que nos fez abordar esta metodologia foi o fato de, ao desenvolver a rede aos poucos, ficar muito mais fácil de se isolar e detectar problemas. Assim, fica mais fácil de adicionar complexidade ao sistema (futuros equipamentos). Por exemplo, ao se garantir que o protocolo funciona adequadamente na

simulação, se houver algum problema ao se conectar o servidor, pode-se ter razoável certeza que o problema se deve a algum problema de interface, e não de protocolo. Segue abaixo um resumo com as etapas de desenvolvimento:

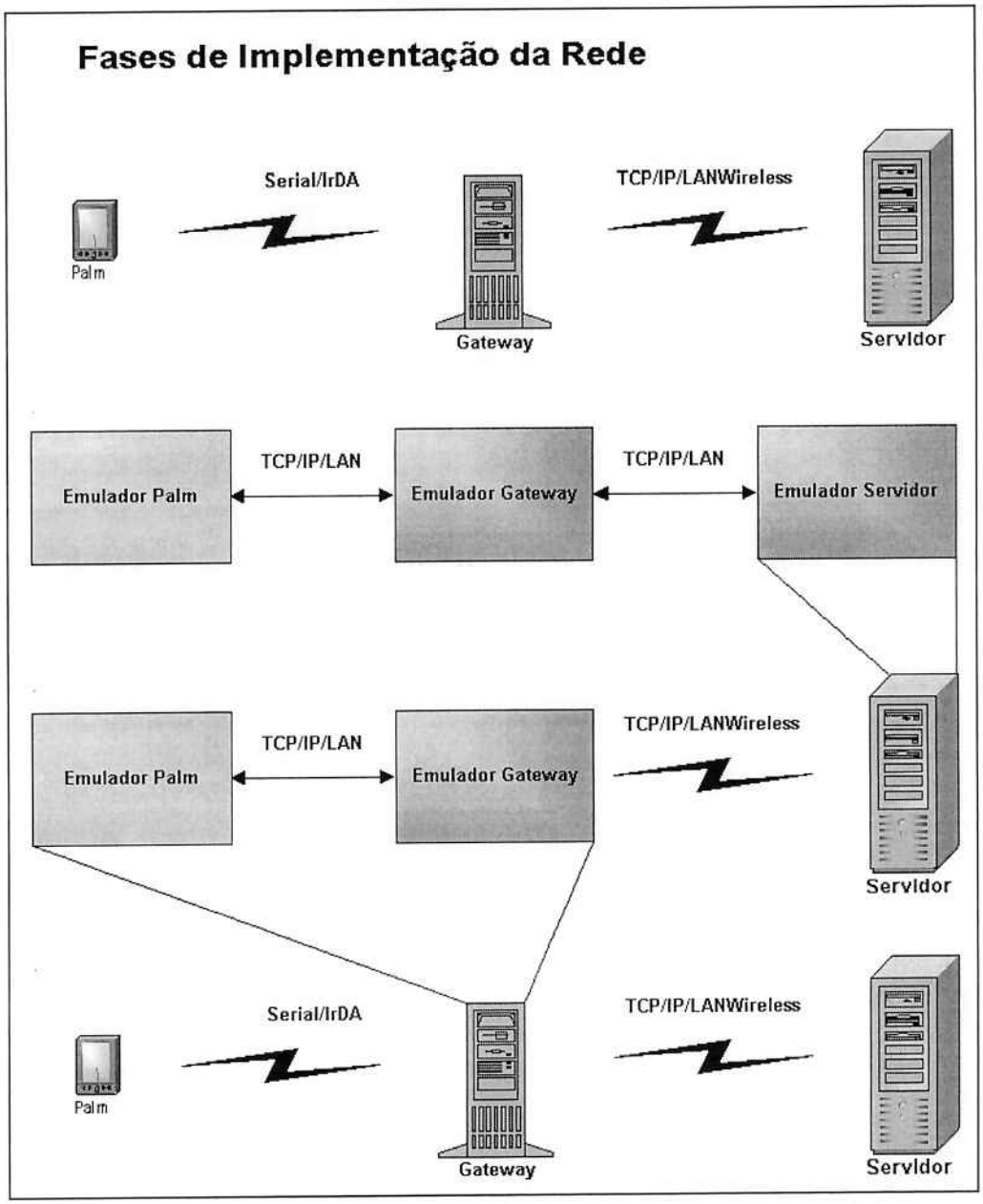


Figura 21: Fases de Implementação da Rede

- *Fase 1*: desenvolvimento dos emuladores em software, utilizando TCP/IP como protocolo de rede em ambas as pontas, utilizando uma rede local como transporte.

- *Fase 2*: substituição do emulador do servidor pelo sistema servidor de produção, rodando o banco de dados e software de inteligência, e o sistema LAN Wireless.
- *Fase 3*: substituição do emulador da Palm pelo equipamento, completando o sistema com a interface infravermelho.

Fase 1 – A rede em software (teste do protocolo)

O primeiro ambiente desenvolvido consistiu no desenvolvimento de três programas em Visual Basic, com transmissão puramente baseada no Winsock (TCP/IP):

- Emulador Palm: programa que implementa a máquina de estados da Palm
- Emulador Servidor: programa que implementa a máquina de estados do servidor
- Emulador Gateway: primeira versão do gateway

Emulador Palm

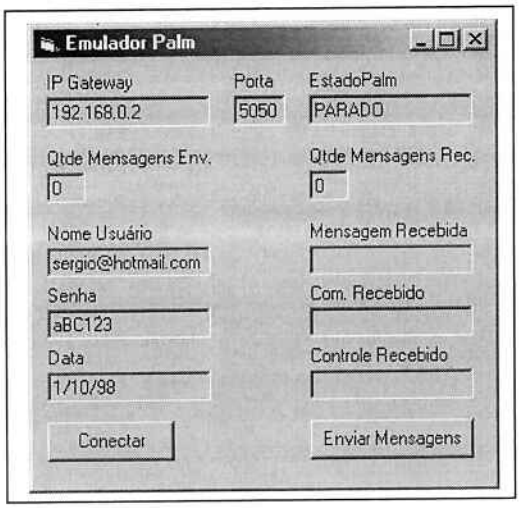


Figura 22: Emulador Palm

O emulador da Palm, mostrado acima, tem a função de simular o funcionamento da Palm, no que diz respeito à transmissão e recepção de dados. Como já mencionado, ela é a implementação em software da máquina de estados da Palm.

Do ponto de vista interno, temos dois blocos principais:

- Máquina de estados: ativada sempre no recebimento de um Campo.
- StringAssembler: módulo de manipulação do Campo.

Adicionalmente, para permitir a simulação, forma incorporados dois botões de controle:

- Conectar: simula o evento "Send" do diagrama de estados, de forma a se iniciar uma transação.
- Enviar Mensagens: botão que é habilitado somente após a autenticação do usuário. Este botão aciona uma segunda janela, mostrada abaixo, que permite o envio de mensagens.

Figura 23: Janela Mensagem

- Enviar mensagem: envia a mensagem escrita para o servidor, junto com os campos de controle adequados. Ao emulador receber aviso do servidor que este recebeu a mensagem, esta janela aparece automaticamente, significando que a Palm pode mandar uma nova mensagem do banco de dados. Fica a critério do usuário quantas mensagens enviar.
- Acabou mensagens: envia aviso ao servidor que a Palm acabou de enviar as mensagens. A partir daí, ela fica pronta para a recepção.

As demais janelas do emulador são necessárias para indicar o endereço IP do gateway, a porta TCP do mesmo, além de janelas adequadas para verificação das mensagens recebidas do servidor. Mais importante é a janela que indica o estado da Palm, exatamente como no diagrama mostrado anteriormente.

Emulador Servidor

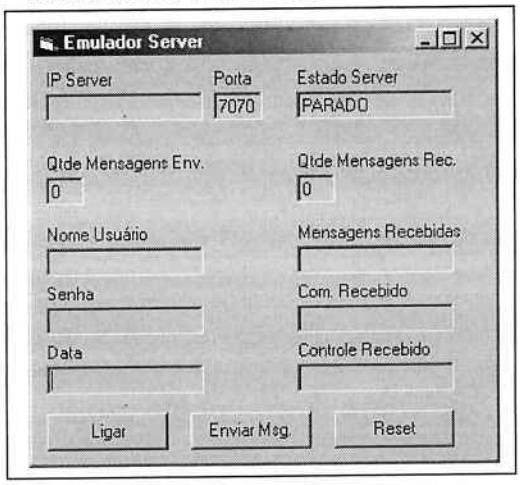


Figura 24: Emulador Servidor

O emulador do servidor, mostrado acima, tem estrutura muito semelhante ao emulador da Palm. Do ponto de vista interno, temos dois blocos principais:

- Máquina de estado: ativada sempre no recebimento de um Campo.
- StringAssembler: módulo de manipulação do campo (igual ao da Palm).

Adicionalmente, para permitir a simulação, foram incorporados botões de controle:

- Ligar: deixa o servidor escutando solicitações de conexão no endereço IP local e porta TCP escolhida pelo usuário.
- Enviar Mgs: botão que é habilitado somente após o término do envio das mensagens pela Palm. Este botão aciona uma segunda janela, mostrada abaixo, que permite o envio de mensagens.
- Reset: fecha o servidor, derrubando a conexão. Este botão foi principalmente utilizado para testar falhas de conexão na rede, porém isto também é possível simplesmente fechando-se qualquer um dos programas (que é uma solução menos elegante, porém tão eficiente quanto)
- Enviar mensagem: funcionamento idêntico ao emulador da Palm.
- Acabou mensagens: envia aviso para Palm que acabou de enviar as mensagens. A transação é então finalizada pelo fechamento da conexão TCP/IP com o gateway.

As demais janelas do Emulador são necessárias para verificação das mensagens recebidas da Palm. Também existe uma janela que indica o estado do servidor.

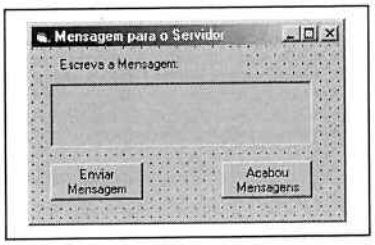


Figura 25: Janela Mensagem

Emulador Gateway

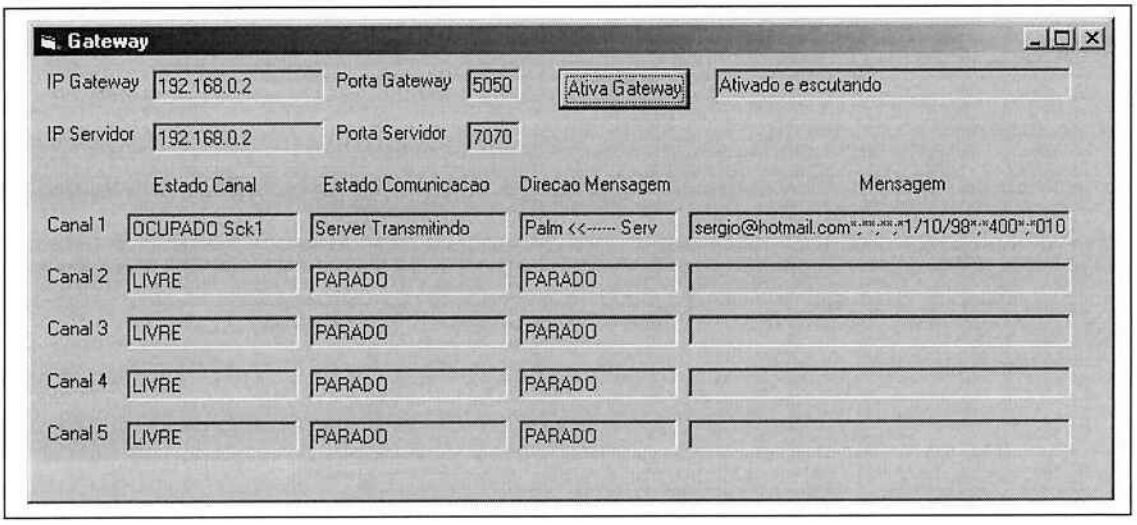


Figura 26: Emulador Gateway

O Gateway é o intermediário entre a Palm e o servidor e, por isso, deve estabelecer duas conexões TCP/IP. Existe um único comando no Gateway:

- Ativa Gateway: este botão ativa o Gateway, deixando uma porta TCP em "listen" para escutar as requisições de conexão da Palm.

Uma vez ativado, o usuário não tem mais nenhum controle sobre o gateway. O gateway está dimensionado para 5 conexões simultâneas, porém só uma será utilizada (no início, o grupo estabeleceu que até 5 palms poderiam se comunicar simultaneamente na mesma interface infravermelho, mas isto não é viável devido a virtualização da serial no infravermelho).

O gateway tem somente a funcionalidade de estabelecer a conexão entre a Palm e o gateway, e passar os dados recebidos de um lado para outro. Para verificação, o programa indica nas janelas os dados e direção de transmissão durante a transação.

Simulação

Para simular a rede, é necessário rodar os três programas simultaneamente, na mesma máquina ou em máquinas diferentes. O único requisito é que se saiba de antemão o endereço IP das máquinas onde os simuladores são executados

O primeiro passo é ativar o emulador do servidor, pelo botão "Ligar". O servidor então estará pronto para receber conexões na porta escolhida pelo usuário. Na janela de endereço, o servidor mostra o seu endereço IP.

Deve-se então ativar o gateway, tomando-se o cuidado de, nas janelas apropriadas, colocar o endereço e porta do servidor. O usuário deve indicar a porta de escuta do gateway, sendo que o endereço é indicado automaticamente após a ativação.

Finalmente, deve-se ativar o emulador da Palm, indicando na janela de endereço o endereço e porta corretos do gateway. O sistema então estará pronto para a simulação.

- Conexão:

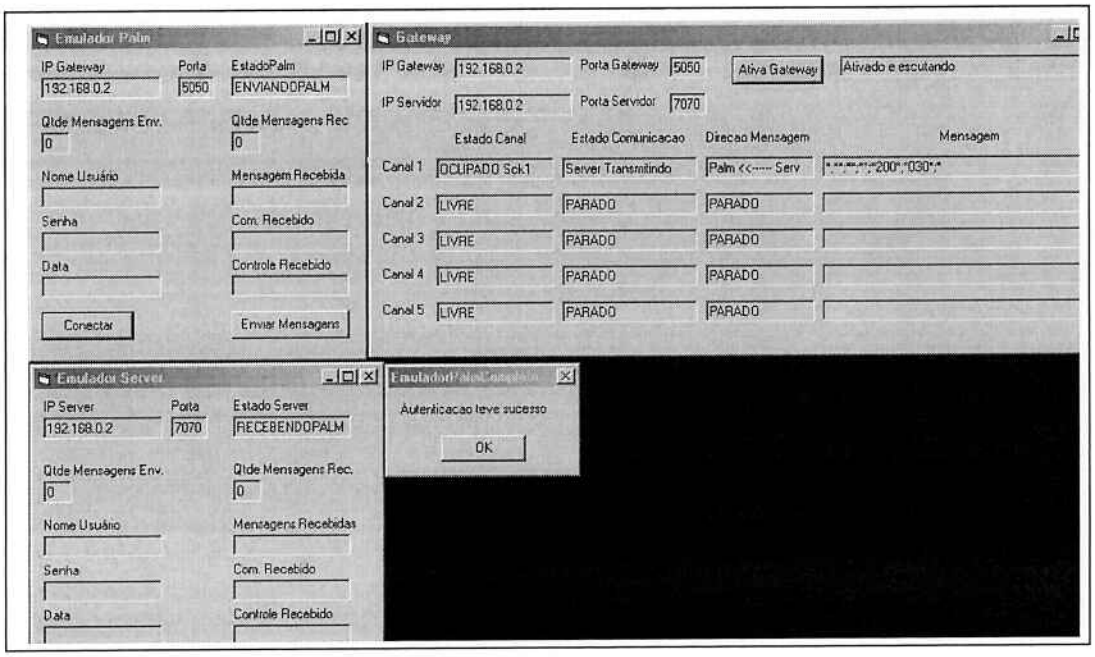


Figura 27: Simulação - Fase de Conexão

Para conectar a Palm ao servidor, basta apertar o botão "Conectar" na Palm. A Palm irá automaticamente para estENVIANDO, e pedirá conexão na porta do Gateway. O Gateway, por sua vez, pedirá conexão na porta do servidor, que ficará pronto para receber a autenticação (estAUTENTICANDO). Se todas as conexões forem fechadas, o

gateway avisa a Palm, que então irá para estAUTENTICANDO e enviará a mensagem de autenticação para o servidor.

O servidor confirmará a autenticação, e uma mensagem de sucesso aparecerá na tela (vide figura). Nesta fase de implantação, o servidor simplesmente aceita a conexão da Palm, uma vez que não tem os recursos do banco de dados para verificar o usuário (na figura, no gateway, podemos ver a reposta de autenticação do servidor, com os campos 200/030, que significa "autenticando/autenticaçãoOK").

Ao receber a confirmação, a Palm vai para estENVIANDOPALM, e o servidor para estRECEBENDOPALM, e o botão "Enviar Mensagens" no emulador da Palm é liberado.

- Palm enviando mensagens:

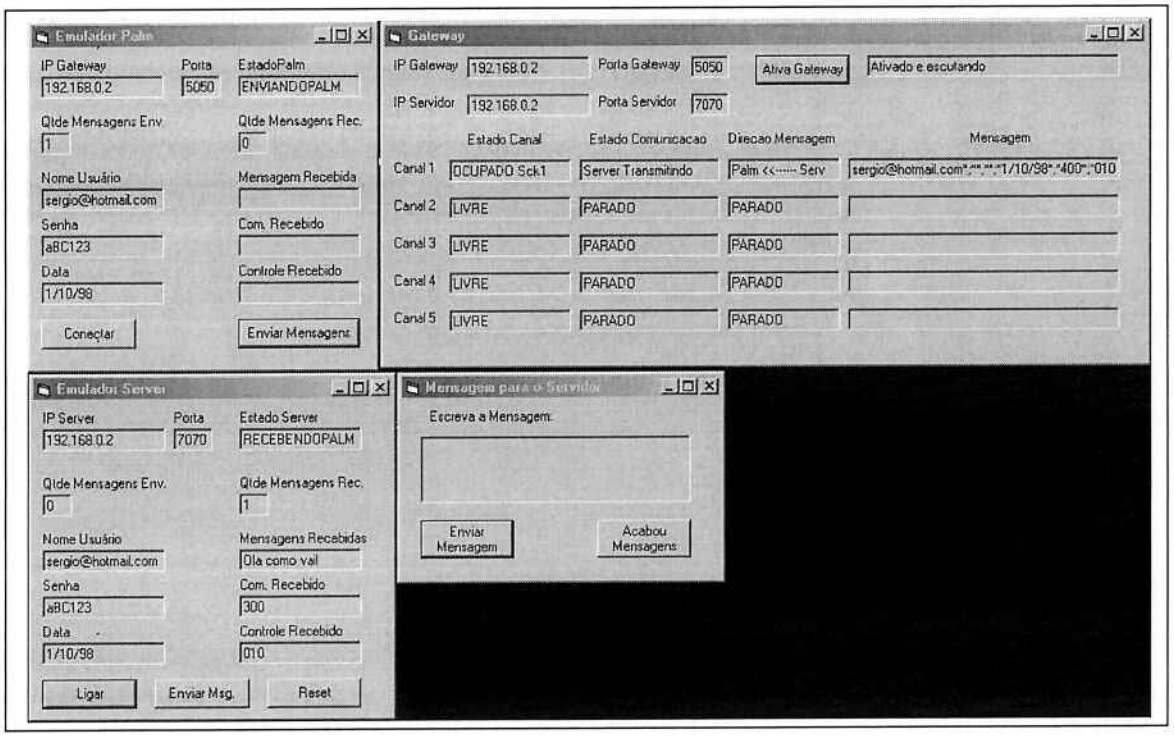


Figura 28: Simulação - Palm enviando Mensagens

Ao apertar o botão "Enviar Mensagens", a tela de mensagens aparecerá, e o usuário poderá simular a Palm enviando mensagens para o servidor. Este, por sua vez, mostrará as mensagens recebidas em sua janela. Toda vez que o servidor confirmar o

recebimento, a janela de mensagens da Palm aparecerá automaticamente, sem necessidade de interferência do usuário.

O processo continua até que o usuário pressione o botão "Acabou Mensagens", indicando o fim das mensagens da Palm. A Palm então vai para estRECEBENDOSERV, e o servidor para estENVIANDOPALM, o que habilita o seu botão de enviar mensagens.

Observe na figura que as janelas de controle tem o objetivo de depurar as informações recebidas pelos programas, o que ajudou muito no desenvolvimento do código fonte dos programas quando havia algum erro. Em especial, observe as janelas que indicam a quantidade de mensagens recebidas e enviadas pelos emuladores.

- Servidor mandando mensagens:

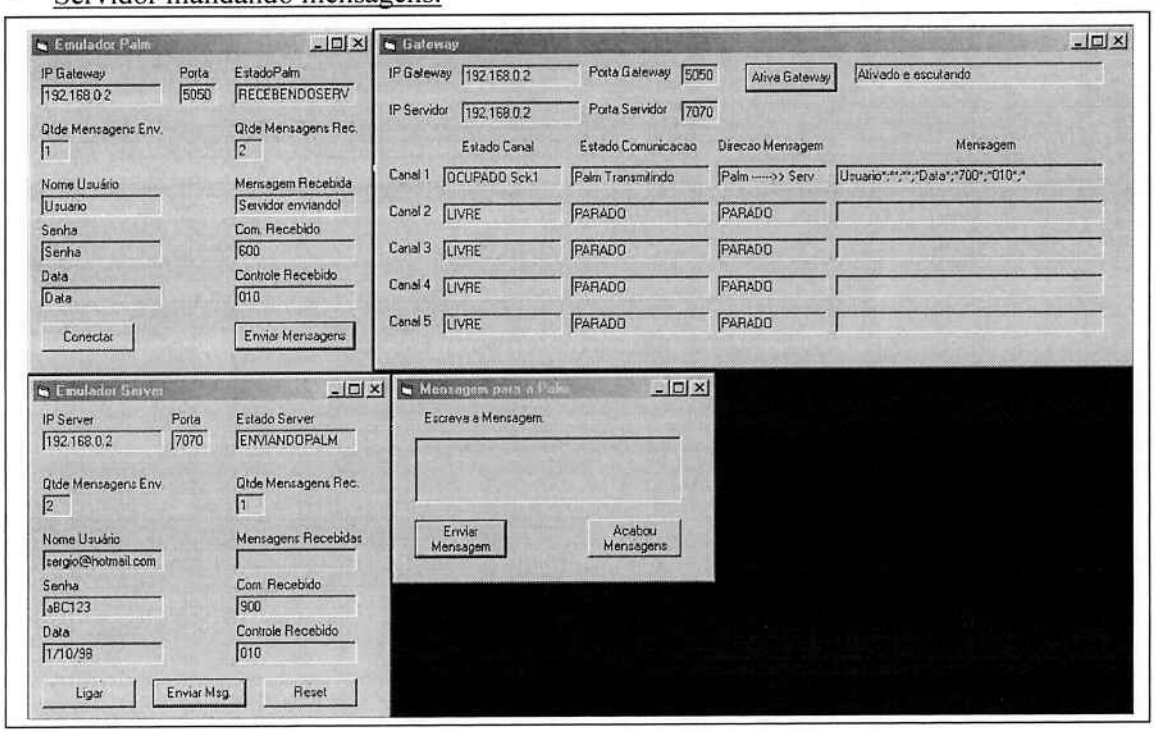


Figura 29: Simulação – Servidor Enviando Mensagens

O processo de envio de mensagens do servidor é idêntico ao da Palm. Observe na figura que os estados das máquinas correspondem aos do diagrama de estados do protocolo.

Ao usuário pressionar o botão "Acabou Mensagens" na tela de mensagens do servidor, o servidor avisa a Palm que as mensagens acabaram e fecha a conexão com o gateway. A Palm fecha a conexão com o gateway ao receber o aviso de fim, ou ao perceber a perda de conexão, o que for primeiro.

- Teste de perda de conexão:

Os emuladores implementam um esquema de "timers" para verificar o estado da conexão. Os timers acionam as subrotinas de verificação periodicamente, em intervalo estabelecido.

Se o gateway verificar a perda de uma conexão, automaticamente fecha a outra. Se algum dos programas verificar a perda de conexão, vão automaticamente para estPARADO, ficando prontos para nova transação.

Os teste de desconexão podem ser feitos fechando-se os programas a qualquer hora da transmissão, ou pelo botão "Reset" do servidor.

Documentação

Para fins de verificação, o grupo disponibilizará os códigos e os executáveis desta simulação. Porém, como não se trata do sistema final, os códigos não estarão adequadamente comentados.

Os executáveis apresentam também alguns pequenos erros de implementação, que não dizem respeito ao funcionamento das máquinas em si, mas sim a pequenos erros no tratamento dos botões quando estes são acionados em momentos incorretos (não foi objetivo nesta fase dar atenção a detalhes de interface com o usuário).

Testes

Os emuladores permitiram ao grupo testar e aprovar o protocolo desenvolvido, com todos os testes correspondendo ao esperado. É claro que, durante a confecção dos programas, vários problemas de implementação foram resolvidos, porém a

especificação do protocolo se mostrou correta (a maioria dos problemas encontraram estavam relacionados com os casos de perda de conexão, que envolviam o tratamento de timers).

Alguns problemas também tiveram que ser resolvidos com relação ao uso do componente Winsock no Visual Basic. Em especial, o fato do fechamento da conexão não ser imediato teve que ser especialmente tratado, através da utilização de uma rotina de espera.

Fase 2 - Integração com o Servidor

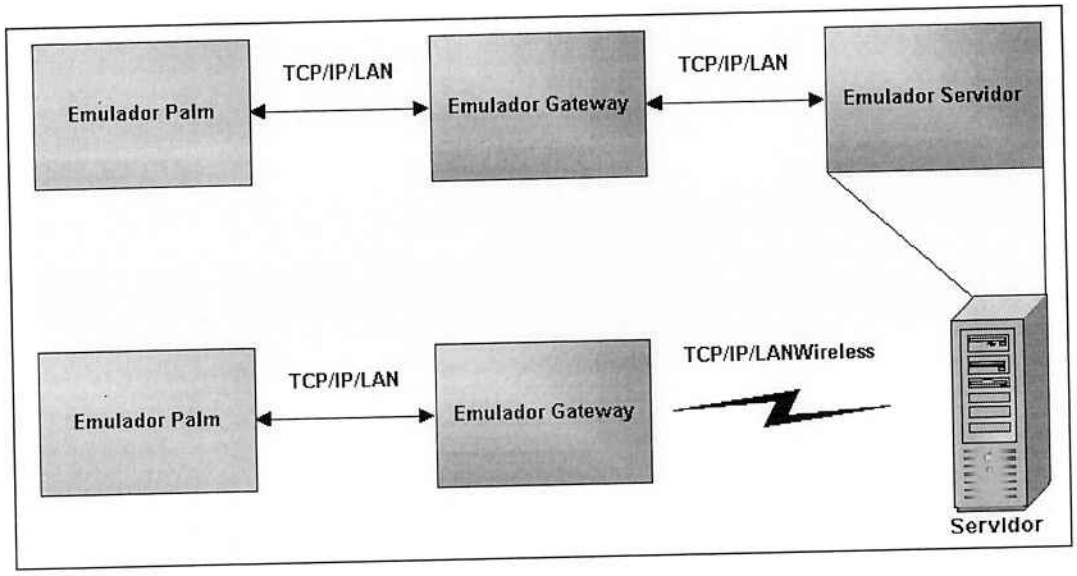


Figura 30: Integração com Servidor

Do Emulador Servidor ao Proxy Servidor

Como toda a lógica de implementação do protocolo já estava pronta no emulador do servidor, o grupo optou por usá-lo como o módulo de interface de rede do servidor. Esta decisão apresentava várias vantagens em relação a idéia original, que era simplesmente descartar o emulador.

A estrutura interna do servidor não precisaria se adaptar ao protocolo de comunicação criado e, portanto, não perderia a generalidade. De fato, o emulador isola o servidor de todos os problemas de comunicação como, por exemplo, checar a rede

para verificar falhas de comunicação. Devido a esta função de "intermediário", chamaremos daqui em diante o "Emulador Servidor" de Proxy Servidor.

Do ponto de vista lógico, para integrar o servidor na rede, poucas mudanças na estrutura interna do emulador são necessárias. São elas:

Evento	Emulador Servidor	Proxy Servidor
Autenticar Usuário	Aceita Autenticação	Verifica autenticação no BD
Receber Mensagem	Mostra mensagem na janela	Escreve mensagem no BD
Enviar Mensagem	Envia mensagem da janela	Carrega mensagem do BD

Neste ponto, ficou evidente que a filosofia de integração de sistemas do projeto estava trazendo os seus dividendos. A decisão acima, junto com as vantagens que ela trouxe, não seriam possíveis se, em paralelo, não houvessem sidos criadas ferramentas adequadas no projeto do servidor. Forma elas:

- A comunicação com o banco de dados se dá através de uso de métodos específicos das classes criadas no desenvolvimento dos componentes, que podem ser diretamente utilizadas no código-fonte dos programas. Estas classe e métodos são disponibilizadas através de bibliotecas dinâmicas(DLLs) registradas no sistema operacional da máquina (detalhes descritos em capítulos anteriores deste documento).
- Os processos internos que ocorrem no servidor, seja para escrita ou consulta no banco de dados, são absolutamente transparentes. Para o programa, tudo se faz por acionamento de funções onde se entram os parâmetros adequados, geralmente obtidos de campos recebidos pela rede, e cujas respostas serão utilizadas para montagem dos campos de respostas, que então serão enviada para a Palm através do gateway.

Estrutura lógica do Proxy Servidor

Na figura apresenta-se um diagrama de blocos que descreve o funcionamento do Proxy Servidor. Recomenda-se que, na leitura do código fonte do Proxy Servidor este diagrama seja consultado (os códigos no CD que acompanha o relatório).

O código fonte do Proxy Servidor está detalhadamente comentado, de forma a indicar claramente onde, como, e com quais classes e métodos o programa utiliza para

acessar o servidor (estes métodos e classes estão descritos em detalhes na seção do servidor).

Manual de uso do Proxy

O proxy foi desenvolvido na linguagem Visual Basic, e pode ser executado em qualquer sistema operacional da família Windows, da versão 95 em diante. Porém, para a integração com o servidor funcionar corretamente, o programa deve ser executado numa máquina onde estejam registradas as bibliotecas de acesso ao banco de dados. Na apresentação deste projeto, o Proxy Servidor será executado na estação servidora junto com o banco de dados.

A interface com o usuário é praticamente idêntica à apresentada para o emulador da Palm (vide abaixo). Nesta versão, forma retiradas toda instabilidade que havia na primeira versão (emulador), e também adicionou-se proteção contra possíveis erros do usuário (ex: apertar botões em horas incorretas).

Figura 31: Proxy Servidor

De fato, para rodar o Proxy Servidor é necessária muito pouca interferência do usuário. Para acioná-lo basta selecionar a porta de operação do proxy na janela "Porta", e depois apertar o botão "Ligar". Deste ponto em diante o programa é totalmente auto-suficiente, não necessitando mais nenhum outro tipo de interferência do usuário.

Ao ligar o proxy, o programa mostra o endereço IP na janela "IP Server", e o estado da máquina de estados interna na janela "Estado Server".

O botão "Reset" está disponível se o usuário quiser reiniciar o proxy. Adicionalmente, existem janelas que mostram informações de controle (estas janelas foram mais utilizadas no desenvolvimento do software, para depuração de erros, mas foram deixadas por oferecer ao usuário informação interface visual com as informações).

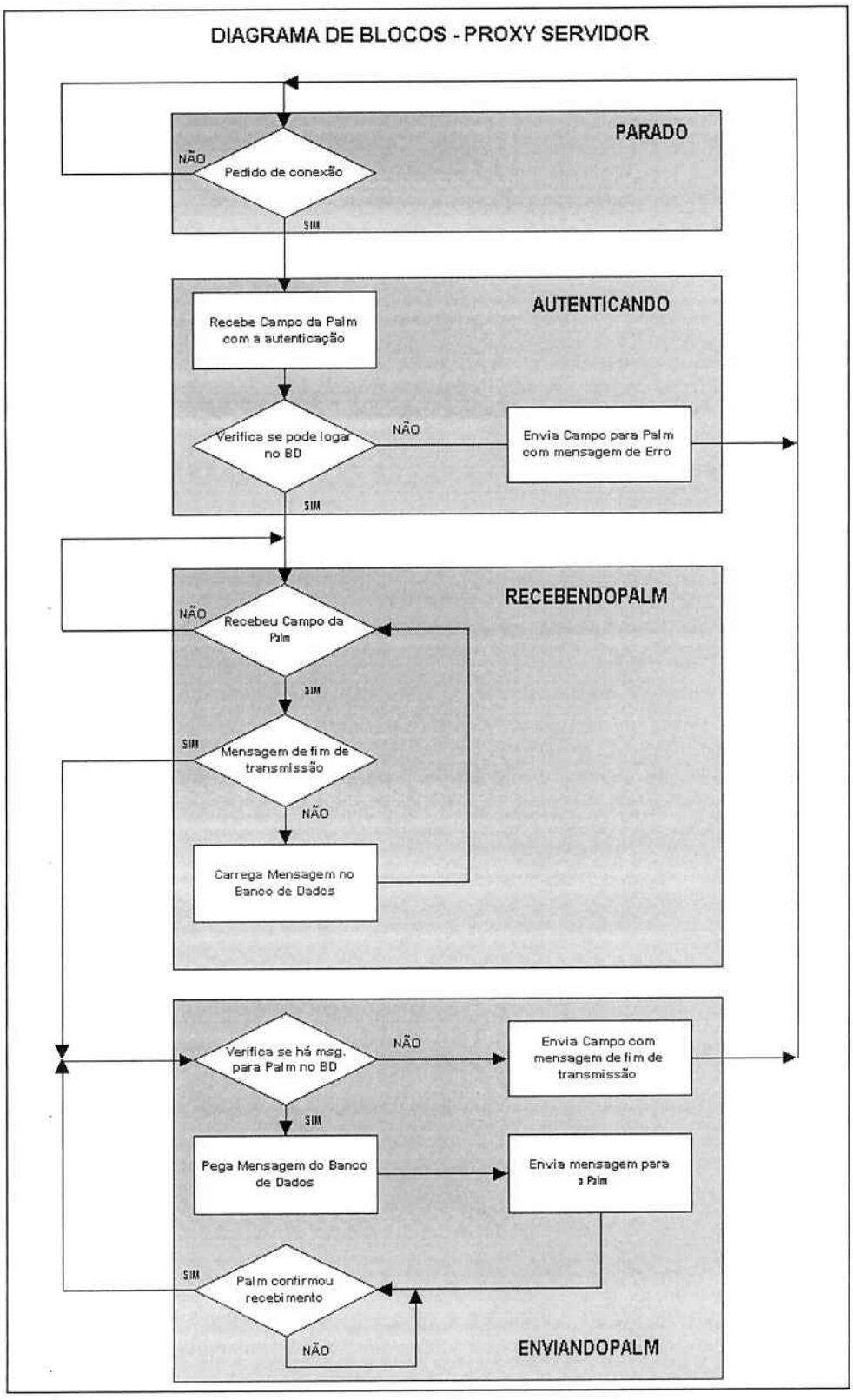


Figura 32: Diagrama de Blocos Proxy Servidor

Fase 3 - Integração com a Palm

Com o sistema funcionando somente com o servidor, a rede foi testada utilizando o emulador da Palm. Com o emulador, foi possível testar toda a lógica de autenticação, envio e recebimento de mensagens.

Para confirmar as transações, consultávamos manualmente as tabelas do banco de dados no servidor. O método de depuração consistia, basicamente, dos seguintes passos:

- Verificar a lógica de autenticação, tentando autenticar usuários válidos e não válidos;
- Se autenticado, verificar a inserção das mensagens enviadas pela Palm após a transação;
- Verificar flag de mensagem "lida" ou "não lida" , após envio de mensagens pelo servidor.

Após testar exaustivamente todas as possibilidades possíveis com o emulador, consideramos completa e testada a integração do servidor na rede. Seguimos então com a última parte do projeto, integrar a Palm.

Traduzindo a máquina de estados para o Palm OS

Nesta fase, já havia sido desenvolvida na Palm toda a lógica de acesso ao banco de dados, bastando então apenas juntá-la com a lógica da máquina de estados da comunicação para finalizar o software da Palm.

Ao contrário do que aconteceu do lado do servidor, não seria possível fazer do emulador Palm um módulo de interface deste software, uma vez que a Palm tem restriões de memória e não aceita o formato dos programas criados em VB.

Fez-se então uma tradução da máquina de estados do protocolo para a Palm. Obviamente, a lógica de funcionamento na Palm preservou exatamente o que havia sido desenvolvido no emulador.

Segue uma descrição em diagrama de blocos da lógica de comunicação da Palm.

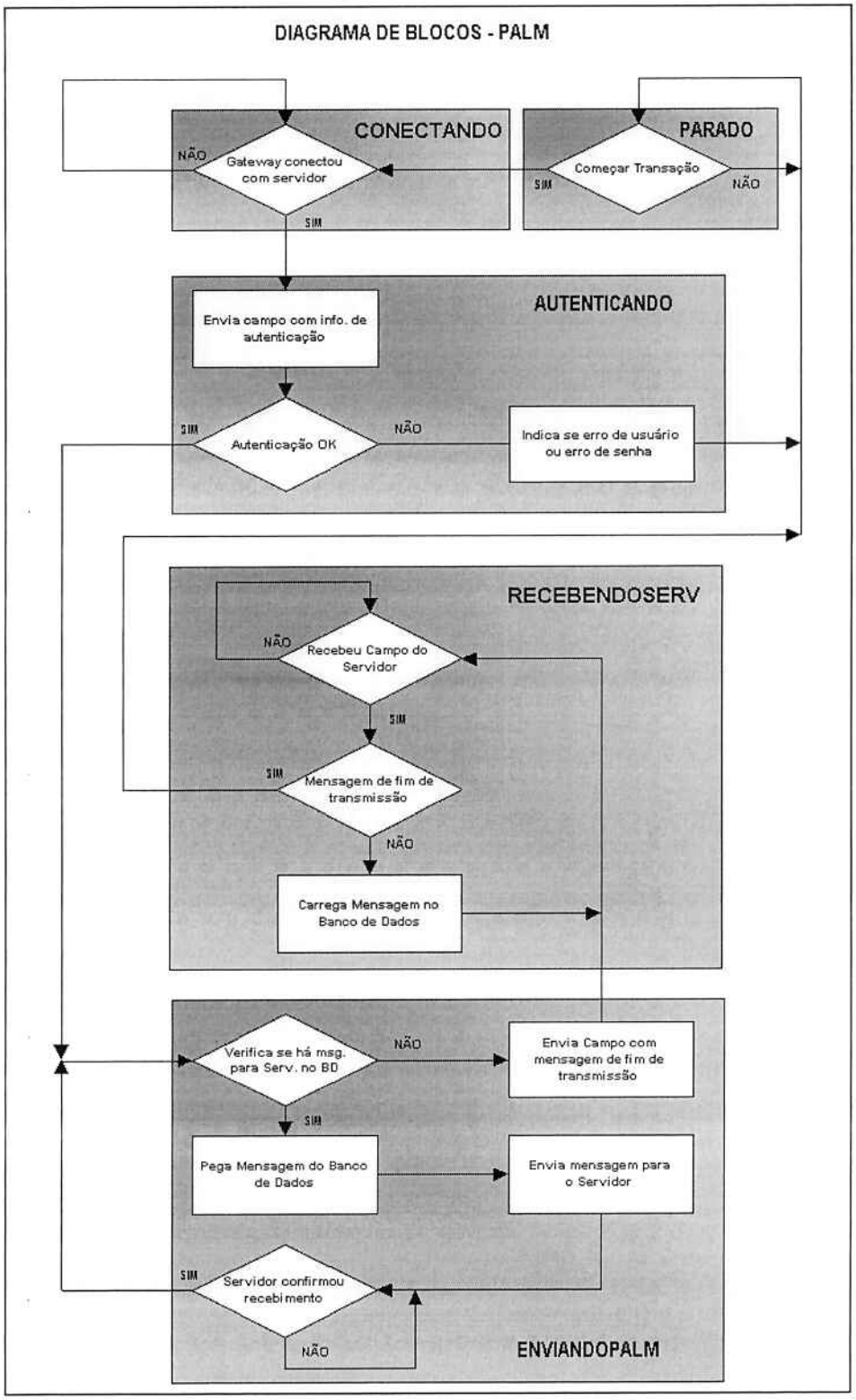


Figura 33: Diagrama de Blocos da Palm

Verificando erro de conexão

Uma segunda diferença fundamental na integração da Palm foi o link infravermelho em si. Ao contrário do TCP/IP, que fornece a qualquer momento o estado da conexão, o tratamento de perda de conexão na serial tem de ser feita no esquema de mensagens de aviso, ou keepalives.

Em termos de implementação, os equipamentos precisam efetivamente checar um ao outro de forma ativa através da serial, com envio de mensagens periódicas entre ambos (no caso do TCP/IP, rotinas de timer checam periodicamente o estado dos Sockets, que indicam ou não a perda de comunicação).

Caso algum equipamento não responda o keepalive, a conexão é considerada perdida, e a transação é automaticamente encerrada (estPARADO). Caso a interface serial acuse erro na transmissão, a conexão também é considerada encerrada.

Modificando o Gateway

O último passo para completar o sistema é adaptar o gateway para comunicação serial. Até agora, o gateway comunicava-se via TCP/IP tanto com o emulador Palm quanto com o servidor, mas agora se faz necessária a substituiçao do bloco de comunicação da Palm para utilizar a serial sobre o infravermelho.

Na implementação, a única mudança é a utilização de rotinas de envio e recebimento de dados via serial, e a utilização de rotinas de keepalive para verificar a conexão com a Palm. A estrutura final do gateway fica como mostrado na próxima figura, que ilustra seu funcionamento na forma de um diagrama de blocos (vide próxima página).

Testes

Os testes com o sistema final foram idênticos aos realizados desde o início do desenvolvimento do sistema, com a validação nas fases de autenticação, transmissão da Palm, transmissão do servidor, e finalização da transação. Os testes visaram validar

tanto o funcionamento do protocolo quanto a integração do mesmo com as rotinas de banco de dados.

Claro que, nesta fase, foi dado ênfase no tratamento da comunicação pelo infravermelho, que foi tendo que ser ajustado conforme os problemas surgiam. Também foram feitos ajustes para se achar a frequência ideal dos keepalives.

Os testes finais visaram garantir a estabilidade do sistema, procurando-se induzir o maior número e tipos de erros possíveis no sistema (erro de interface, perda de conexão, etc...). Conforme as instabilidades foram detectadas, os programas foram sendo aprimorados, até deixar o sistema confiável.

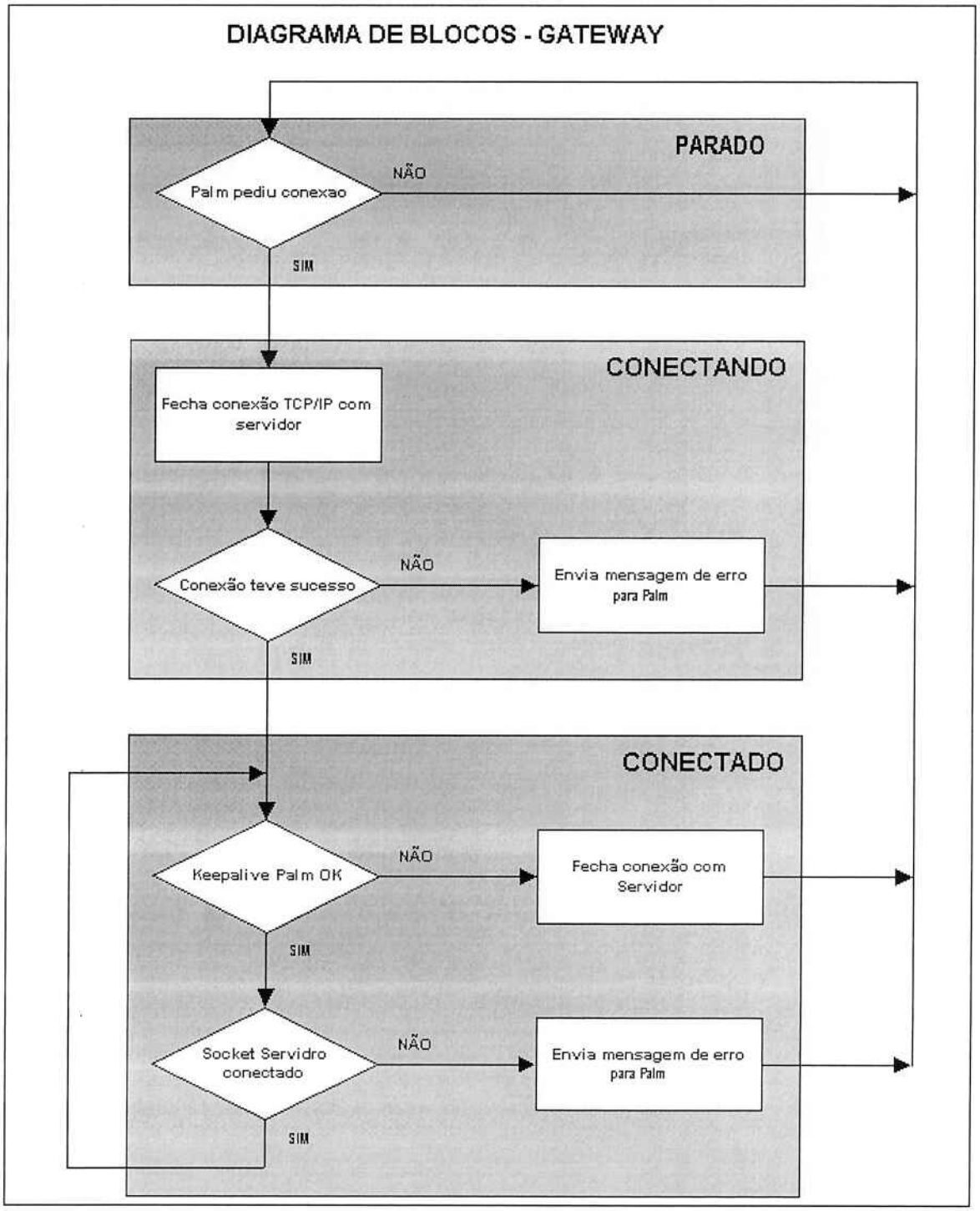


Figura 34: Diagrama de Blocos do Gateway

O Sistema Final – Rede Integrada

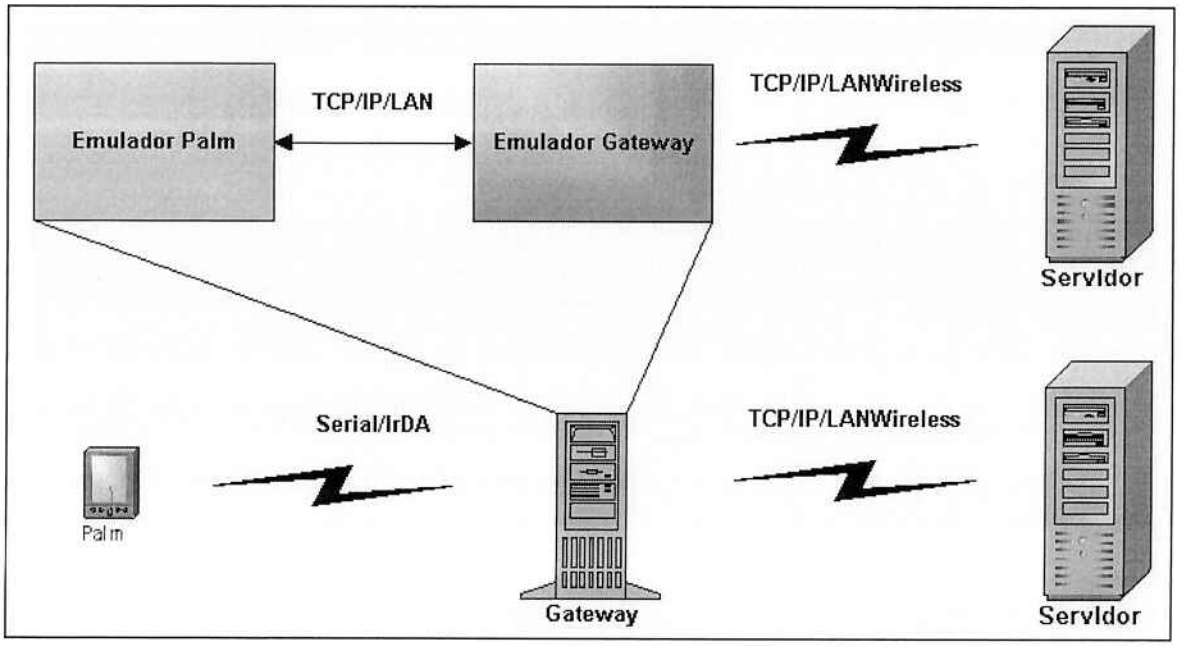


Figura 36: O Sistema Completo

O sistema final, apresentado pelo grupo, é somente uma pequena amostra do potencial pleno do projeto. Temos no sistema apenas um servidor, um gateway e uma Palm, que foram os recursos conseguidos pelo grupo até o momento.

Porém, o sistema já está praticamente pronto para migrar para a arquitetura completa, com vários gateways na rede se comunicando simultaneamente com o servidor. Para realizar este passo, só é necessário um pequeno ajuste no Proxy Servidor, para que ele aceite a conexão de múltiplos gateways.

De qualquer forma, para o que foi originalmente proposto pelo grupo, consideramos que todos os objetivos foram atingidos.

INTERFACE PALM

A programação do lado da Palm foi inicialmente baseada no compilador Metrowerks CodeWarrior Lite 4.01, que é uma versão gratuita disponível desta ferramenta, que é a ferramenta oficial de desenvolvimento para Palm OS, baseada em C++. A única restrição é que toda aplicação desenvolvida com ela, por ser gratuita, passa por uma tela inicial gerada pelo Codewarrior Lite.

Atualmente, existem versões bem mais recentes deste compilador (a mais recente é a versã0 8.0), porém, não há versão Lite gratuita para download. A restrição que isto impôs ao grupo é de que as APIs utilizadas na programação não são as mais recentes. Está sendo usado o SDK 3.0 (API das funções do PalmOS – Software Development Kit).

Há hoje em dia também uma implementação de referência da plataforma J2ME (Java 2 Micro Edition), que oferece recursos para programação em Java para o Palm OS. Acreditamos que seria mais fácil e menos trabalhoso elaborar um sistema utilizando esta plataforma, no entanto, no início do desenvolvimento deste projeto ela não estava disponível, e não houve tempo hábil para um novo desenvolvimento em Java.

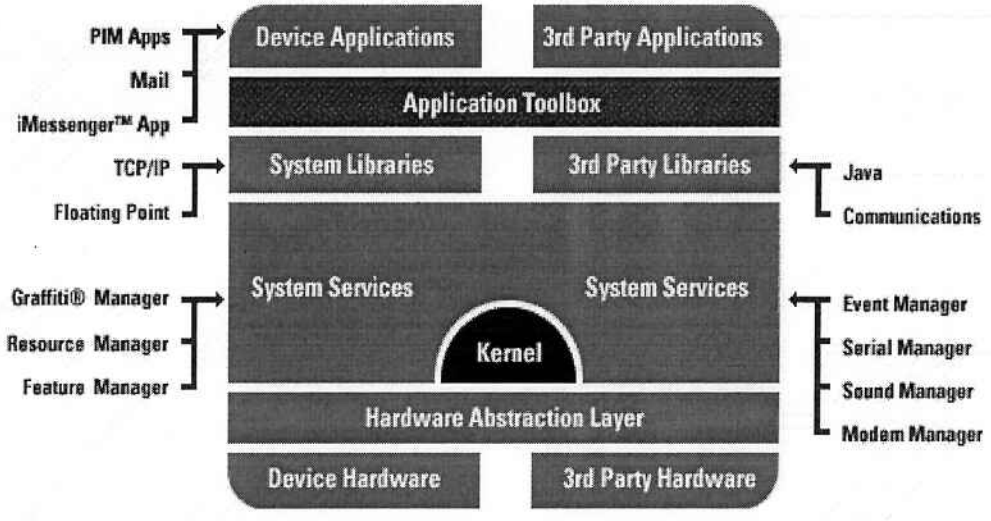


Figura 37: Arquitetura de Programação para a Palm

O compilador Codewarrior compila código na linguagem C/C++ para o processador 68K da Motorola, rodando o PalmOS. Além do compilador, ele oferece uma interface de desenvolvimento IDE amigável, que facilita o desenvolvimento das telas e recursos visuais disponíveis na Palm.

Este compilador gera código objeto para diversas plataformas, mas a versão utilizada foi a que continha as bibliotecas e plug-ins necessários para a geração de código para os processadores 68K da Motorola, que tem uma de suas variantes como processador das Palm Pilots.

Fez parte também dos recursos de desenvolvimento o emulador Palm Emulator, desenvolvido pela Palm Inc., com a ROM equivalente à Palm utilizada nos testes, que acompanha o CodeWarrior, e se mostrou essencial para o teste mais rápidos das aplicações, visto que não foi necessário fazer o deploy da aplicação para a Palm Pilot a cada nova funcionalidade ou correção implementadas.

A interface de desenvolvimento (IDE) oferecida pelo CodeWarrior para programação para Palm OS oferece, além do gerenciador de projetos e do Emulator, infra-estrutura necessária para debug, construção de recursos gráficos através de interface gráfica (Constructor) e uma ferramenta para desenvolvimento de Conduits (adaptadores no PC para troca de informações com a Palm utilizando o HotSync).

Atualização:

Após ter praticamente terminado todo o desenvolvimento para a Palm, e faltando apenas a codificação das funções de envio de dados pela porta infra-vermelha, a equipe percebeu que o CodeWarrior Lite 4.0 não oferecia suporte (e portanto não compilava) as funções oferecidas pelas últimas versões da API do PalmOS para transmissão de dados seriais pela porta infra-vermelha.

Um pouco de pesquisa, e recomendações do ClubePalm (http://www.clubepalm.com.br/) levaram-nos a adotar a plataforma de desenvolvimento Falch Net Studio 2.5 (http://www.falch.net/). Esta plataforma oferece recursos e um ambiente de desenvolvimento semelhante ao oferecidos pelo CodeWarrior, no entanto,

compila código de versões mais recentes do SDK. Houve um certo re-trabalho no sentido de adequar o programa já pronto, uma vez que muitas funções do SDK 3.5 utilizado são distintas das de versões anteriores.

Programando para Palm OS

Para o projeto em questão, antes de dar início ao desenvolvimento propriamente dito, foi especificado o escopo do programa que deveria rodar na Palm, de acordo com os seguintes pré-requisitos:

- Aplicativo simples e amigável, com interface intuitiva;
- Tamanho reduzido para não ocupar espaço desnecessário na Palm do usuário;
- Processamento reduzido (maior parte do processamento precisa ser do lado do PC)

De acordo com as especificações gerais do projeto, o sistema na Palm deveria ser capaz de armazenar mensagens para leitura, numa 'Caixa de Entrada', além de permitir que novas mensagens fossem escritas e arquivadas numa 'Caixa de Saída'. Posteriormente, deveria haver uma opção para o usuário fazer a conexão da Palm com o PC através de sua porta infra-vermelha. Esta conexão faria com que as mensagens do PC fossem descarregadas e armazenadas na 'Caixa de Entrada' (através de um protocolo que deveria ser implementado na Palm), e as mensagens da 'Caixa de Saída' deveriam ser enviadas através de um protocolo de envio de dados, e posteriormente excluídas desta caixa de mensagens.

O primeiro passo então, foi, utilizando o módulo Constructor do Codewarrior, desenhar todas as telas. Uma vez desenvolvidas as telas, foi possível dar início ao desenvolvimento dos módulos que tratam todos os eventos possíveis no sistema.

Vale ressaltar que a programação para o ambiente Palm OS é bastante trabalhosa e difícil, uma vez que é necessário cuidar do gerenciamento de memória do dispositivo, visto sua escassez em dispositivos hand held dotados de Palm OS. Esta preocupação

sempre acompanhou o projeto, e foi essencial para manter o pequeno tamanho do programa final.

Cabe observar também que a programação para a Palm é totalmente orientada a eventos, ou seja, o módulo principal consiste num loop infinito, que fica observando a ocorrência de eventos, que podem ser de vários tipos (clique do usuário num botão, clique do usuário numa opção de menu, entrada de dados, início de beaming, bateria fraca, etc...). De acordo com o tipo e prioridade do evento, um módulo especial é chamado para tratá-lo, conforme o diagrama abaixo.

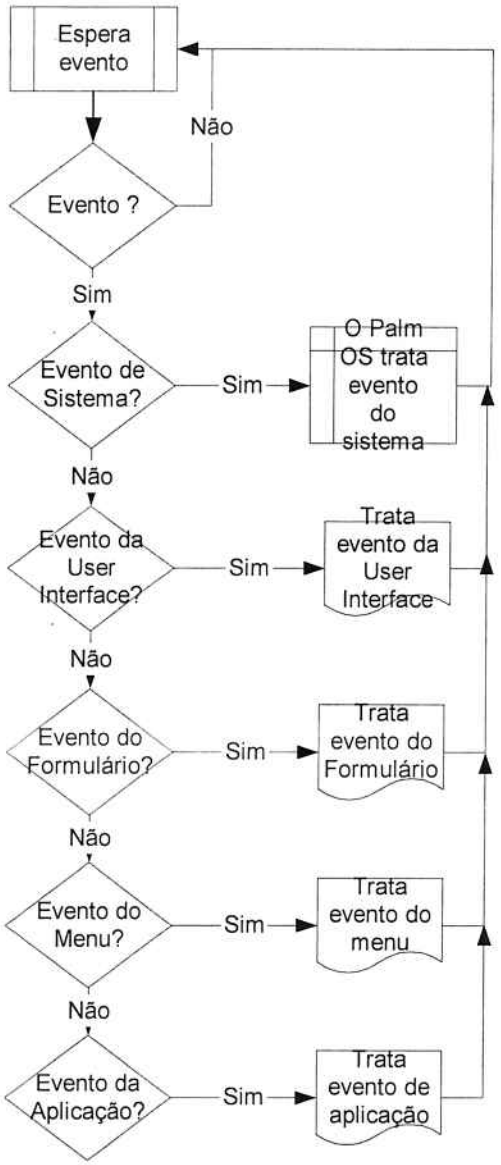


Figura 38: Programação Orientada a Eventos

Seguem algumas telas comentadas:

Tela 1: No menu da Palm, há opção para ver informações sobre os autores do projeto.

Tela 2: A primeira tela que aparece no programa da Palm. O usuário pode escolher se quer gerenciar suas mensagens (consultar caixa de entrada / saída), ou conectar com o servidor através do infra-vermelho para atualizar as mensagens.

Tela 3: Ao escrever uma nova mensagem, o usuário deve necessariamente conhecer o login do destinatário de sua mensagem. Cada mensagem possui, além do destinatário, um assunto e um texto.

Tela 4: As caixas de mensagem – Cada nova mensagem que o usuário escreve é armazenada em sua caixa de saída. As mensagens na caixa de saída são enviadas para o PC quando há uma nova conexão através da interface infra-vermelha. As novas mensagens vindas do PC, são armazenadas na caixa de entrada.

Tela 5: Quando o usuário clica em Conectar, na tela inicial, ele deve em seguida digitar seu login e senha, para que a conexão com o PC, por meio da interface infra-vermelha seja efetuada, e as caixas de mensagens sejam atualizadas.

INFRA-ESTRUTURA NECESSÁRIA

O desenvolvimento deste projeto envolveu o uso de uma infra-estrutura relativamente grande, tanto no que diz respeito a software, tanto na parte de hardware. Nesta seção são descritos todos os componentes, os sites de seus fabricantes, e sua função na arquitetura do projeto (sem considerar as partes desenvolvidas pela equipe).

Hardware:

- *Servidor*: Um computador PC, com processador Pentium III de 1.1 GHz, dotado de 512MB de memória RAM e 30GB de disco foi utilizado como servidor do sistema. Este servidor teve que possuir estas características devido ao software pesado que teve que suportar.

- *Gateway*: Um computador PC, com processador Pentium III de 750 Mhz, dotado de 256 MB de memória RAM e 20 GB de disco foi utilizado como gateway do sistema. A configuração deste computador pôde ser menor que o do servidor, visto que rodará software mais leve. Tem como requisito indispensável possuir uma porta serial disponível para conexão com o dispositivo infra-vermelho.

- *Placas de Rede PCI Wireless Orinoco*: considerando o objetivo de desenvolver um sistema de comunicação, foram usadas duas placas de rede PCI wireless Orinoco Silver, da Lucent Technologies para prover para a camada de rede transparência das camadas física e de enlace na comunicação entre servidor e gateway (http://www.lucent.com/).

- *Extended Systems XTNDAccess IrDA PC Adapter*: dispositivo de comunicação infra-vermelha, fabricado pela Extended Systems (http://www.extendedsystems.com/), que deve estar conectado a uma porta serial disponível no gateway. Uma vez que os drivers utilizados não são específicos deste fabricante, qualquer dispositivo compatível com o padrão IrDA (http://www.irda.org/) poderia ser utilizado.

- *Palm Pilot V*: Foi utilizada uma PalmPilot III, da Palm Inc. (http://www.Palm.com/), utilizando o Palm OS 3.3. (o sistema foi testado também com a Palm V). Poderia ser usado qualquer dispositivo handheld (PDA), dotado de porta infra-vermelha, que suportasse o sistema operacional Palm OS 3.0.

Software:

No Gateway:

- *Windows 2000 Professional*: Foi utilizado o sistema operacional Windows 2000 Professional. Poderia ser utilizado qualquer sistema operacional Windows, da Microsoft, desde que dotado de suporta para rodar componentes COM+ (padrão a partir do Windows 98, e com driver disponível para Windows 95) – (http://www.microsoft.com/).

- *IrComm2k v1.1.0*: Este driver mapeia a interface infra-vermelha para uma porta serial COM virtual, permitindo que a mesma API disponível no Windows 2000 para a porta serial seja usada para a porta infra-vermelha, facilitando em muito a programação, visto que componentes prontos disponíveis na API do Windows 2000 pudessem ser utilizados para programação em Visual Basic. Este driver é oferecido numa verão paga pelo fabricante da interface IrDA, mas foi encontrada uma versão desenvolvida por um estudante sob a licença de software livre GNU, no site (http://www.gsm.org.uk/gsm.html)

No Servidor:

- *Windows 2000 Server*: Este sistema foi adotado devido à facilidade com que oferece recursos de servidor WEB com suporte à páginas dinâmicas MS Active Server Pages (ASP) – IIS 5.0, e por permitir o uso da tecnologia COM+ da Microsoft, através de seu Gerenciador de Componentes. (http://www.microsoft.com/)

Para Desenvolvimento:

- *MS Visual Studio 6.0*: foram desenvolvidos componentes utilizando-se o MS Visual Basic 6.0, que foram registrados no servidor, e utilizados tanto no desenvolvimento das páginas web (ASP) que rodam sob o servidor IIS 5.0, com extensões de FrontPage instaladas, quanto para o desenvolvimento da aplicação residente no gateway. O desenvolvimento das páginas ASP foi feito utilizando-se o MS Interdev 6.0, por seu poderoso ambiente de desenvolvimento.

- *MetroWerks CodeWarrior Lite 4.01*: ambiente de desenvolvimento para o Palm OS, que gera arquivos executáveis no formato PRC, e que oferece uma biblioteca gráfica para definição das telas da aplicação na Palm. Já existem versões mais recentes do software, no entanto, optou-se por esta, por ser a última versão disponibilizada gratuitamente que permite o uso de todos os recursos necessários para o projeto.

- *Falch Net Studio 2.5*: ambiente de desenvolvimento para o Palm OS, que gera arquivos executáveis no formato PRC, e que oferece uma biblioteca gráfica para definição das telas da aplicação na Palm. Por ser gratuita, pelo menos para fins não comerciais, pudemos usar esta que é a últimav versão do ambiente de desenvolvimento.

PRÓXIMOS PASSOS

Visando transformar este projeto num produto comercialmente viável, são descritas a seguir algumas diretivas que deveriam ser seguidas com este objetivo, trazendo para o sistema o estado da arte da tecnologia vigente atualmente.

- *XML para troca de dados (parser XML na Palm)*

  A troca de dados entre a Palm e o Servidor poderia se dar utilizando-se XML e padrões de TAGs pré-definidos para este tipo de aplicação. Isto facilitaria a a interoperabilidade do sistema com outros (no caso específico da aplicação desenvolvida, outros sistemas de troca de mensagens). Considerando que já existem bons parsers XML para as mais diversas linguagens de programação no Desktop PC, faltaria encontrar ou até mesmo desenvolver um parser para o Palm OS.

- Vale a pena migrar para blue tooth?

  Deve ser feita uma pesquisa para verificar se o desenvolvimento utilizando blue tooth compensa, visto que a maioria dos modelos de Palm não oferece entrada para este novo padrão. Seria importante um estudo de mercado que analisasse as expectativas em relação ao blue tooth, o preço de atualização das Palms, etc.

- Desenvolver novamente os componentes e o servidor gateway usando arquitetura .NET da Microsoft

  Utilizando a nova plataforma .Net da Microsoft, é possível instanciar objetos e prover serviços através da Internet sem a necessidade de configuração complicada nas máquinas que rodam os aplicativos clientes. Isto significa que não é necessário a implementação de um sistema de comunicação TCP entre os módulos do sistema, visto que os objetos do servidor poderiam ser instanciados diretamente no gatewa, simplificando todo a arquitetura do sistema.

- Desenvolver aplicação para Plataforma Windows CE
  Visto que o PalmOS não é exclusivo, e existe ainda uma vasta gama de equipamentos baseados no Windows CE, também dotados de interfaces infra-vermelhas, seria interessante o desenvolvimento de uma versão do aplicativo atualmente existente na Palm para este sistema operacional.

- Utilização da porta USB ao invés da serial

  Visto que a porta USB é mais veloz que a serial, e os computadores modernos são capazes de reconhecer automaticamente mudanças de hadware neste porta, seria interessante substituir a comunição da porta serial por uma na porta USB. O equipamento de interface para isso existe, e pode ser comprado facilmente, por um preço pouco maior que o equivalente para a porta serial. A grande mudança no software seria encontrar os componentes do Visual Basic que se comunicassem com facilidade por esta porta, ou que virtualizassem o dispositivo infra-vermelho da mesma maneira que fazemos atualmente.

- Desenvolvimento de novas aplicações

  Troca de e-mails é apenas uma das infindáveis idéias de aplicações baseadas neste sistema. Recentemente saiu na Revista Veja que o sistema Hands, que oferece informações de restaurantes e cinemas pela Palm instalou terminais nos principais aeroportos nos quais seus usuários poderiam atualizar as informações em suas PalmPilots através da porta infra-vermelha destes. No caso do projeto em questão, diversas empresas poderiam oferecer serviços personalizados, através de terminais (nossos gateways), como extratos-bancários, e-mails, etc...

- Implementação do Gateway em hardware

  Passos para o futuro incluem a possibilidade de implementar o gateway completamente em hardware, numa pequena caixa com uma interface infravermelho e uma antena para a LAN wireless. Desta forma, além do gateway

utilizar um espaço mínimo, permitiria a implementação de redes em ambientes corporativos. Poderia-se trabalhar num modelo de atualização de "firmwares", conforme atualizações do gateway forem necessárias.

CONCLUSÃO

Praticamente um ano foi o tempo que o desenvolvimento deste projeto tomou. Acreditamos que, neste período, pudemos vivenciar todas as fases do desenvolvimento de um projeto de grande porte (apesar de este não ser particularmente um projeto grande), desde as fases de planejamento e especificação, até as fases de implementação e testes.

Consideramos este trabalho essencial para que pudessemos por em prática alguns dos principais conceitos estudados em teoria nestes anos de faculdade, tendo agregado um enorme valor à formação dada pela Escola.

Acreditamos termos atingido os objetivos apresentados no início deste documento de maneira satisfatório, e enxergamos um grande potencial deste projeto, caso seja elaborado um planejamento maior, e uma nova especificação de seus requisitos para atender tanto ao mercado acadêmico quanto ao mercado comercial.

BIBLIOGRAFIA

Sites / Mídia Eletrônica:

- MICROSOFT CORPORATION. EUA. Site institucional da empresa. Disponível em http://www.microsoft.com/. Último acesso em Novembro/2001
- MSDN LIBRARY. EUA. Site com conteúdo para desenvolvedores Microsoft. Disponível em http://msdn.microsoft.com/. Último acesso em Novembro/2001.
- EXTENDED SYSTEMS INC. EUA. Site institucional da empresa. Disponível em http://www.extendedsystems.com/. Último acesso em Novembro/2001
- INFRARED DATA ASSOCIATION. EUA. Site da ssociação que define os padrões do Infra-vermelho. Disponível em http://www.irda.org/. Último acesso em Novembro/2001
- PALM INC. EUA. Site institucional da empresa, disponível em http://www.Palm.com/. Último acesso em Novembro/2001
- PALM DEVELOPERS SITE. EUA. Site com recursos para desenvolvedores. Disponível em http://www.palmos.com/dev/. Último acesso em Novembro/2001
- PALM GEAR. EUA. Site de downloads de softwares para PalmOS,. Disponível em http://www.palmgear.com/. Último acesso em Novembro/2001
- CLUBE PALM. Brasil. Site com material para desenvolvedores Palm. Disponível em http://www.clubepalm.com.br/. (Contato com Márcio Alexandroni). Último acesso em Novembro/2001
- FALCH.NET. EUA. Site do DeveloperStudio for PalmOS. Disponível em http://www.falch.net/. Último acesso em Dezembro/2001.

Livros:

- Pattison, Ted. **Programming Distributed Applications with COM+ and Microsoft Visual Basic 6.0**, 2nd Edition, Microsoft Press, 2000.
- Manuais e Documentação do MetroWerks CodeWarrior Lite 4.01: CodeWarrior Palm OS Tutorial For Windows, Code Warrior Targeting Palm OS, Palm OS Companion, Palm OS Reference, Constructor for Palm OS, Development Tools Guide e outros.
- Jones, Russel A. **Mastering Active Server Pages 3**, Ed Sybex 1999.
- Vários. **Palm Programming**, Sams, 1998