UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

GUILHERME GOETTEMS SCHNEIDER

Estudo prático da ferramenta BREW através do desenvolvimento de um jogo multijogador em rede

Projeto de Diplomação

Prof. Dr. Alexandre da Silva Carissimi
Orientador

Porto Alegre, dezembro de 2004

AGRADECIMENTOS

Ao meu orientador, prof. Dr. Alexandre da Silva Carissimi, pela sugestão do tema, pela sua experiência transmitida e pelo tempo dedicado.

À minha família, em especial à minha irmã, Betina Goettems Schneider, e minha mãe, Maria Elisabet Goettems, pelo apoio e compreensão durante as horas de trabalho.

À minha namorada, Lígia Gabriela Buchfink, pelo carinho e incentivo nos momentos mais difíceis.

À empresa em que trabalho, iProcess, pela aposta em seus funcionários e pelo total apoio ao estudo e ao desenvolvimento das pessoas.

À Universidade Federal do Rio Grande do Sul, pelos excelentes serviços prestados à comunidade e pela oportunidade que tive de conviver com pessoas tão distintas, como meus colegas e professores com os quais pude compartilhar conhecimento e experiências durante os últimos anos.

SUMÁRIO

LISTA DE ABREVIATURAS

.NETcf – .NET Compact Framework
AEE – Application Execution Environment
AMPS – Advanced Mobile Phone System
ANATEL – Agência Nacional de Telecomunicações
API – Application Program Interface
ARM – Advanced RISC Machines
ASIC – Application Specific Integrated Circuit
BAR – BREW Applet Resource
BCI – BREW Compressed Images
BREW – Binary Runtime Environment for Wireless
BRI – BREW Resource Intermediate
BSC – Base Station Controller
BSS – Base Station System
CCC – Central de Comutação e Controle
CDC – Connected Device Configuration
CDMA – Code Division Multiple Access
cHTML – Compact HTML
CLDC – Connected Limited Device Configuration
D-AMPS – Digital Advanced Mobile Phone System
DLL – Dynamic Link Library
EDGE – Enhanced Data rates for GSM Evolution
EM – Estação Móvel
ERB – Estação Rádio Base
EUA – Estados Unidos da América
FDM – Frequency Division Multiplexing
GCC – GNU Compiler Collection
GPRS – General Packet Radio Service
GSM – Global System for Mobile Communication
HDML – Handheld Devices Markup Language
HTML – Hypertext Markup Language
HTTP – Hypertext Transfer Protocol
IIS – Internet Information Server
IMEI – International Mobile Station Equipment Identity
IMSI – International Mobile Subscriber Identity
IP – Internet Protocol
J2EE – Java 2 Enterprise Edition
J2ME – Java 2 Micro Edition
J2SE – Java 2 Standard Edition
JPEG – Joint Photographic Experts Group

JVM – Java Virtual Machine
MIDP – Mobile Information Device Profile
MIF – Module Information File
MIME – Multipurpose Internet Mail Extensions
MMIT – Microsoft Mobile Internet Toolkit
P2P – Peer-to-peer
PASI – Provedor de Acesso a Serviços Internet
RISC – Reduced Instruction Set Computer
SDE – Smart Device Extensions
SDK – Software Development Kit
SIM – Subscriber Identity Module
SMP – Serviço Móvel Pessoal
SMS – Short Message Service
TCP – Transmission Control Protocol
TDM – Time Division Multiplexing
TDMA – Time Division Multiple Access
TIA – Telecommunications Industry Association
UDP – User Datagram Protocol
UMTS – Universal Mobile Telecommunications System
WAP – Wireless Application Protocol
W-CDMA – Wideband CDMA
WML – Wireless Markup Language
XHTML – Extended Hypertext Markup Language
XML – Extensible Markup Language

RESUMO

O mercado de telefonia celular tem crescido muito nos últimos anos em todo o mundo. O avanço da tecnologia propiciou o surgimento de novas aplicações e serviços, diferentes da tradicional comunicação pessoal por voz. Os novos aparelhos são capazes de executar programas e isto abriu um imenso leque de oportunidades de negócio para operadoras e desenvolvedores.

Neste contexto, surgiu o BREW, uma solução completa para o desenvolvimento e distribuição de aplicações para telefones celulares. O BREW foi projetado para lidar com as limitações destes dispositivos, otimizando o uso de recursos e facilitando a portabilidade das aplicações.

Este trabalho propõe-se a estudar esta tecnologia através da criação de um jogo para celular que permita a interação entre vários jogadores. O jogo deverá respeitar as limitações destes aparelhos e rodará em um ambiente de emulação.

Palavras-chave: celular, telefonia, BREW, jogos, entretenimento.

Practical Study of BREW tools through the development of a multiplayer network game

ABSTRACT

Mobile phone's market had been growing a lot in the lasts years all over the world. The advance of technology supported the appearance of new applications and services, different from the traditional personal voice communication. The newest devices are able to run programs, opening a huge world of business opportunities for operators and developers.

In this context, BREW came as a complete solution for development and distribution of applications for cell phones. BREW was designed to handle the limitations of this kind of devices, optimizing the use of resources and making the applications portable for the different models.

This work intend to study this technology through the development of a game for cell phones that allows interaction between many players. The game will respect the device's limitations and will run on an emulation environment.

Keywords: cell phone, wireless, BREW, games, entertainment.

LISTA DE FIGURAS

LISTA DE TABELAS

1 INTRODUÇÃO

O fenômeno da disseminação da telefonia móvel em todo o mundo possibilitou que as pessoas se comunicassem com mais agilidade. Ficou muito mais fácil “encontrar” alguém que possui um aparelho celular. Até poucos anos atrás, estes dispositivos eram usados especialmente para isto: comunicação pessoal por voz. O rápido avanço da tecnologia, contudo, possibilitou que novos serviços, como *download* de aplicativos, troca de e-mails com imagens e acesso à Internet, fossem disponibilizados pelas operadoras e suportados pelos aparelhos modernos.

Dispositivos com mais recursos de memória e processamento suportam aplicativos cada vez mais interessantes, como agendas, sistemas de localização, organizadores de compromissos e jogos. Os aplicativos ficaram também mais atraentes ao usuário após o surgimento de visores coloridos e com maior resolução, possibilitando o uso de imagens com mais detalhe e definição. A melhoria da qualidade sonora também tem sido uma característica explorada pelos fabricantes para incentivar as pessoas a trocar seu celular por um modelo mais novo. Com isso, a multimídia conquistou definitivamente seu lugar no mundo da telefonia móvel, onde os jogos ocupam posição de destaque.

Enfim, os aparelhos celulares de hoje são multifuncionais, verdadeiros computadores compactos de extrema mobilidade, e não têm mais a finalidade única da comunicação. Devido ao aumento no leque de possibilidades de uso destes dispositivos, a demanda pelo desenvolvimento de novos aplicativos tem aumentado bastante nos últimos anos. A variedade de modelos, entretanto, é um problema para os desenvolvedores, uma vez que os aplicativos podem se tornar obsoletos rapidamente ou se limitarem a aparelhos de um fabricante em particular.

Neste contexto, ferramentas específicas para criação de programas em celular surgiram, de modo a atenuar o problema da portabilidade e facilitar o desenvolvimento. Algumas tecnologias foram adaptadas para trabalhar com os pequenos aparelhos, como a linguagem Java, que é muito difundida na indústria do desenvolvimento de sistemas e, através da J2ME, ganhou força no mundo sem fio. Outras foram criadas, como o BREW, especificamente visando aproveitar ao máximo as características dos dispositivos móveis.

Uma das principais diferenças entre a J2ME e o BREW é que os programas Java são interpretados e executados por uma máquina virtual enquanto que os programas BREW são compilados para código nativo, executando diretamente sobre o *hardware*. Para dispositivos com capacidade de processamento limitada, a diferença de desempenho pode ser considerável. Além disso, o BREW é uma solução completa de desenvolvimento, controle e distribuição de aplicativos.

No Brasil, o número de telefones móveis superou o número de linhas fixas em agosto de 2003, segundo dados da Agência Nacional de Telecomunicações (Anatel). O acesso à Internet através de telefones celulares já é uma realidade no país, porém, ainda hoje, os principais usos dos aparelhos celulares pelos brasileiros são a comunicação

pessoal por voz e a troca de mensagens de texto. As tecnologias de celular mais utilizadas no Brasil são GSM, TDMA e CDMA. A solução BREW chegou ao mercado nacional em 2004, através da operadora Vivo, e promete uma grande revolução no uso dos telefones celulares, além da abertura de inúmeras oportunidades de negócio para pequenas empresas.

1.1 Objetivos do trabalho

O objetivo deste trabalho é fazer um estudo prático da tecnologia BREW, desenvolvida pela Qualcomm, através da implementação de um jogo, que deverá rodar num emulador de celular. O jogo precisa se adequar às características deste tipo de plataforma e explorar seus recursos através do BREW. A principal funcionalidade dos aparelhos celulares – a comunicação – deverá ser aproveitada, permitindo a interação entre vários jogadores. A idéia é avaliar as ferramentas de desenvolvimento do BREW, relatando as principais dificuldades encontradas, bem como as limitações existentes na criação de aplicações para celulares.

1.2 Organização do trabalho

Este trabalho está dividido em seis capítulos incluindo esta introdução.

O capítulo 2 começa com um breve histórico da comunicação sem fio. Em seguida, são apresentados a arquitetura básica e os componentes de um sistema de telefonia celular. Logo após, são descritas as principais tecnologias usadas no Brasil e como está organizado o sistema brasileiro de telefonia móvel. Por fim, são listadas as operadoras que atuam no estado do Rio Grande do Sul e seus principais serviços prestados.

No capítulo 3 serão estudados os mais conhecidos protocolos, linguagens e plataformas de desenvolvimento para aplicativos em celular, onde serão apontadas as principais características e tendências de evolução de cada um.

O capítulo 4 é focado na tecnologia BREW, detalhando sua arquitetura, ambiente de desenvolvimento e seu modelo de negócios.

O capítulo 5 descreve o processo de criação e desenvolvimento de um jogo utilizando a tecnologia estudada no capítulo anterior. Serão discutidas aqui as estratégias de implementação ao longo do projeto.

No capítulo 6 serão apresentadas as conclusões deste trabalho, as dificuldades encontradas e uma avaliação final da tecnologia estudada.

2 TECNOLOGIAS DE TELEFONIA CELULAR

Cada vez mais estamos vivendo num mundo sem fio. Aparelhos se comunicam sem a utilização dos mesmos, através da tecnologia *Bluetooth*. Já existem redes locais que os dispensam (padrão 802.11) e os telefones ganharam mobilidade a partir do surgimento dos sistemas de telefonia celular. Não demorou muito para que estes sistemas fossem usados para transmissão de informações diversas, além da voz. Isso, aliado à evolução dos próprios aparelhos, propiciou o surgimento de muitos serviços usando esta infraestrutura, como o acesso à Internet, *download* de aplicativos e jogos em rede.

Neste capítulo serão descritas as principais tecnologias de telefonia celular e a evolução desta área no sentido de melhor suporar as transimssões de dados.

2.1 Histórico

A idéia da comunicação sem fio é bastante antiga. Pode-se dizer que sua origem foi a invenção do rádio em 1880 por Nikolai Tesla. O telefone havia surgido quatro anos antes, inventado por Alexander Graham Bell.

Da combinação das duas tecnologias surgiu o primeiro sistema de telefonia móvel, implantado experimentalmente em 1946 pela Bell Telphone Company, chamado de MTS (*Mobile Telephone Service*). No Brasil, a telefonia móvel foi introduzida em 1972, na cidade de Brasília.

Comercialmente, o primeiro sistema surgiu em 1979 em Tóquio, no Japão, pela NTT (Nippon Telephone & Telegraph). Nos EUA, o primeiro sistema comercial entrou em operação em 13 de outubro de 1983, em Chicago, e utilizava a tecnologia *Advanced Mobile Phone System* (AMPS). No Brasil, a implantação de sistemas de celular iniciou-se em 1990.

Na década de noventa começaram a ser utilizados sistemas de telefonia celular digitais. Neste período ocorreu a popularização dos telefones celulares que eram basicamente para uso da comunicação por voz. A evolução, tanto dos aparelhos quanto das tecnologias de comunicação, fez surgir novas aplicações para estes dispositivos. Vários sistemas diferentes surgiram, e a tendência atual e futura é que elas se integrem, formando um único sistema global de comunicação sem fio.

2.2 Arquitetura do sistema de telefonia celular

Um sistema de telefonia celular possui três componentes básicos: o aparelho celular, também chamado de estação móvel (EM), as estações rádio base (ERBs) e as centrais de comutação e controle (CCCs). A área de abrangência do sistema, também chamada de área de cobertura, é dividida em regiões chamadas células, cada uma com uma ERB. Esta possui antenas que fazem a comunicação sem fio com as EMs que estão dentro de sua célula.

Comunicações sem fio são feitas através de ondas eletromagnéticas e utilizam uma certa faixa de freqüência. As faixas de freqüência são regulamentadas pelo governo para garantir que não haja sobreposição. Cada tecnologia de celular especifica a faixa no espectro de freqüência eletromagnética que utiliza. Este meio deve ser compartilhado entre as diversas EMs que estão numa mesma célula em um determinado instante. Um canal de comunicação é uma divisão lógica deste meio para uso de uma EM. Para que não haja sobreposição, células vizinhas utilizam conjuntos distintos de canais. Um exemplo, onde os canais foram divididos em sete grupos, é ilustrado pela Figura 2.1. Cada letra representa um grupo diferente de canais.

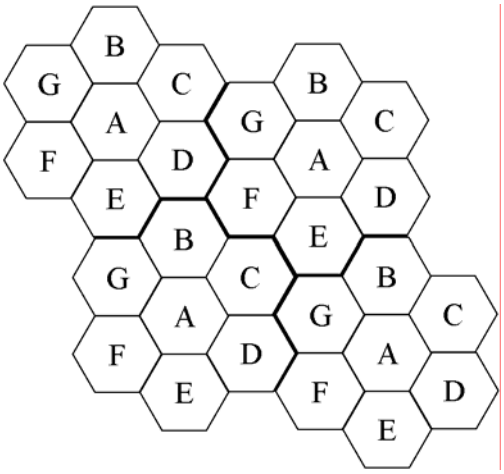

Figura 2.1: Células e canais de freqüência

Uma EM, estando em deslocamento, pode sair da sua célula atual. Neste caso, a ERB que abrange a próxima célula deve assumir a comunicação com ela. Este processo é chamado de *handoff* e deve ocorrer sem que haja interrupção na transmissão, sendo totalmente transparente para o usuário. Devido ao fato de que células vizinhas possuem canais diferentes, este processo inclui uma troca de canal, gerenciado pela CCC, e leva cerca de 300 ms (TANENBAUM, 2003).

Um conjunto de ERBs se liga a uma CCC geralmente através de cabos. A CCC controla quais EMs estão associadas a cada ERB, gerencia o *handoff* e comunica-se com o restante do sistema telefônico, através de ligações à central de telefonia fixa e a outras CCC. Esta aquitetura é ilustrada pela Figura 2.2.

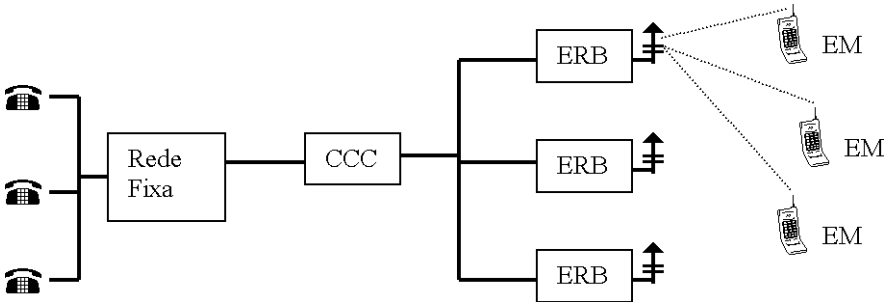

Figura 2.2: Arquitetura simplificada de um sistema de telefonia móvel

Em algumas arquiteturas de sistemas de telefonia móvel pode existir um nível a mais nesta hierarquia. Neste caso, as ERBs não se ligam diretamente às CCCs e sim a uma *Base Station Controller* (BSC) que controla várias ERBs, contemplando uma região denominada de *Base Station System* (BSS). Várias BSCs, então, se ligam a uma CCC.

A rede de telefonia celular se integra com a rede de telefonia fixa através da ligação das CCCs às centrais de telefonia fixa. A rede de telefonia fixa é o principal meio de

acesso à Internet no Brasil, onde usuários conectam-se através de uma chamada local a um provedor de acesso a serviços Internet (PASI). O acesso discado, cuja taxa de transferência é limitada a 56 kbit/s, é tarifado pela companhia telefônica da mesma maneira que uma ligação comum. O PASI pode cobrar uma taxa fixa mensal, cobrar por horas conectadas ou até mesmo oferecer seus serviços gratuitamente.

2.3 Tecnologias de comunicação

As tecnologias de comunicação definem tanto as faixas de freqüência que utilizam como o modo de codificação das informações. Para que fluxos independentes de informação possam existir é preciso que o meio de comunicação seja compartilhado de maneira adequada. As duas formas mais comuns de se fazer isso são a multiplexação por divisão de freqüência, *Frequency Division Multiplexing* (FDM) e a multiplexação por divisão de tempo, *Time Division Multiplexing* (TDM), mostradas na Figura 2.3 (STALLINGS, 2000). Na FDM os sinais são transmitidos simultaneamente, porém em faixas de freqüências diferentes, enquanto que na TDM cada fluxo de comunicação pode utilizar todo espectro de freqüência, mas são intercalados no tempo.

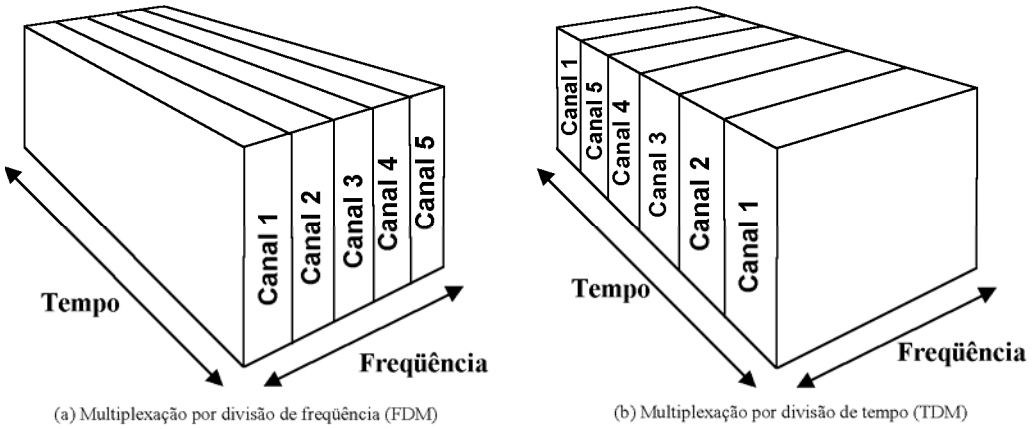

Figura 2.3: FDM e TDM

Quanto à maneira de estabelecer e manter uma comunicação, existem duas técnicas utilizadas pelas tecnologias de comunicação: comutação de circuitos e comutação de pacotes. Na comutação de circuitos um canal de comunicação é alocado e fica dedicado a uma única conexão até que ela encerre. Se, por um lado, isto garante disponibilidade contínua e ininterrupta, por outro, um recurso reservado é um desperdício quando há pouca ou nenhuma informação sendo transmitida. Já na comutação de pacotes, as informações são divididas em blocos de tamanho fixo e são enviadas sob demanda, evitando desperdícios, porém estando sujeito a congestionamentos repentinos e a possíveis atrasos no recebimento dos pacotes. Em geral, o modelo de cobrança no uso de uma conexão por comutação de circuito é pelo tempo de utilização, enquanto que na comutação de pacotes cobra-se pelo volume de informações transmitido.

Ao longo do tempo, as tecnologias de comunicação sem fio para telefonia celular utilizaram diferentes técnicas de multiplexação e comutação e foram classificadas em gerações.

A primeira geração (1G) é caracterizada por tecnologias analógicas, como o *Advanced Mobile Phone System* (AMPS) que já foi o sistema de telefonia celular mais difundido nos EUA. Nesta geração, praticamente com o propósito exclusivo da comunicação por voz, as redes estabelecem conexões através de comutação por circuito e a divisão de canais é feita por FDM. As tecnologias 1G já suportavam o *handoff* e o

roaming, que é o uso do celular em área controlada por uma operadora diferente da que foi habilitado.

As tecnologias digitais atualmente usadas, como TDMA (IS-136), CDMA (IS-95A), e GSM são a segunda geração (2G). Essas tecnologias ainda fazem comutação de circuitos para transmissão de voz, porém caracterizam-se pelo uso da técnica de comutação de pacotes para transmissão de dados. Isto faz com que as operadoras possam cobrar pelo volume de informação transmitido e não pelo tempo de conexão para a transmissão de dados. Comercialmente lançou-se a idéia da geração dois e meio (2,5G), depois que as primeiras tecnologias digitais sofreram melhorias, aumentando suas capacidades de transmissão.

Tem-se a expectativa de que, cada vez mais, as redes de telefonia celular sejam utilizadas para transmissão de dados (TANENBAUM, 2003). Tecnologias de terceira geração (3G) ainda estão sendo desenvolvidas e visam unificar os sistemas de comunicação em nível global, além de permitir maiores velocidades de transmissão, possibilitando uso de multimídia nos dispositivos móveis. Nessa geração a técnica de comutação de pacotes é utilizada inclusive na comunicação por voz.

2.3.1 AMPS – Advanced Mobile Phone System

Foi o mais antigo e mais difundido sistema celular analógico nos EUA. Opera nas freqüências de 824-849 MHz para transmissão e de 869-894 MHz para recepção. A banda é dividida em canais, que ocupam 30 kHz em cada sentido da comunicação (transmissão e recepção), tornando possível a existência de 832 canais. Destes, 790 são designadas para transmissão de voz e 42 para dados, que basicamente são os sinais de controle. Os canais são divididos em duas bandas (A e B), para que possam ser explorados comercialmente por mais de uma empresa na mesma região.

As células no AMPS têm diâmetros que medem em geral de 10 a 20 km e, na prática, utilizam em torno de 45 canais cada uma (TANENBAUM, 2003).

Apesar de analógico, esse sistema também pode ser utilizado para transmissão de dados, porém com capacidade normalmente limitada a taxas de transferências de 10 kbit/s, muito menor do que em canais discados convencionais, operando freqüentemente a apenas a 1 kbit/s, devido ao ruído. O AMPS foi gradativamente sendo substituído pelas tecnologias digitais (2G).

2.3.2 TDMA – Time Division Multiple Access

O TDMA (IS-136) foi um dos primeiros sistemas digitais (2G) e surgiu nos EUA. Este sistema tem a mesma arquitetura do AMPS e utiliza a mesma faixa de freqüência, porém pode utilizar o mesmo canal para mais de uma ligação, através da multiplexação por tempo (TDM), além da FDM. Deste modo, o TDMA, que também é conhecido como *Digital Advanced Mobile Phone System* (D-AMPS), suporta três vezes mais chamadas que o AMPS. O consumo de energia do aparelho é reduzido, pois as células são menores, diminuindo a potência necessária para a comunicação, impactando numa maior duração da bateria.

O canal de controle do TDMA é digital, o que possibilitou a criação do *Short Message Service* (SMS), um serviço de troca de mensagens curtas muito difundido atualmente no Brasil. Outros serviços, como identificação do número chamador, chamada em espera e conferência, também foram introduzidos. A taxa de transmissão de voz é de 13 kbit/s. Aparelhos móveis que utilizam o sistema TDMA geralmente também operam em AMPS, o que facilita o *roaming* entre esses dois sistemas.

2.3.3 CDMA – Code Division Multiple Access

A tecnologia CDMA foi desenvolvida nos EUA pela empresa Qualcomm e caracteriza-se pelo uso da tecnologia de espalhamento espectral, permitindo com que várias chamadas compartilhem as mesmas faixas de freqüência (portanto, sem FDM) simultaneamente (portanto, sem TDM). As diversas ligações são identificadas pelo uso de códigos diferentes.

O primeiro sistema com tecnologia CDMA foi o TIA IS-95, publicado em 1993. Este padrão sofreu duas revisões: IS-95A (1995) e IS-95B (1999). A família IS-95 também é conhecida como cdmaOne. A tecnologia CDMA IS-95A é muito difundida comercialmente como uma das principais tecnologias 2G. Nesta revisão são possíveis canais de comunicação de dados de até 14,4 kbit/s. Já a revisão IS-95B, que integrou outras normas existentes, possibilita taxas de transmissão de dados de até 115 kbit/s. Pela sua maior capacidade de transmissão, esta é uma tecnologia considerada 2,5G.

O cdmaOne opera nas mesmas faixas de freqüência que o AMPS e o TDMA, estando dividido em faixas maiores, de 1,25 MHz, mas que podem conter até 60 ligações simultâneas, diferenciadas entre si pelo código identificador. O CDMA utiliza um codificador de voz que permite transmissão a taxas variadas, de 1,8 kbit/s até 14,4 kbit/s, otimizando o meio de comunicação. Como resultado, essa tecnologia é capaz de suportar de 8 a 10 vezes mais ligações que o AMPS.

De acordo com o *CDMA Developers Group* (CDG, 2004) a primeira tecnologia 3G usada comercialmente foi o CDMA2000 1xRTT, em outubro de 2000 na Coréia. Esse sistema duplica a capacidade de transmissão de voz e eleva para até 307 kbit/s a taxa de recepção de dados, permitindo assim o uso de aplicações multimídia. No entanto, muitos consideram esta uma tecnologia 2,5G. A tecnologia subseqüente, CDMA2000 1xEV, que alcança taxas de até 2,4 Mbit/s, é de fato 3G. O CDMA2000 utiliza largura de banda de 5 MHz.

Existe ainda o *Wideband CDMA* (W-CDMA), proposto pela empresa Ericsson e também utiliza 5 MHz de banda. O W-CDMA pode fazer *handoff* com células GSM, o que não ocorre com o CDMA2000. O W-CDMA é uma tecnologia 3G e foi adotado na Europa, sendo chamado de *Universal Mobile Telecommunications System* (UMTS).

2.3.4 GSM – Global System for Mobile Communication

A necessidade de um sistema que facilitasse o *roaming* internacional inspirou a criação do padrão GSM, publicado em 1990 pelo *European Telecommunications Standards Institute* (ETSI), sendo incompatível com os sistemas analógicos da época. A idéia já vinha sendo desenvolvida desde o início da década de 80 na Europa. O GSM, caracterizado por sua arquitetura aberta, é o sistema atualmente mais utilizado no mundo.

Originalmente o GSM operava nas freqüências de 900 MHz (GSM 900), mas foi adaptado para operar também nas freqüências 1800 MHz (DCS 1800) e 1900 MHz (PCS 1900), conforme Tabela 2.1. O GSM utiliza FDM com 200 kHz de banda por canal e também faz TDM, dividindo um canal em 8 *slots* de tempo.

No Brasil, usa-se o sistema DCS 1800, cujo espectro de freqüência é dividido nas bandas C, D e E.

Tabela 2.1: Freqüências de operação do GSM

Sistema	**GSM 900**	**DCS 1800**	**PCS 1900**
Transmissão	880-915 MHz	1710-1785 MHz	1850-1910 MHz
Recepção	925-960 MHz	1805-1880 MHz	1930-1990 MHz

Os celulares com tecnologia GSM utilizam um cartão que identifica o assinante, denominado *Subscriber Identity Module* (SIM). Isto permite que um usuário troque de aparelho e continue com o seu número e suas informações. Cada SIM *Card* possui um número de 15 dígitos chamado de *International Mobile Subscriber Identity* (IMSI) que identifica usuários em todo o mundo. Da mesma forma, os aparelhos são também identificados por um número de 15 dígitos, o *International Mobile Station Equipment Identity* (IMEI).

As CCCs do GSM contêm um banco de dados com os registros dos assinantes, visitantes (em *roaming*) e aparelhos do sistema celular. As autenticações dos usuários são feitas pela CCC, assim como são geradas as chaves para conexões criptografadas, quanto utilizadas.

Os sistemas de localização padronizados para o GSM conseguem estimativas precisas sobre a localização dos aparelhos móveis. Isso possibilita a criação de diversas aplicações baseadas neste serviço, como mapas e busca por estabelecimentos próximos.

A rede original do GSM foi otimizada para a transmissão de voz. Com o objetivo de aperfeiçoar o uso da rede para transmissão de dados, surgiu o *General Packet Radio Service* (GPRS) que viabilizou uma série de novos serviços, como envio e recebimento de fotos e navegação na Internet.

O GPRS é um serviço de comunicação de dados que permite ao usuário utilizar a Internet a qualquer momento, sem a necessidade de fazer uma ligação que reserve banda para a comunicação. O GPRS utiliza comutação de pacotes, que otimiza o uso dos recursos da rede, pois é utilizada apenas quando há dados a serem transmitidos. As taxas de transmissão podem chegar até a 115 kbit/s, quando são utilizadas as 8 fatias de tempo para a mesma transmissão. Porém, na prática, essas taxas costumam ser bem menores (BUCKINGHAM, 2004).

O GPRS evoluiu para a tecnologia *Enhanced Data rates for GSM Evolution* (EDGE), visando aumentar as taxas de transmissão. O EDGE usa uma nova modulação variando a amplitude das ondas transmitidas. Essa modulação aumenta a eficiência espectral, possibilitando taxas três vezes maiores. Além disso, o EDGE é compatível com o GPRS, o que facilita a migração entre essas tecnologias.

2.4 Modelo de negócio

Existem dois modos de contratação dos serviços das operadoras à disposição dos clientes: pós-pago e pré-pago. No plano pós-pago, o usuário recebe a conta correspondente ao uso dos serviços do mês anterior. Geralmente, as operadoras oferecem diferentes planos, cada um com um certo valor mínimo a ser cobrado por mês que dá direito a uma determinada quantidade de serviço (minutos de ligações, número de mensagens de texto enviadas, etc.). Se o usuário utiliza mais do que essa quantidade, ele paga a diferença, que é acrescida na sua conta. A outra modalidade são os planos pré-pagos, preferidos por quem usa pouco seu celular, ou o utiliza mais para receber chamadas. Neste caso, o usuário compra cartões que possuem um código a ser inserido no aparelho dando direito a créditos. Esses créditos vão sendo descontados à medida que o cliente utiliza os serviços da operadora. Geralmente os créditos possuem um período de validade, obrigando o usuário a inserir novos créditos de tempos em tempos.

Atualmente as operadoras atuam em três linhas básicas de negócio: comunicação pessoal, serviços oferecidos pelas próprias operadoras, como *downloads*, SMS, etc. e acesso à Internet, através da qual clientes têm a sua disposição serviços prestados por terceiros.

A principal linha de negócio ainda é o da comunicação por voz. Entretanto, novos serviços não param de surgir. Outras maneiras de comunicação incluem, por exemplo, o envio de mensagens, correio eletrônico e bate-papo interativo. Novas tecnologias permitem que dispositivos móveis possam determinar sua localização geográfica com razoável precisão, tornando possível o surgimento de aplicações de navegação, trânsito, mapas e até de jogos no estilo caça-ao-tesouro. Outros tipos de aplicações que têm despertado muito interesse são as que integram sistemas corporativos visando aumentar a produtividade das empresas e ainda as que exploram comércio por celulares, ou *m-commerce* (*mobile comerce*). Aplicações de entretenimento, como tons musicais, troca de fotografias, vídeo e jogos têm se destacado entre os usuários mais jovens.

2.5 Operadoras no Rio Grande do Sul

A Agência Nacional de Telecomunicações (Anatel) é o órgão que regulamenta os serviços de telecomunicações no Brasil. Quando surgiram os primeiros sistemas celulares, a Anatel dividiu o espectro de freqüência utilizado pelas tecnologias nas bandas A e B, com o objetivo de permitir que mais de uma operadora de telefonia celular pudesse atuar numa mesma região.

Em 2001, foi criado um novo modelo de prestação de serviço celular, chamado Serviço Móvel Pessoal (SMP), que divide o Brasil em três regiões (I, II e III). Foram também acrescidas novas faixas de freqüência, alocadas para as bandas C, D e E.

A Tabela 2.2 mostra as operadoras com suas tecnologias que atuam no estado do Rio Grande do Sul, que faz parte da região II do SMP.

Tabela 2.2: Operadoras de SMP que atuam no Rio Grande do Sul

Operadoras Autorizadas SMP	**Banda**	**Tecnologias**
Celular CRT S.A. (Vivo)*	A	AMPS, TDMA e CDMA
Tim Sul S.A. (Tim)**	A	AMPS, TDMA e GSM
Telet S.A. (Claro)	B	TDMA e GSM
Tim Celular S. A. (Tim)	D	GSM
Brasil Telecom Celular S. A. (BrT Celular)	E	GSM

* Atua em todo RS com exceção da região de Pelotas.

** No RS atua apenas na região de Pelotas.

Além do serviço de telefonia, as operadoras do Rio Grande do Sul oferecem dezenas de outros serviços. O mais conhecido é a troca de mensagens de texto (SMS), também chamadas de torpedos, que podem ser enviadas tanto a partir de um aparelho celular como através dos *sites* das operadoras, que possuem acordos para permitir a troca de mensagens entre clientes de operadoras distintas.

Serviços que são comuns a todas as operadoras são: troca de mensagens com imagens, *download* de músicas e tons de chamada, notícias, secretária, conexão com a Internet, entre outros.

A Vivo, através da parceira com a Qualcomm, comercializa aplicativos BREW que custam de R$1,00 a R$10,00, com diversas modalidades de aquisição: assinatura mensal, compra ilimitada, por número de usos ou número de dias (VIVO, 2004). Tim, Claro e Brasil Telecom oferecem jogos em Java a preços que variam entre R$4,00 e R$10,00 (TIM, 2004; CLARO, 2004; BRASIL, 2004). Em todos os casos há também tarifação pelo *download* dos aplicativos, que pode ser por tempo de conexão ou quantidade de dados transferidos, dependendo da tecnologia que o aparelho utiliza.

2.6 Resumo

Este capítulo apresentou uma introdução aos sistemas de telefonia celular, através de um breve histórico e da apresentação de sua arquitetura básica. As principais tecnologias de comunicação foram caracterizadas, retratando a constante evolução das mesmas. Foi visto também como as operadoras atuam neste mercado e quais os serviços disponíveis no Brasil.

É importante destacar que a evolução tanto das tecnologias de comunicação quanto dos próprios aparelhos, que possuem cada vez mais poder de processamento e memória, acabou fazendo com que os celulares fossem usados para outras finalidades, além da comunicação pessoal por voz, como troca de mensagens de texto, agendas, acesso à Internet, jogos e música.

3 PROTOCOLOS, LINGUAGENS E PLATAFORMAS DE PROGRAMAÇÃO SEM FIO

O grande crescimento do mercado de telefonia móvel levou a criação de muitos aparelhos e tecnologias de comunicação diferentes por parte dos fabricantes. Celulares com mais poder de processamento e memória capacitaram a oferta de serviços além da comunicação pessoal. Com isto, surgiu a demanda pela criação de aplicativos para esta plataforma. Para que haja interoperabilidade dos serviços, ou seja, para que os diversos modelos de dispositivos possam trocar informações uns com os outros, se faz necessário a definição de protocolos padronizados.

Ferramentas surgiram pela necessidade de facilitar e agilizar o desenvolvimento desses aplicativos, o que é essencial num mercado em crescimento e muito competitivo. O uso de linguagens de programação de alto nível, juntamente com bibliotecas padronizadas, aumenta a portabilidade das aplicações.

Protocolos de acesso à Internet precisaram ser adaptados às necessidades e restrições dos pequenos aparelhos para que estes pudessem acessar informações disponibilizadas pela rede mundial.

3.1 Protocolo de acesso à Internet

Em 1997 grandes fabricantes de aparelhos de comunicação sem fio, com o objetivo de padronizar o acesso dos dispositivos móveis à Internet, desenvolveram o *Wireless Application Protocol* (WAP), que abrange um conjunto de tecnologias e protocolos. O WAP é utilizado na comunicação entre o aparelho e um *gateway* que está conectado à rede e faz a conversão do WAP para o *Hypertext Transfer Protocol* (HTTP), que é protocolo padrão para transferência de documentos na Internet, como mostra a Figura 3.1.

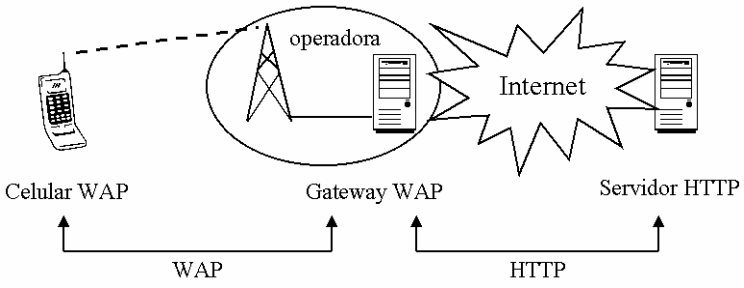

Figura 3.1: Arquitetura de funcionamento WAP

O WAP foi criado com base nas características dos dispositivos móveis, tais como visor reduzido, baixo poder de processamento, poucas teclas e principalmente baixas taxas de transmissão de informação (FOO; LEE; WATSON; WUGOFSKI, 2000).

Dispositivos baseados em WAP são incapazes de tratar transmissão de multimídia, pois operam a apenas 9600 bit/s. Além disso, o WAP utiliza um sistema de comutação de circuitos, fazendo com que os usuários paguem por tempo de conexão. Por fim, esta tecnologia não utiliza a *Hypertext Markup Language* (HTML), linguagem de marcação padrão de documentos na Internet, fazendo com que dispositivos WAP acessem somente páginas convertidas para *Wireless Markup Language* (WML). Por estas razões, o WAP, em sua primeira versão, foi um fracasso (TANENBAUM, 2003).

Em 2001 surgiu a segunda versão do protocolo, chamada de WAP 2.0. Esse protocolo prevê o uso das maiores capacidades dos novos sistemas, suportando inclusive aplicações multimídia, e faz comutação de pacotes. Além disso, o WAP 2.0 aceita protocolos padrão da Internet como *Transmission Control Protocol* (TCP) e HTTP e a linguagem de marcação *Extended Hypertext Markup Language* (XHTML), que é a última versão da HTML.

Em 1999, surgiu no Japão, através da empresa NTT DoCoMo, um serviço chamado i-mode (*information-mode*). Mais do que um protocolo de acesso à Internet, o i-mode possui um novo sistema de transmissão que faz comutação de pacote para dados a 9600 bit/s. O i-mode suporta o uso da linguagem Java e utiliza a *compact HTML* (cHTML) para exibição de páginas *web*. Os usuários desse sistema têm acesso a centenas de serviços oficiais (controlados pela NTT DoCoMo) e a milhares de *sites* na Internet. Com isso, o i-mode fez um grande sucesso no Japão.

3.2 Linguagens de programação

Existem dois tipos de linguagens bem distintas: para criação de documentos, pelo uso de marcas que definem sua formatação; e para criação aplicativos, pelo uso instruções que manipulam dados e acessam recursos do aparelho.

Sistemas *web* baseados em requisições, recuperação de informações e navegação através de *links* são feitos com documentos. Documentos são criados por linguagens de marcação, como a HTML, que indicam a formatação do documento, localização de figuras e *links*. Navegadores interpretam essas linguagens e montam a visualização do documento. Pode-se adicionar interatividade através da criação de *scripts*, como o JavaScript, que são interpretados e executados no próprio dispositivo, atuando sobre o documento.

Já as linguagens para criação de aplicativos, como C e Java, são bem mais genéricas, permitindo maior controle sobre as funcionalidades e recursos dos dispositivos. Os programas tanto podem ser compilados para uma linguagem intermediária e depois serem interpretados por uma máquina virtual como podem ser compilados diretamente para o código nativo do processador do aparelho.

3.2.1 Linguagens de criação de documento

As linguagens HTML e JavaScript são as linguagens mais utilizadas para criação de documentos disponibilizados na *web*. Estas linguagens, no entanto, foram criadas para visualização de documentos em computadores pessoais, e não são adequadas para dispositivos que possuem restrições como telas reduzidas, a não existência *mouse* e o alto custo para utilização da rede. Surgiram, então, linguagens específicas para este tipo de aparelho.

Uma das primeiras linguagens de marcação para dispositivos de mão foi a *Handheld Devices Markup Language* (HDML), desenvolvida em 1996 pela empresa Openwave que na época chamava-se Unwired Planet. Semelhante à HTML, as marcas servem para definir a formatação do documento, que deve ser visualizado nos pequenos aparelhos

através dos micronavegadores. A HDML foi largamente utilizada comercialmente e acabou evoluindo para a *Wireless Markup Language* (WML).

A principal diferença da WML em relação à HDML é o fato ser baseada no padrão *Extensible Markup Language* (XML). Um documento WML forma um *deck* que pode conter vários *cards*, que são as páginas exibidas pelo aparelho móvel, por onde o usuário pode navegar. Um *deck* é carregado inteiramente na memória do dispositivo, facilitando assim a navegação entre *cards* do mesmo *deck*.

Documentos WML podem conter *scripts*, escritos na linguagem WMLScript, que são interpretados e executados pelo navegador do dispositivo sem fio. O *script* é armazenado em um arquivo separado ao do documento WML, que por sua vez pode invocar as funções WMLScript. Estes *scripts* permitem uma maior interatividade do usuário com os documentos.

O sistema i-mode utiliza a linguagem *Compact Hypertext Markup Language* (cHTML) para criação de documentos. Como o nome sugere, é baseada na HTML, porém com várias limitações como a falta de suporte a imagens no padrão *Joint Photographic Experts Group* (JPEG), a inexistência de *scripts* e ausência de estilos de página.

3.2.2 Linguagens de criação de aplicativos

A linguagem C tornou-se muito popular por ser uma linguagem de uso genérico e produzir programas eficientes. Com o tempo foram surgindo variações, e a linguagem evoluiu para o C++, criado por Bjarne Stroustrup, passando a suportar programação orientada a objetos. A linguagem C++ foi construída de maneira a manter a maior compatibilidade possível com C.

Hoje, o C++ é muito difundido e é usado em aplicações diversas. Existe uma biblioteca padrão com classes para armazenamento e manipulação de dados eficiente. A linguagem permite que se programe tanto em baixo nível (estilo C), manipulando ponteiros e alocação de memória, como alto nível, através do uso de classes e abstrações.

A Microsoft criou a linguagem C# (C *sharp*), baseada em C++ e em algumas características da linguagem Java. Esta linguagem, no entanto, só pode ser utilizada nos ambientes da própria Microsoft.

A linguagem Java, que permite altos níveis de abstração, através de seu paradigma orientado a objetos, foi desenvolvida pela Sun Microsystems e tem como principal característica sua portabilidade. Tipos de dados e *bytecodes* (instruções Java) foram definidos de maneira genérica, sendo independentes de máquina ou sistema operacional. Para cada sistema existe uma Máquina Virtual Java ou *Java Virtual Machine* (JVM) que interpreta e executa os *bytecodes* e faz o gerenciamento dos recursos e objetos do programa.

O Java também é muito conhecido por seus mecanismos de segurança, permitindo inclusive que seus programas sejam carregados via rede de origens não confiáveis, podendo executar sem comprometer o sistema do usuário.

3.3 Plataformas de desenvolvimento

A imensa corrida por novas aplicações para dispositivos de mão fez surgir a necessidade da existência de ambientes de desenvolvimento específicos. Ferramentas facilitam muito a construção desses aplicativos automatizando várias tarefas. Elas ajudam o programador a manter o controle sobre o que está sendo feito e facilitam a depuração e o teste, inclusive com programas de emulação de dispositivos. A existência

de bibliotecas de rotinas comuns, propicia inúmeras vantagens no desenvolvimento, como reusabilidade e redução de erros. Deseja-se também que as aplicações possam rodar em diversos modelos de dispositivos a fim de garantir a viabilidade do negócio de produção das mesmas.

Desta forma essas ferramentas acabam por agilizar o processo de desenvolvimento, reduzindo custos e aumentando a qualidade final dos aplicativos. Além disso, os aplicativos tendem a ficar com interfaces parecidas quando utilizam as mesmas bibliotecas de desenvolvimento, o que facilita o aprendizado dos usuários quanto ao seu manuseio.

Existem duas fortes plataformas de desenvolvimento utilizadas na criação de aplicativos para celulares: J2ME, que utiliza a linguagem Java e BREW, que nativamente suporta C e C++.

3.3.1 J2ME – Java 2 Micro Edition

A linguagem Java tornou-se muito popular e foi sendo continuamente aperfeiçoada até o surgimento do Java 2, desenvolvido em três plataformas: *Java 2 Standard Edition* (J2SE), para programas e aplicativos independentes do lado do cliente; *Java 2 Enterprise Edition* (J2EE), para sistemas empresariais e de negócios; e *Java 2 Micro Edition* (J2ME), para aplicativos em dispositivos com recursos limitados.

A J2ME é um subconjunto da J2SE, de onde alguns recursos não aplicáveis a dispositivos com menor capacidade foram removidos. O ambiente de execução J2ME possui duas configurações padrão diferentes: a *Connected Device Configuration* (CDC) e a *Connected Limited Device Configuration* (CLDC). A CLDC é específica para dispositivos com alta limitação de potência e memória e alto custo de utilização da rede, como os telefones celulares. O CDC, por outro lado, é utilizado em dispositivos que não possuem tantas restrições. Acima dessas configurações existem os perfis que definem a funcionalidade numa categoria específica de dispositivo. No caso dos pequenos dispositivos móveis esse perfil é o *Mobile Information Device Profile* (MIDP).

Diversos aparelhos foram lançados com suporte ao MIDP. Porém, este perfil possui diversos inconvenientes. O HTTP, por exemplo, é o único protocolo para comunicação de dados requerido pelo padrão. Além disso, funções específicas de cada dispositivo não poderiam ser acessadas por aplicações Java. Isto fez com que cada fabricante acrescentasse suas próprias extensões ao MIDP, tornando-os incompatíveis (BENDAS, MYLLYAHO, 2002). O MIDP 2.0, no entanto, veio para solucionar esses problemas.

O serviço i-mode possui uma extensão chamada i-appli que permite o *download* e execução de aplicativos J2ME. Este ambiente define um perfil, similar ao MIDP, baseado em características específicas dos dispositivos i-mode como interfaces de usuário e uso do protocolo HTTP.

3.3.2 Microsoft Visual Studio .NET

O Visual Studio .NET é um conjunto de ferramentas para desenvolvimento de aplicações para computadores pessoais, serviços *web* e aplicações móveis. Possui um ambiente integrado de desenvolvimento ou *integrated development environment* (IDE), permitindo o compartilhamento de ferramentas entre diferentes linguagens de programação.

Existe uma extensão do Visual Studio .NET, chamada *Microsoft Mobile Internet Toolkit* (MMIT) que permite a criação de sistemas *web* acessíveis por dispositivos móveis, como celulares e computadores de mão. O MMIT é distribuído gratuitamente e roda sobre o *Internet Information Server* (IIS), que é o servidor de aplicações *web* da Microsoft. Isto quer dizer que os usuários acessam estas aplicações exclusivamente

através de uma conexão de rede, necessitando apenas de um navegador. O MMIT identifica as características do dispositivo e pode gerar conteúdo em formato HTML, cHTML ou WML.

Para o desenvolvimento de aplicativos que rodam no dispositivo, a Microsoft lançou o *.NET Compact Framework* (.NETcf) e o *Smart Device Extensions* (SDE). Estas extensões do Visual Studio .NET permitem ao programador utilizar as linguagens de programação C# e Visual Basic .NET e um subconjunto da biblioteca de objetos da solução .NET na criação de aplicativos para dispositivos móveis. Para executar tais aplicativos, os aparelhos precisam ter instalado o ambiente de execução do .NETcf.

3.3.3 BREW – Binary Runtime Environment for Wireless

A tecnologia BREW foi desenvolvida pela empresa Qualcomm dos EUA e foi lançada em 2001. Essa tecnologia abrange muitos aspectos relativos a aplicativos para dispositivos móveis, que vão desde a fase de desenvolvimento e testes até a distribuição e ambiente de execução.

Através do BREW, desenvolvedores criam aplicativos que compartilham as rotinas oferecidas pelo *Application Execution Environment* (AEE). Várias ferramentas, como editor de recursos, conversão de imagens e emuladores são oferecidas para os desenvolvedores, sendo que a plataforma de programação utilizada é o Visual Studio 6 ou .NET da Microsoft e a linguagem nativa é o C++.

A principal característica dos aplicativos BREW é que, com a padronização do acesso às funcionalidades básicas dos aparelhos, através das bibliotecas do BREW, o mesmo código fonte pode ser compilado para cada modelo de dispositivo. Assim, os programas rodam sem a necessidade de uma máquina virtual, tornando sua execução mais rápida. Porém, apenas aparelhos que suportam o BREW podem utilizá-lo.

Cada aplicativo BREW deve passar por testes para ser registrado, recebendo uma identificação única. Somente assim pode ser disponibilizado pelas operadoras a seus usuários.

3.4 Resumo

Este capítulo abordou tecnologias que permitem o acesso dos aparelhos sem fio à Internet e algumas das linguagens que podem ser utilizadas na criação de aplicações para estes dispositivos. Por fim, foram caracterizadas as principais plataformas de desenvolvimento.

A padronização do protocolo de acesso à Internet é um processo necessário e que vem, aos poucos, acontecendo. Da mesma forma, os aparelhos tendem a convergir para algumas poucas tecnologias e a adotar os padrões mais difundidos. A partir daí, aplicações podem ser desenvolvidas genericamente e funcionar em qualquer modelo de dispositivo. Atualmente ainda existem várias alternativas, com diferentes características. Algumas ainda podem evoluir e outras desaparecer.

4 BINARY RUNTIME ENVIRONMENT FOR WIRELESS (BREW)

A constante evolução da tecnologia e a grande expansão do mercado de telefonia móvel fizeram surgir muitos fabricantes e vários modelos de telefones celulares vêm sendo lançados. A demanda por novas aplicações, além das tradicionais, impulsiona a indústria de desenvolvimento na área da comunicação sem fio. Porém, os dispositivos móveis, além de terem capacidades limitadas, caracterizam-se por possuírem recursos e interfaces proprietárias, dificultando a criação de aplicativos genéricos.

Nesse contexto surgiu a solução BREW que tem como missão oferecer um ambiente de execução de aplicativos, *Application Execution Environment* (AEE), eficiente para o qual aplicativos desenvolvidos possam ser portados sem dificuldades para a maioria dos dispositivos móveis. O AEE do BREW é baseado em eventos e implementa uma biblioteca de classes que gerencia recursos e serviços do aparelho. Essas classes são acessadas através de uma *Application Program Interface* (API). Assim, programas que utilizam essas APIs podem rodar em qualquer dispositivo que suporte o BREW.

A Qualcomm, empresa criadora do BREW, foi fundada em 1985 e atua no mercado da comunicação sem fio pesquisando e desenvolvendo serviços e produtos. A empresa introduziu a tecnologia CDMA em 1989, que acabou sendo base do padrão IS-95 adotado pela *Telecommunications Industry Association* (TIA) em 1993. O sistema foi lançado comercialmente em 1995 e hoje é uma das principais tecnologias de comunicação sem fio. Em 2001, a Qualcomm lançou o BREW.

Atualmente os desenvolvedores têm a disposição cinco versões do BREW: 1.0, 1.1, 2.0, 2.1 e 3.0. A escolha de qual usar se baseia em dois critérios: plataforma alvo e recursos requeridos. Quanto mais antiga a versão, maior a quantidade de aparelhos que poderão rodar o aplicativo, pois versões mais novas são compatíveis com as anteriores. Por outro lado, recursos mais avançados poderão estar disponíveis apenas nas versões mais recentes. A maioria dos dispositivos que suportam o BREW hoje implementa a versão 1.1. Poucos modelos implementaram as versões 2.0 e 2.1 e devem ser lançados aparelhos que suportem a versão 3.0 até o final do ano de 2004 (QUALCOMM, 2004b). Na versão 3.0 a Qualcomm optou por retirar alguns dos recursos adicionados nas versões 2.0 e 2.1, quebrando a regra da compatibilidade retroativa.

Aplicativos BREW podem utilizar conteúdo *Multipurpose Internet Mail Extensions* (MIME). A MIME define uma estrutura com regras para codificação de mensagens que possuem conteúdo além de texto puro, como, por exemplo, conteúdo multimídia. Ela pode ser estendida para suportar novos tipos de conteúdo que devem ser interpretados por um programa manipulador.

4.1 Arquitetura do BREW

O BREW possui uma arquitetura aberta, no sentido de que fabricantes de aparelhos celulares têm acesso livre à sua especificação para suportarem o BREW. No dispositivo, o BREW é um pequeno programa, o AEE, que faz acesso direto ao *Application Specific Integrated Circuit* (ASIC), que é um circuito integrado projetado para operações específicas com otimizações arquiteturais utilizado como elemento de processamento em sistemas digitais, como os de um aparelho celular. O AEE disponibiliza uma interface para que os aplicativos possam acessar os recursos do aparelho de maneira genérica.

O BREW pode ser estendido através da inserção de novos módulos, como, por exemplo, uma API de jogos avançados ou uma máquina virtual Java, que podem ser compartilhados entre aplicações. Um módulo é composto por uma ou mais classes do BREW. Os *applets* são classes que podem ser executadas pelo usuário do dispositivo enquanto que as demais fornecem rotinas que podem ser chamadas por outras classes, inclusive de módulos diferentes. A Figura 4.1 ilustra esta arquitetura do BREW.

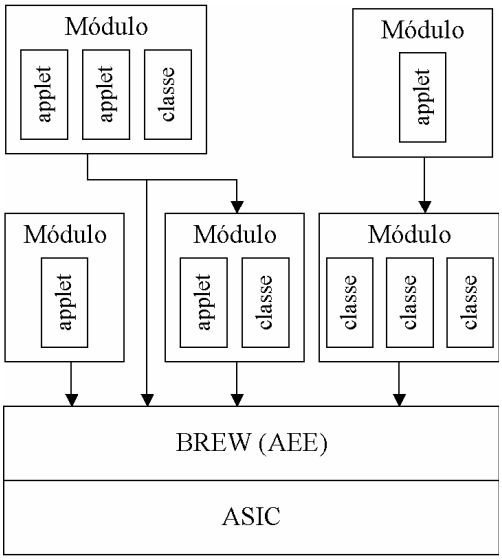

Figura 4.1: Arquitetura do BREW

Um módulo BREW é descrito por um *Module Information File* (MIF) que registra quais são as classes que pertencem ao módulo e quais são os identificadores dessas classes. Para cada classe do tipo *applet* são armazenados ícones e o seu nome, para que sejam exibidos no menu de *applets* do dispositivo. Além da descrição das classes e *applets*, o MIF contém informações sobre os níveis de privilégio do módulo, que determinam quais funções da API do BREW podem ser chamadas pelas suas classes. O MIF armazena também quais notificações que cada *applet* recebe de outras classes do BREW.

No caso de uma classe BREW ser um manipulador de conteúdo MIME, os tipos MIME que a classe manipula devem ser registrados no MIF. Deste modo, outras aplicações que tratam conteúdo MIME podem utilizar esta classe quando necessitam do tipo de conteúdo manipulado por ela.

O desenvolvedor pode criar diversos MIFs com imagens para dispositivos com características visuais distintas, como o número de cores e resolução.

O modelo dos programas BREW é baseado em eventos. Quando um evento ocorre, como o pressionamento de uma tecla ou a chegada de um pacote de dados da rede, é chamada a rotina do *applet* que faz o tratamento desses eventos. O ambiente BREW

possui um temporizador que encerra um aplicativo se ele demorar muito a tratar um evento, em geral por estar em um laço muito grande ou a espera de uma condição (*busy waiting*). Isto evita que o processador do dispositivo seja monopolizado por um aplicativo e também força que os programas sejam realmente orientados a eventos.

4.2 Ferramentas de desenvolvimento

A solução BREW disponibiliza uma série de ferramentas para o desenvolvimento de seus aplicativos. Esse *kit* de desenvolvimento de *software*, ou *software development kit* (SDK), contudo, só pode ser instalado nos sistemas operacionais Windows NT 4.0, Windows 2000 ou Windows XP. Versões anteriores do Windows não são suportadas, devido ao requerimento do tratamento de caracteres Unicode (QUALCOMM, 2004b). Além disso, a única plataforma de desenvolvimento oficialmente suportada é o Visual Studio, que também é da Microsoft.

O BREW SDK inclui manuais com a documentação das APIs e bibliotecas, utilitários para conversão de sons e imagens, emulador de celular, aplicativos de exemplo, editor de recursos, editor de informações de módulo e configurador de dispositivo. A documentação do BREW SDK está disponível inclusive na língua portuguesa. Os principais componentes e arquivos envolvidos no processo de desenvolvimento de aplicativos BREW estão relacionados na Figura 4.2.

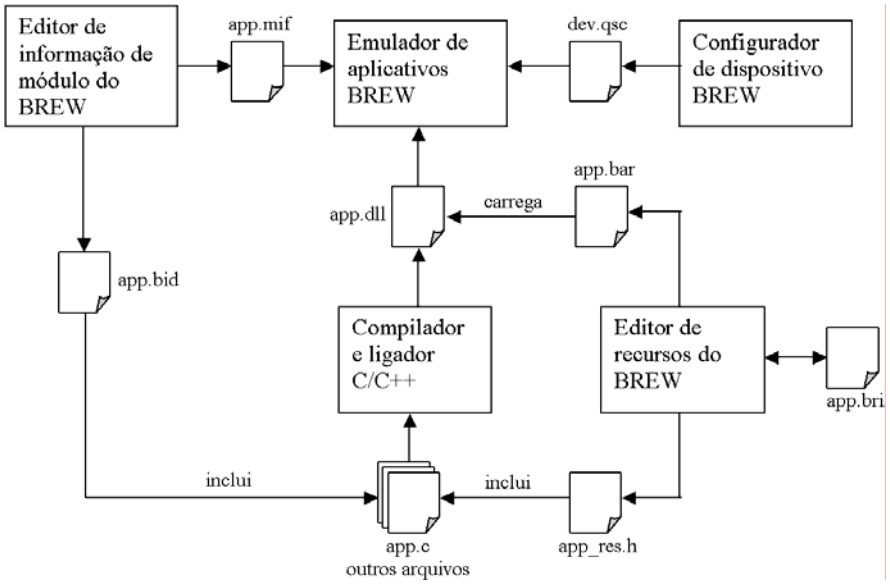

Figura 4.2: Componentes e arquivos envolvidos no desenvolvimento BREW

Tipicamente, o desenvolvedor inicia o processo de criação de um novo aplicativo utilizando o editor de informação de módulo registrando informações sobre classes e *applets* a serem criadas. A criação de um arquivo de recursos é recomendável, porém não obrigatória. Então, os programas são codificados em C ou C++ e fazem chamadas às APIs do BREW. O compilador gera o código binário do aplicativo que é carregado pelo emulador de celular. Diversos modelos de dispositivos podem ser utilizados pelo emulador sendo criados pelo configurador de dispositivo. A seguir, estes componentes serão descritos com mais detalhes.

4.2.1 Editor de informações de módulo

O editor de informações de módulo manipula MIFs através de uma interface simples. Pelo editor, o desenvolvedor registra quais *applets* e classes compõem um

módulo e informa seus identificadores. Podem ser criados identificadores locais, escolhidos pelo desenvolvedor, porém, estes só podem ser utilizados durante o desenvolvimento. Posteriormente, as classes devem ser registradas no *site* da Qualcomm que gera identificadores únicos para cada uma. O editor de MIF cria um arquivo com extensão .BID para cada classe. Este arquivo contém o identificador da classe e o desenvolvedor deve incluí-lo em seu projeto para poder instanciar a classe em seu programa.

O editor de MIF permite a importação de imagens, usadas como ícones dos *applets*, o registro de manipuladores de tipos MIME, a declaração de classes externas utilizadas e a definição dos níveis de privilégio do módulo.

4.2.2 Editor de recursos

O armazenamento de recursos em arquivos, além de tornar o desenvolvimento de aplicativos mais organizado, é um mecanismo que aumenta bastante a portabilidade de um aplicativo. Armazenando todas as informações de texto em um arquivo de recursos, por exemplo, permite facilmente que um aplicativo seja multilíngüe. Para isto, basta que seja criado um arquivo de recursos para cada língua e que o aplicativo carregue o arquivo de recursos conforme o idioma padrão do dispositivo ou escolhido pelo usuário. Interfaces, como menus e imagens, também podem estar armazenadas em um arquivo de recursos. Assim, pode-se criar arquivos de recursos para dispositivos específicos, com características visuais diferentes, por exemplo, mantendo o código do aplicativo inalterado.

O editor de recursos do BREW permite a criação de três tipos de recursos: *strings* (texto), imagens e diálogos.

Um recurso de *string* é formado por caracteres no formato Unicode, ASCII (ISOLATIN 1), KSC5601 ou S-JIS. Os formatos de imagens suportados pelo BREW SDK 1.1 são BMP, PNG e BCI. Um diálogo consiste em um ou mais controles que permitem a entrada de dados do usuário e aparecem em uma mesma tela no dispositivo. A Tabela 4.1 mostra os tipos de controle de diálogo suportados pelo editor de recursos do BREW e uma breve descrição de cada um.

Tabela 4.1: Tipos de controle suportados pelo editor de recursos do BREW

Tipo de controle	**Descrição**
Menu	Lista de opções mostradas uma abaixo da outra.
Tecla de *software*	Lista de opções mostradas com botões deslizantes horizontalmente.
Lista	Lista de opções que exibe somente o item de menu correntemente selecionado na tela.
Seleção de data	Lista de opções de datas mostradas em formato de calendário.
Data	Usado para selecionar datas no formato “dia da semana / dia do mês, ano”.
Cronômetro	Usado para contar tempo decorrido.
Relógio	Usado para especificar o tempo.
Contagem regressiva	Usado para contar regressivamente um intervalo de tempo.
Texto	Usado para manipular entradas de texto.
Modo de visualização de ícones	Lista de opções mostradas com ícones.

O editor de recursos do BREW utiliza um formato de arquivo intermediário para armazenar seu conteúdo, que possui extensão .BRI, de *BREW Resource Intermediate*. Este arquivo é utilizado como entrada para o compilador de recursos que cria um

arquivo de recursos para o *applet*, cuja extensão é .BAR, de *BREW Applet Resource* (BAR), e um arquivo de cabeçalho na linguagem C, cuja extensão é .H. A Figura 4.3 ilustra este processo.

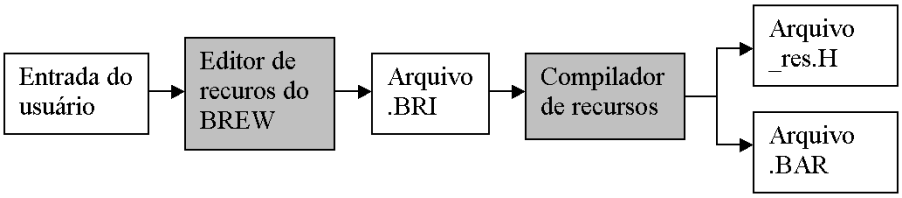

Figura 4.3: Processo de criação de recursos

Os recursos de um determinado tipo possuem identificadores únicos dentro de uma aplicação. Estes identificadores são colocados no arquivo de cabeçalho para que o programador tenha acesso aos recursos criados.

4.2.3 Compilador

Aplicativos criados em BREW podem ser compilados para teste, sendo executados num emulador de celular. Para isto, o programador deve utilizar o ambiente de desenvolvimento Microsoft Visual Studio. A partir da versão 1.1, o BREW SDK possui um assistente de aplicativos (*wizard*) para criação de projetos no Visual Studio, que automatiza o processo de configuração do ambiente e cria esqueletos de código para novos aplicativos BREW.

Para cada classe BREW utilizada, o desenvolvedor deve incluir o arquivo de extensão .BID correspondente, que contém a identificação da classe gerada pela Qualcomm ou localmente pelo editor de MIF. Apesar de sua extensão, o arquivo .BID contém, na verdade, uma definição de macro em linguagem C que associa o nome da classe a seu identificador. Assim, pode-se trabalhar com identificadores locais durante o desenvolvimento e apenas trocar os arquivos .BID quando forem gerados os identificadores oficiais.

Caso sejam utilizados arquivos de recursos, o programador deve incluir o arquivo _res.H gerado pelo editor de recursos. Este arquivo também possui macros em linguagem C com os identificadores dos recursos e uma referência ao arquivo .BAR que contém propriamente os recursos.

O programa é compilado para uma *Dynamic Link Library* (DLL) que será carregada pelo emulador de celular. É possível utilizar os recursos de depuração do ambiente Visual Studio, executando o programa passo-a-passo e acompanhando os resultados pelo emulador.

4.2.4 Emulador de celular

O emulador de celular do BREW permite testar as classes e os *applets* desenvolvidos. Diversos dispositivos móveis, com características diferentes de teclas, telas e quantidade de memória, podem ser emulados. Essas características são armazenadas por um arquivo de configuração do dispositivo que possui a extensão .QSC. Este arquivo também contém a imagem do dispositivo que ele representa. A Figura 4.4 mostra alguns desses modelos.

Figura 4.4: Exemplos de dispositivos do emulador de celular do BREW

Quando o emulador é iniciado ele carrega as configurações de um dispositivo e procura por arquivos MIF em um diretório especificado. Os MIFs possuem os nomes e os ícones dos *applets* que são exibidos na tela do dispositivo. Quando um *applet* é selecionado, o emulador carrega a DLL correspondente e a executa. O usuário pode interagir com o aplicativo através das teclas do dispositivo emulado e ver as respostas na sua tela.

A partir da versão 3.0 do BREW SDK, o configurador de dispositivo foi integrado ao emulador, que passou a ser chamado de simulador.

4.2.5 Configurador de dispositivo

O configurador de dispositivo permite que sejam criados ou editados os dispositivos usados pelo emulador. Em geral, esta ferramenta é utilizada pelos fabricantes de aparelhos celulares, que assim podem disponibilizar seus modelos para que os desenvolvedores os utilizem no emulador do BREW.

Uma imagem é utilizada para a definição da aparência visual do aparelho. A tela do dispositivo é definida em termos de profundidade de cores e resoluções horizontal e vertical. A posição das teclas é definida conforme a imagem do dispositivo e suas funções são atribuídas através da associação de eventos que são enviados para os aplicativos quando são ativadas. O configurador especifica a quantidade máxima de memória que o dispositivo dispõe e pode definir valores para a emulação de velocidade, como tempo de atualização da tela e tempo de escrita de arquivos. O arquivo de configuração de um dispositivo é salvo com a extensão .QSC e é carregado pelo emulador do BREW.

4.2.6 Conversor de som e de imagem

A Qualcomm criou uma tecnologia de codificação de voz chamada *PureVoice*, cujos arquivos possuem a extensão .QCP. O BREW SDK possui um conversor que

transforma arquivos de som do formato WAV em QCP e vice-versa. O módulo de reprodução de som do BREW SDK suporta os formatos MP3 e MIDI, além do QCP.

O BREW possui também um formato próprio para armazenamento de imagens e animações chamado *BREW Compressed Images* (BCI). O BREW SDK possui uma ferramenta que faz a conversão de imagens de formato padrão, como BMP e PNG, para o BCI. O editor permite também que sejam criadas animações, através de uma seqüência de imagens.

4.2.7 Documentação

A documentação do BREW SDK 1.1 foi traduzida para diversas línguas, inclusive para o português e está dividida em pequenos manuais. O manual do usuário proporciona uma visão geral do BREW SDK, seus componentes e instruções para iniciar o desenvolvimento de aplicativos e de uso do emulador. As funções e estruturas de dados do BREW que o desenvolvedor utiliza encontram-se no guia de referência da API do BREW. O editor de recursos, o editor de MIF, o configurador de dispositivo e o conversor de imagens possuem manuais próprios que descrevem seus usos. O conversor PureVoice está documentado no manual de utilitários. Material adicional sobre o BREW pode ser encontrado no *site* da Qualcomm.

4.3 Ciclo de desenvolvimento BREW

O ciclo de desenvolvimento de um aplicativo BREW, desde a concepção até a utilização pode ser dividido em etapas. A primeira etapa, de criação e codificação, requer o uso do BREW SDK, que é distribuído gratuitamente pela Qualcomm, e o ambiente de programação Visual Studio, comercializado pela Microsoft. Nesta etapa, o desenvolvedor é capaz de criar aplicativos que rodam em um ambiente de emulação e o investimento necessário é pequeno. Neste trabalho, foi exercitada esta primeira fase do desenvolvimento, descrita no capítulo 5.

Para que um aplicativo realmente chegue ao usuário final, é necessário passar por outras etapas que envolvem custos mais elevados. Para que seja usado em um celular, em primeiro lugar, precisa ser compilado especialmente para um. A Qualcomm recomenda o *ARM Realview Cross Compiler*, que compila C e C++ para processadores compatíveis com *Advanced RISC Machines* (ARM), que são utilizados pela maioria dos celulares. Esta ferramenta, entretanto, precisa ser adquirida pelo desenvolvedor. Existe a possibilidade de se usar o compilador gratuito GCC. A própria Qualcomm disponibiliza os arquivos necessários para o seu uso, porém não presta suporte sobre esta ferramenta.

O desenvolvedor deve registrar-se na Qualcomm para ter acesso a ferramentas fundamentais para colocar seu produto em um dispositivo real. Desenvolvedores autenticados têm acesso ao gerador de identificador de classes, às ferramentas para assinar digitalmente um módulo (AppSigner) e acessar o sistema de arquivos do BREW (AppLoader), depuração (Logger) e testes automatizados (Grinder) em dispositivos reais.

Para que um aplicativo possa executar em um dispositivo real, precisa ser assinado digitalmente pelo desenvolvedor, através do AppSigner, mesmo que seja somente para testes.

O BREW possui um sistema de arquivos, que fica no dispositivo, para armazenar e organizar pastas e arquivos. O AppLoader permite que o desenvolvedor acesse estas estruturas e possa colocar seus aplicativos no celular que, para ser utilizado neste processo, precisa ser habilitado para teste. Para isto, o aparelho deve ser enviado para

uma central da Qualcomm. Usuários finais utilizam um procedimento diferente para instalar aplicativos BREW, que é via o sistema de distribuição da operadora.

O BREW Logger permite a depuração de aplicativos rodando diretamente num aparelho celular. Com a ferramenta Grinder pode-se gerar seqüências de eventos para realização de testes automatizados.

Para que os aplicativos desenvolvidos cheguem ao mercado, O BREW define um sistema de distribuição, *BREW Distribution System* (BDS), que especifica o relacionamento entre as diversas entidades que participam do processo de negócio como um todo. As operadoras podem negociar diretamente com os desenvolvedores através de uma *extranet*, formando uma espécie de mercado virtual. Para que um aplicativo seja aceito para comercialização, as exige-se que ele tenha certificação TRUE BREW, após passar por testes de qualidade que são oferecidos por algumas empresas, representando mais um gasto para o desenvolvedor.

Depois de pronto e certificado, o desenvolvedor precisa determinar o modo de comercialização através da negociação com uma operadora de telefonia celular onde são determinados os preços e modo de cobrança, que pode ser por instalação, mensalidade ou número de usos. Recomenda-se também a utilização de demos, que podem ser usados por um período curto de tempo, ou com redução de funcionalidades.

A operadora organiza um catálogo de aplicativos BREW que oferece a seus clientes. Este catálogo é acessado através do próprio dispositivo sem fio. O BDS define um protocolo para a transmissão e instalação de aplicativos no celular do cliente, que inclusive gerencia o pagamento.

A cadeia de valor de um aplicativo BREW é mostrada pela Figura 4.5, onde aparecem as entidades envolvidas desde a criação até o uso do aplicativo.

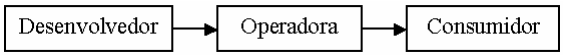

Figura 4.5: Cadeia de valor para aplicativos BREW

Isto quer dizer que o que o consumidor paga é dividido apenas entre operadora e desenvolvedor. É interessante comparar esta cadeia de valor com a cadeia de valor de jogos sofisticados tradicionais para computadores pessoais e *videogames*, também conhecidos como jogos AAA ou *triple A*, mostrada pela Figura 4.6(a). Mesmo os jogos casuais, que são mais simples e são distribuídos pela Internet possuem uma cadeia de valor, conforme Figura 4.6(b), bem maior do que a do modelo de negócios proposto pela solução BREW (NOLOGY, 2003).

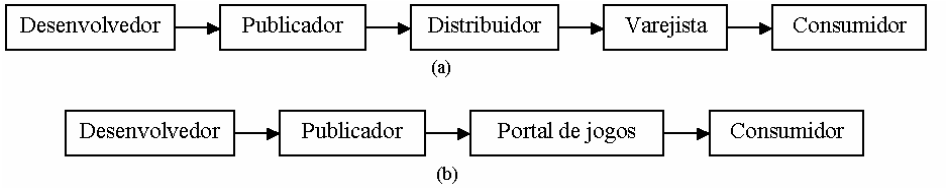

Figura 4.6: Cadeia de valor para jogos AAA (a) e jogos casuais (b)

Isto quer dizer que para estes jogos o valor pago pelo consumidor acaba sendo mais diluído entre as diversas entidades intermediárias e o desenvolvedor acaba recebendo uma parcela bem mais reduzida.

4.4 Resumo

O BREW surgiu como uma solução completa para a criação e comercialização de aplicativos para celulares. Ele trata tanto dos aspectos técnicos como dos aspectos de negócio, e já possui uma grande aceitação no mercado.

O BREW entrou no Brasil em 2004 através da operadora Vivo, que utiliza tecnologia CDMA e possui cerca de 25 milhões de clientes, correspondendo a 42% do mercado nacional (TELECO, 2004). A Vivo aposta na terceirização do desenvolvimento e oferece sua infraestrutura para comercialização dos aplicativos, dividindo a receita com os desenvolvedores.

5 IMPLEMENTAÇÃO DE UM JOGO

A indústria de jogos para celulares tem crescido rapidamente acompanhando a expansão do mercado de telefonia móvel e das novas tecnologias. Esta área desperta o interesse de muitas pessoas e as operadoras de telefonia sem fio estão investindo para oferecer diversas opções aos consumidores e lucrar com este negócio.

Do ponto de vista técnico, os jogos gerenciam processador, interfaces, controles, sons, imagens, cadência e comunicação (rede). Por isso, torna-se interessante a implementação de um jogo como meio de estudar os recursos de uma tecnologia como o BREW.

Existem restrições e características a serem consideradas na criação de um jogo para celular. Com base nisto, fez-se o levantamento de requisitos do jogo a ser desenvolvido. Então, escolheu-se um jogo que se adequasse às características especificadas e fez-se um projeto que foi implementado utilizando o BREW SDK para ser executado num emulador de celular. Os detalhes deste processo serão descritos neste capítulo.

5.1 Características de jogos para celulares

Jogos em celular devem ser simples. Primeiro porque os dispositivos móveis possuem restrições: o visor é pequeno, as teclas não são adequadas para jogos que requerem muita agilidade, o processador e a memória são limitados e o custo de conexão é alto. O segundo motivo é que os usuários não esperam jogos complexos em celulares. Normalmente esses jogos são utilizados como um passatempo de curta duração, como numa sala de espera, por exemplo. Por isso, o jogo deve ser fácil de aprender e trazer diversão rapidamente.

Do ponto de vista de conectividade, existem três tipos de jogos: os não conectados, ou *stand-alone*, os jogos *peer-to-peer* (P2P) e os jogos em rede (BENDAS; MYLLYAHO, 2002).

Os jogos não conectados tipicamente vêm integrados com o dispositivo. Esses jogos, envolvendo apenas um jogador e um aparelho, podem ser otimizados para sua plataforma e utilizar todos os recursos disponíveis, como som, vibração e imagens coloridas. Jogos não conectados propiciam respostas rápidas aos movimentos do usuário, pois não necessitam de dados externos (rede).

Jogos P2P utilizam canais de comunicação entre os jogadores. Podem ser utilizadas interfaces infravermelho ou redes *Bluetooth*, nas quais a distância entre os jogadores é limitada por estas tecnologias, ou então conexões telefônicas, na qual os jogadores podem estar fisicamente distantes.

Jogos em rede suportam vários participantes que compartilham um certo mundo virtual. Neste tipo de jogo sempre existe um servidor que armazena as informações sobre este mundo e intermedeia a comunicação entre os jogadores. A qualidade da

interatividade é influenciada diretamente pela latência e pelas taxas de transmissão da rede.

Os jogos conectados, tanto P2P como em rede, possuem um desafio a mais para os jogadores, que é a disputa entre humanos, o que não ocorre com os jogos não conectados, a menos que haja mais de um dispositivo de controle, como no caso dos *viedogames*. O fato de estes jogos dependerem de um canal de comunicação, no entanto, diminui a velocidade de resposta para o jogador e pode fazer com que o comportamento do jogo fique um tanto incerto, pois as mensagens podem chegar com atrasos aleatórios e ou mesmo serem perdidas.

5.2 Escolha do jogo

O propósito deste trabalho é estudar uma tecnologia e não o de criar um jogo revolucionário ou muito sofisticado. Uma característica interessante dos celulares é sua capacidade de aproximar pessoas, através da comunicação. Deseja-se aproveitar este atributo na criação de um jogo onde várias pessoas possam interagir, ou seja, um jogo multijogador.

A primeira idéia foi a de encontrar um jogo simples que funcionasse no modelo cliente-servidor, de código fonte aberto e escrito em linguagem C ou C++. Deste modo, implementar-se-ia uma adaptação do programa cliente para que funcionasse num celular e que se comunicasse com um servidor já existente. Evitar-se-ia a criação completa do jogo, focando o desenvolvimento na plataforma BREW.

Dois jogos que se encaixaram nessas características foram analisados mais detalhadamente: X-Pilot e X-Blast. No primeiro, cada jogador controla uma nave e voa pelo espaço atirando nos seus oponentes enquanto luta pela própria sobrevivência. No segundo, existe um labirinto onde os jogadores tentam destruir uns aos outros utilizando bombas. Ambos os jogos tiveram versões iniciais simples e, devido ao fato de estarem em poder público na Internet, foram evoluindo e ficando cada vez mais sofisticados.

A maior dificuldade no estudo desses jogos é que, apesar de terem código fonte aberto, pouca documentação foi encontrada sobre as implementações e protocolos utilizados. Além disso, desejavam-se informações sobre as primeiras versões destes jogos, pois eram mais simples. Porém, elas foram praticamente abandonadas devido às versões mais recentes serem mais interessantes para seus jogadores.

Fadado a decifrar várias e várias linhas de código, preferiu-se pela criatividade de inventar um jogo especialmente para os fins deste trabalho. A principal desvantagem desta abordagem, no entanto, é a necessidade de se implementar tudo do zero, inclusive um programa servidor caso seja utilizado. A principal vantagem, por outro lado, é a flexibilidade.

5.3 Projeto

Baseado nas características examinadas dos aparelhos celulares e nos desejos de que o jogo seja multijogador e seja concebido, e não adaptado, elaborou-se uma idéia básica de como seria este jogo. Então, escolheu-se a arquitetura sob a qual especificou-se o comportamento geral do jogo que foi detalhado através do projeto das interfaces. Por fim, foi realizada a consolidação do projeto através de uma visão global dando subsídios para o início da implementação do jogo.

5.3.1 Especificação funcional

O jogo inventado simula uma corrida de tartarugas e foi batizado de *Tartajat*. Cada jogador controla sua tartaruga através das teclas do seu telefone celular. A tartaruga que andar mais rapidamente chegando ao final primeiro é a vencedora. Para complicar, o jogador deve sincronizar as passadas da sua tartaruga, mantendo-as num certo ritmo para que ela não perca o fôlego e comece a andar mais lentamente.

Os jogadores se organizam em grupos para disputa de corridas. Ao fim de cada corrida são divulgados os resultados com a classificação e os tempos de chegada de cada tartaruga.

5.3.2 Arquitetura

Como se desistiu da idéia de portar um jogo existente, onde apenas o módulo cliente seria implementado e conectar-se-ia a um servidor, abriu-se a possibilidade do uso do modelo P2P. A vantagem, neste caso, é que apenas um programa precisa ser implementado. Porém, a existência de vários equipamentos processando o jogo independentemente dificulta o sincronismo e a determinação do estado atual verdadeiro do jogo. Precisaria haver um consenso, por exemplo, de qual jogador ganhou a corrida.

O modelo cliente-servidor, por outro lado, exige a implementação de dois programas diferentes. Porém, o servidor pode guardar um estado único do jogo e determinar o vencedor, por exemplo. Outra vantagem desta abordagem é que o processamento do jogo fica dividido entre os programas clientes e o servidor. E isto ajuda a resolver a questão de limitação de memória e processamento dos aparelhos celulares. A implementação do cliente fica facilitada também pelo fato de manter comunicação exclusivamente com o servidor.

Deste modo, resolveu-se seguir com o modelo cliente-servidor. Para jogar Tartajat, a pessoa deve se conectar ao servidor, criar um novo jogo e esperar que seus amigos se conectem ao jogo criado. O servidor gerencia vários jogos ao mesmo tempo, armazena as posições das tartarugas enviadas pelos clientes, faz a cronometragem da corrida e informa os resultados finais. O cliente controla o movimento e o fôlego de sua tartaruga e exibe as demais de acordo com informações recebidas pelo servidor.

5.3.3 Especificação técnica

Cada jogo possui um jogador líder, que é o que cria e dá nome ao jogo. Os demais jogadores podem pesquisar quais são os jogos existentes no servidor e escolher entrar em um deles. O servidor controla quais jogadores estão conectados, quais jogos existem e quais jogadores participam de cada jogo. Um jogador não pode estar em dois ou mais jogos ao mesmo tempo. Durante a corrida, o servidor coleta informações das posições das tartarugas atualizando a situação do jogo aos participantes, de tempos em tempos. O servidor também detecta possíveis perdas de conexão dos jogadores e os remove do jogo.

Quando o programa cliente conecta-se ao servidor ocasiona a criação de um novo jogador. A pessoa decide se deseja criar um novo jogo ou entrar em um jogo já criado, pedindo a lista de jogos disponíveis no servidor. O servidor controla quais os jogadores estão vendo a lista de jogos e os mantém atualizados quando jogos são criados ou ficam indisponíveis. Quando a pessoa entra em um jogo ela passa a ver a lista dos jogadores que atualmente participam do jogo. O servidor atualiza esta lista quando jogadores entram ou saem do jogo.

O jogador líder pode fechar o seu jogo para que outros jogadores não possam mais entrar. Quando um jogo é fechado ele sai da lista de jogos disponíveis e o jogador líder

tem a opção de iniciar uma corrida. Quando isto acontece, o servidor informa todos os clientes para preparar o início de uma corrida. Quando todos confirmam que estão prontos, o servidor aguarda um instante para então enviar o sinal de largada.

Durante a corrida cada programa cliente controla a situação de sua tartaruga. O jogador alterna o pressionamento de duas teclas fazendo sua tartaruga andar. A cada movimento da tartaruga ela perde uma quantidade de fôlego que é recuperado à medida que o tempo passa. Quando maior o fôlego da tartaruga, maior é a sua passada. A cada passo da tartaruga é enviada uma mensagem ao servidor informado sua nova posição. O servidor é responsável por informar as posições das tartarugas a todos os jogadores.

O servidor identifica as tartarugas que cruzaram a linha de chegada e, quando a corrida é encerrada, informa aos jogadores a classificação final com a colocação e os tempos de chegada de cada tartaruga. Então, o jogador líder pode encerrar o jogo ou iniciar uma nova corrida.

5.3.4 Interfaces

Para se ter uma idéia mais precisa de como será o funcionamento do jogo, especificou-se as interfaces utilizadas pelo jogador e as interações entre o programa cliente e o servidor.

5.3.4.1 Tela de login

O *login* consiste simplesmente na apresentação do jogador ao servidor. Não existe uma base de dados que armazene informações dos jogadores. Na tela mostrada pela Figura 5.1, o jogador insere seu nome (1) e escolhe entre sair do jogo (2) ou fazer o *login* no servidor (3). O botão de sair simplesmente encerra a execução do aplicativo.

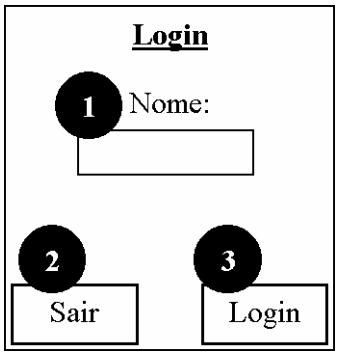

Figura 5.1: Tela de *login*

Quando o botão de *login* é acionado, uma mensagem solicitando a criação de um novo jogador é enviada ao servidor. O servidor então registra o jogador com o nome informado e devolve uma mensagem de confirmação ao cliente. Enquanto o usuário não recebe essa confirmação ele fica numa tela de espera. Passado um tempo limite sem resposta do servidor, o jogador é informado de que a operação não foi bem sucedida e a tela de *login* é exibida novamente.

5.3.4.2 Tela do menu principal

Confirmado o *login*, o jogador passa para a tela do menu. Nesta tela, conforme a Figura 5.2, ele tem a opção de criar um novo jogo (1), ver a lista de jogos existentes (2), ou sair (3).

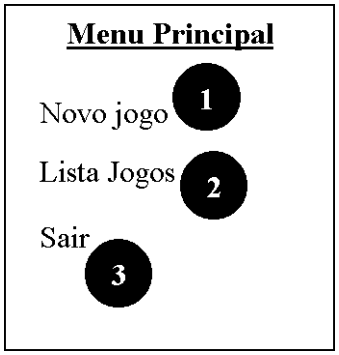

Figura 5.2: Tela do menu principal

Quando o jogador escolhe criar um novo jogo ele vai para a tela de criação (Figura 5.3). Se ele escolhe ver a lista de jogos, o cliente envia uma mensagem ao servidor requisitando a lista. O servidor devolve a lista de jogos e registra que este jogador está vendo a lista, para mantê-lo atualizado. Então, o programa cliente vai para a tela que mostra a lista de jogos (Figura 5.4). Se ocorrer algum problema na comunicação e o programa cliente não receber a lista de jogos, o jogador é informado que não foi possível obter a lista e volta para a tela do menu. Quando o jogador seleciona sair no menu, o programa envia uma mensagem ao servidor informado que o jogador não existe mais e volta para a tela de *login*.

5.3.4.3 Tela de criação de um novo jogo

Na tela de criação de novo jogo, ilustrada pela Figura 5.3, o jogador entra com um nome para o jogo (1). Quando o botão de criar for selecionado (2), é solicitada ao servidor a criação de um jogo. O servidor, então, registra o jogo com o nome dado e seu líder, que é o jogador que o criou. Quando o jogador cria um jogo ele automaticamente faz parte daquele jogo. Quando o servidor confirma a criação do jogo, o programa cliente vai para a tela de lista de jogadores da Figura 5.6. Se o cliente não receber a confirmação ou selecionar o botão de cancelar (3) ele volta para a tela do menu.

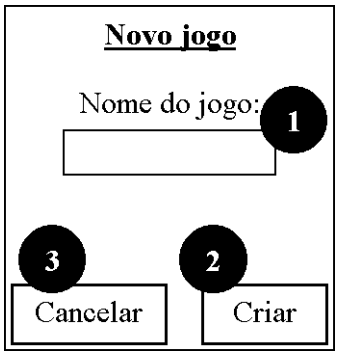

Figura 5.3: Tela de criação de um novo jogo

5.3.4.4 Tela da escolha de jogo

Na tela que mostra a lista de jogos (Figura 5.4), o jogador seleciona qual jogo deseja participar (1). Se, neste meio tempo, algum jogo for criado, deixar de existir ou ficar indisponível por ter sido fechado, o servidor envia uma mensagem com a lista de jogos atualizada para os clientes que estão na tela a lista de jogos. O botão de entrar (2) só fica disponível se há algum jogo selecionado. Quando acionado, é enviada uma mensagem ao servidor, informando que o jogador entrou no jogo. O servidor responde mandando a lista dos jogadores participantes do jogo e então aparece a tela com a lista dos jogadores

(Figura 5.5). Se o cliente não receber resposta, o jogador é informado que não foi possível entrar no jogo e ele volta para a tela da lista de jogos. Escolhendo o botão voltar (3), o programa envia uma mensagem ao servidor informado que o jogador não está mais vendo a lista de jogos e vai para a tela do menu.

Figura 5.4: Tela de escolha de jogo

5.3.4.5 Tela da lista de jogadores

Quando o jogador escolhe e entra num jogo, passa a ver a lista dos jogadores (Figura 5.5) e deve esperar até que uma corrida comece. Cada vez que um jogador entrar ou sair deste jogo, o cliente receberá uma mensagem do servidor com a lista atualizada dos participantes e reexibirá a lista de jogadores (1). Se a pessoa desiste e sai do jogo (2), o cliente avisa o servidor para que retire o jogador deste jogo e volta para a tela de escolha de jogo.

Figura 5.5: Tela da lista de jogadores

Depois que o jogador cria um jogo, é exibida a tela da lista dos jogadores para o jogador líder (Figura 5.6). Além de acompanhar a lista dos participantes do jogo (1), o jogador líder pode fechar ou abrir seu jogo (2), iniciar uma corrida (3) ou encerrar o jogo (4).

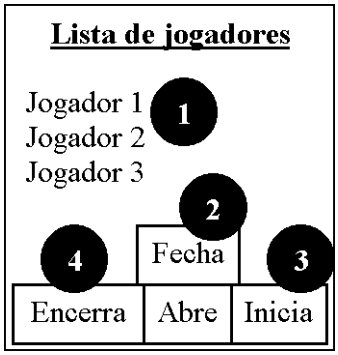

Figura 5.6: Tela da lista de jogadores para o jogador líder

Inicialmente o jogo está aberto, ou seja, está disponível para que novos jogadores entrem. Se o jogador líder decide fechar o jogo, é enviada uma mensagem ao servidor para que o jogo seja fechado. Então, o servidor, retira o jogo da lista dos jogos disponíveis e envia a lista atualizada para os jogadores que estão na tela de escolha de jogo. Quando o jogo está fechado, o botão de fechar é substituído pelo botão de abrir. Se o jogo for aberto novamente, o servidor é informado, recoloca o jogo na lista dos jogos disponíveis e atualiza a lista para os clientes que a estão vendo. O botão de iniciar uma corrida (3) só aparece quando o jogo está fechado. Ao ativar este botão, uma mensagem é enviada ao servidor informando o início de uma nova corrida. O servidor repassa esta informação aos participantes deste jogo que vão para a tela de corrida.

5.3.4.6 Tela de corrida

Cada tartaruga fica numa raia de uma pista de corrida, no estilo das de 100 m rasos olímpico. A informação de em qual raia a tartaruga de um jogador está é enviada pelo servidor no momento em que ele informa o início de uma corrida.

O cliente informa o servidor quando está pronto para a largada. Quando todos os clientes estiverem prontos, o servidor espera um certo tempo aleatório e envia o sinal de largada. A partir daí o programa cliente aceita os comandos do jogador, permitindo que sua tartaruga ande. O jogador deve alternar o pressionamento de duas teclas do seu celular, representando passadas com patas direita e esquerda da tartaruga. Se o jogador apertar duas ou mais vezes em uma mesma tecla consecutivamente, apenas o primeiro toque é considerado.

O programa cliente gerencia informações sobre a sua tartaruga, que possui uma posição e uma medida de fôlego. Ele calcula o tamanho de uma passada que é diretamente proporcional ao fôlego atual da tartaruga. O fôlego é reduzido de uma certa quantidade a cada passada da tartaruga e é restaurado ao longo do tempo. A cada movimento da tartaruga, é enviada uma mensagem ao servidor com a nova posição atingida.

A tela de corrida, mostrada na Figura 5.7, informa ao jogador sua colocação atual (1), que é calculada no cliente e não necessariamente reflete a realidade do servidor, seu fôlego (2) e, graficamente, as tartarugas na pista de corrida (3).

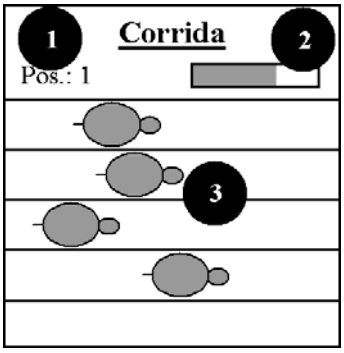

Figura 5.7: Tela de corrida

O servidor mantém informação das posições de todas as tartarugas e do tempo transcorrido desde o início da corrida e envia mensagens aos clientes com as posições de todas as tartarugas. O servidor verifica também quais tartarugas cruzaram a linha de chegada e os seus tempos. Quando todas as tartarugas completam a corrida, o servidor envia mensagens com a colocação e o tempo de chegada de cada uma.

5.3.4.7 Tela de classificação

Quando recebe o resultado final da corrida, o programa cliente mostra a tela com a classificação, como na Figura 5.8. Depois de avaliar o seu desempenho (1), o jogador volta para a tela com a lista dos jogadores (2), onde o jogador líder pode decidir iniciar uma nova corrida.

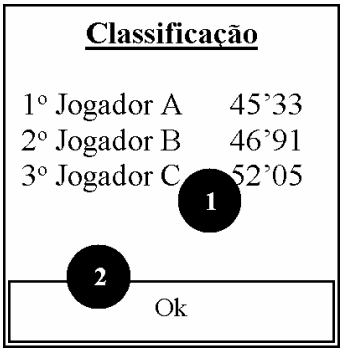

Figura 5.8: Tela de classificação

5.3.5 Resultado

O diagrama da Figura 5.9 mostra uma visão global do comportamento do jogo com os principais estados do cliente, mensagens trocadas e ações do servidor.

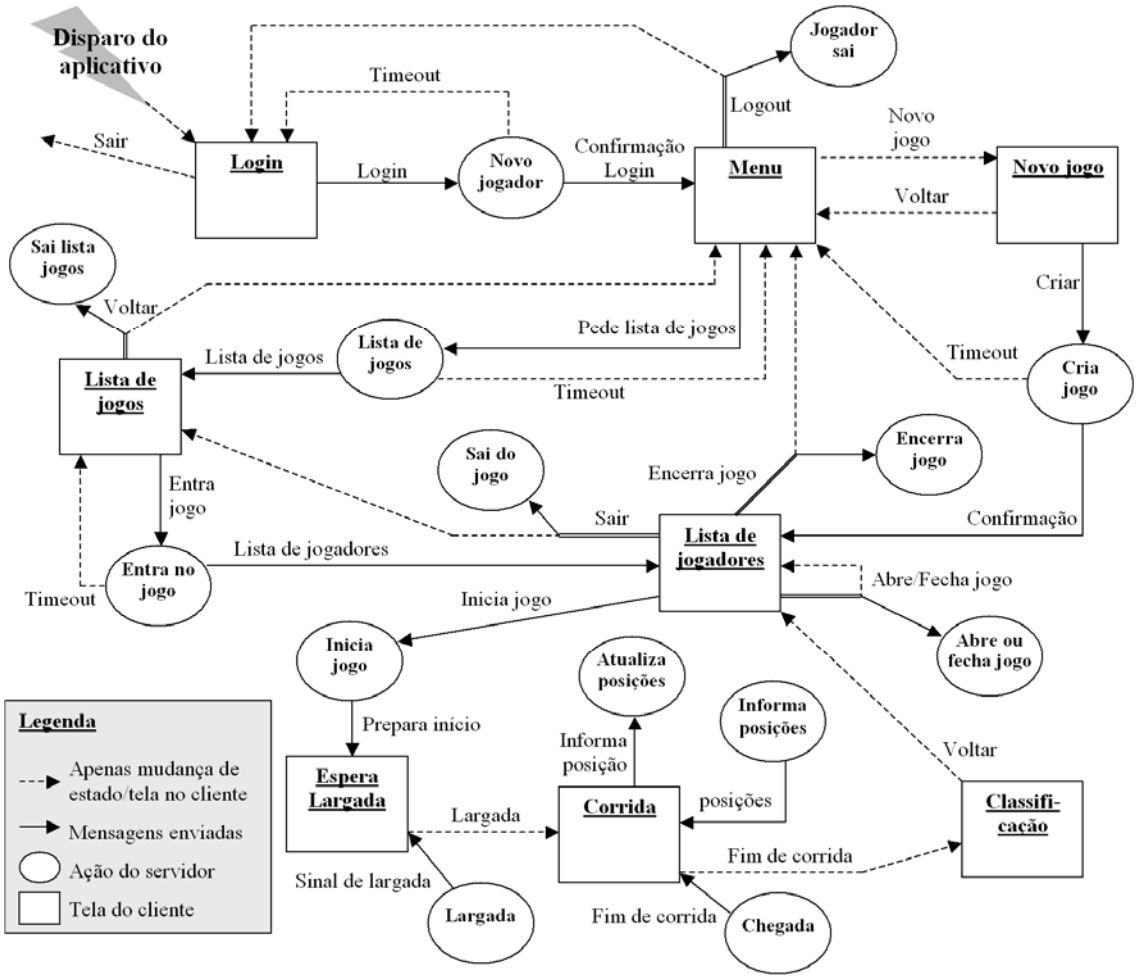

Figura 5.9: Diagrama de mensagens e transições de estados

5.4 Implementação

Em primeiro lugar, foi definido o protocolo de comunicação entre o programa cliente e o programa servidor. Depois, foi implementado o servidor e, por último o cliente foi desenvolvido.

5.4.1 Protocolo de comunicação

A primeira decisão a ser feita foi a escolha do protocolo de transporte a ser utilizado. Existem dois protocolos de transporte sobre o *Internet Protocol* (IP) que são amplamente usados: o *Transmission Control Protocol* (TCP) e o *User Datagram Protocol* (UDP), ambos suportados pelo BREW. O TCP é um protocolo orientado a conexão confiável. Possui mecanismos de retransmissão e garante o ordenamento das mensagens. Já o UDP é um protocolo orientado a datagramas e não é confiável. Porém, sendo um protocolo mais leve, possui menor latência que o TCP e economiza recursos da rede, pois têm cabeçalhos menores e não faz retransmissão.

Como o jogo não é uma aplicação crítica, onde a precisão das informações seria muito importante, e como a velocidade e economia de recursos são fatores importantes, o decidiu-se pelo UDP.

Foi definido um formato básico para todas as mensagens do protocolo, ilustrado pela Figura 5.10. O primeiro *byte* é dividido em duas partes. Visto que várias mensagens poderão ser mensagens de confirmação, o primeiro *bit* é usado para isto. Se estiver

ligado, indica que esta é uma mensagem de confirmação (ack). Os demais *bits* do primeiro *byte* contêm o código que identifica o tipo de mensagem. Depois, podem vir zero ou mais *bytes* contendo mais informações conforme o tipo de mensagem.

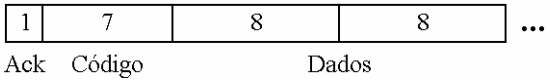

Figura 5.10: Formato das mensagens

Com base no projeto do jogo, definiram-se as mensagens e as informações necessárias que clientes e servidor precisam trocar, que estão descritas na Tabela 5.1. Os inteiros de 16 bits transmitidos no formato *big endian*: o primeiro *byte* contém os 8 *bits* mais significativos. As *strings*, utilizadas nas transmissões dos nomes de jogos e jogadores são conjuntos de caracteres de um *byte* cada, terminando com um *byte* de valor zero e são restritas a no máximo 16 *bytes* (15 caracteres mais o *byte* de terminação).

Tabela 5.1: Mensagens trocadas entre cliente e servidor

Mensagem	**Cód.**	**Ack**	**Dados**
Novo jogador (C)	1	0	*String* com o nome do jogador.
Novo jogador (S)	1	1	Não há dados.
Cria jogo (C)	2	0	*String* com o nome do jogo.
Cria jogo (S)	2	1	Não há dados.
Entra jogo (C)	3	-	Um *byte* contendo o número do jogo.
Abre jogo (C)	4	-	Não há dados.
Fecha jogo (C)	5	-	Não há dados.
Inicia jogo (C)	6	0	Não há dados.
Inicia jogo (S)	6	1	Um *byte* contendo o número da raia da tartaruga.
Remove jogador (C)	7	-	Não há dados.
Lista de jogadores (S)	8	-	Primeiro *byte* contém o número de jogadores da lista. Seguem *strings* com o nome dos jogadores, no máximo 6.
Lista de jogos (S)	9	-	Primeiro *byte* contém o número de jogos. Seguem *strings* com o nome dos jogos, no máximo 10.
Encerra jogo (C)	10	-	Não há dados.
Heartbeat (S)	11	0	Não há dados.
Heartbeat (C)	11	1	Não há dados.
Atualiza posições (C)	12	0	Um inteiro de 16 *bits* com a posição da tartaruga.
Atualiza posições (S)	12	1	O primeiro *byte* contém o número de tartarugas. Para cada tartaruga, segue um inteiro de 16 *bits* com a sua posição.
Fim de corrida (S)	13	-	O primeiro *byte* contém o número de tartarugas. Para cada uma, um *byte* com o número da tartaruga e um inteiro de 16 *bits* com o tempo de chegada em décimos de segundo.
Sai da lista de jogos (C)	14	-	Não há dados.
Sai do jogo (C)	15	-	Não há dados.
Largada (S)	16	-	Não há dados.

Mensagens marcadas com um (C) são as enviadas pelos clientes, e as marcadas com um (S) são as enviadas pelo servidor. A identificação de quem enviou a mensagem é obtida pelos dados do protocolo de transporte (porta) e de rede (endereço IP). Deste

modo, não há necessidade de se utilizar um identificar explícito de jogador na composição da mensagem.

5.4.2 Implementação do servidor

O servidor foi implementado na linguagem C++, através do ambiente Visual Studio. O programa foi dividido em diversas classes que realizam tarefas específicas e interagem entre si como mostra a Figura 5.11. As setas indicam que uma classe utiliza a outra.

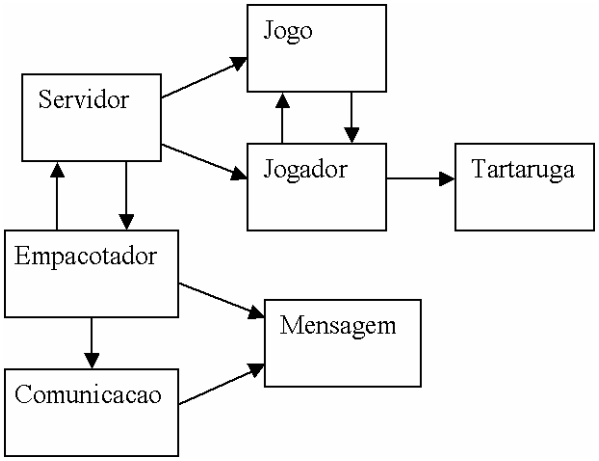

Figura 5.11: Diagrama de classes do servidor

O objetivo do servidor é gerenciar os jogadores, jogos existentes e suas corridas. A classe Servidor trata do controle lógico dessas estruturas, através de métodos que executam essas atividades. A Tabela 5.2 mostra os métodos existentes na classe Servidor e suas descrições.

Tabela 5.2: Métodos da classe Servidor

Método	**Descrição**
abreJogo (lider)	Abre o jogo do jogador líder e atualiza a lista de jogos.
ackIniciaJogo (jogador)	Registra que um jogador está preparado para a largada. Quando todos estiverem preparados, aguarda-se de 1 a 3 segundos aleatoriamente e o sinal de largada é enviado.
atualizaListaJogadores (jogo)	Envia mensagens aos participantes de um jogo com a lista dos jogadores deste jogo.
atualizaListaJogos ()	Envia mensagem com a lista de jogos abertos a todos os jogadores que estão vendo a lista.
atualizaPosicoes (jogo)	Envia mensagem com a posição de todas as tartarugas aos participantes de uma corrida.
criaJogador (nome)	Cria jogador com o nome passado como parâmetro e envia mensagem de confirmação.
encerraJogo (lider)	Encerra o jogo do jogador líder, retira todos os participantes e atualiza a lista de jogos.
entraJogo (jogador, jogo)	Se o jogo está aberto, insere o jogador no jogo e atualiza a lista de jogadores.
fechaJogo (lider)	Fecha o jogo do jogador líder e atualiza a lista de jogos.
fimCorrida (jogo)	Encerra a corrida e envia a classificação final aos seus participantes.
informaPosicao (jogador, posicao)	Registra nova posição da tartaruga do jogador e detecta o fim da corrida.
iniciaJogo (lider)	Prepara inicio de uma corrida e envia mensagem de aviso a todos os jogadores.
listaJogos (jogador)	Envia a lista de jogos ao jogador.
novoJogo (nome, lider)	Cria um novo jogo com um nome e um jogador líder associado.
removeJogador (jogador)	Remove o jogador do servidor e encerra seu jogo se for um jogador líder.
saiJogo (jogador)	Retira jogador do jogo de que participa atualmente e atualiza a lista de jogadores.
saiListaJogos (jogador)	Registra que o jogador não está mais vendo a lista de jogos.

A classe Servidor manipula três classes que guardam informações importantes: classe Jogo, classe Jogador e classe Tartaruga.

A classe Jogo armazena um número de identificação, seu nome, os jogadores participantes, o jogador líder, seu estado que pode ser aberto, fechado ou em corrida e o tempo da última largada, caso já tenha ocorrido. A classe Jogador armazena o nome do jogador, seu número de identificação, o jogo do qual faz parte e sua tartaruga. A classe Tartaruga contém seu número, sua localização e seu tempo de chegada da última corrida.

A classe Servidor interage fortemente com a classe chamada Empacotador. Esta classe faz a montagem das mensagens que serão enviadas pela rede e interpreta as mensagens que chegam. A classe Servidor chama métodos da classe Empacotador que montam as mensagens de acordo com o protocolo definido. Quando novas mensagens chegam seus elementos são identificados pela classe Empacotador que chama os métodos apropriados da classe Servidor. Ou seja, a classe Empacotador é a classe que trata do protocolo de comunicação do Tartajat no lado do servidor. Ela mantém uma tabela com endereços IP e portas UDP utilizadas por cada jogador. Assim, pode identificar os remetentes das mensagens que chegam e enviar mensagens para os jogadores certos.

A classe que efetivamente envia e recebe as mensagens pela rede é a classe Comunicacao. Esta classe utiliza a biblioteca de *sockets* do Windows e disponibiliza uma interface bem simples (um método para enviar mensagens e um para receber) utilizada pela classe Empacotador. Esta abordagem facilita a portabilidade do servidor, pois, para utilizar uma biblioteca de *sockets* diferente basta trocar a classe Comunicacao.

Quando o servidor é inicializado, ele configura uma porta UDP para receber as mensagens dos clientes. Foi definida a porta de número 2000, através da qual os clientes se comunicam com o servidor para jogar Tartajat.

5.4.3 Implementação do cliente

Para a implementação do cliente Tartajat foi escolhido o BREW SDK 1.1, pois esta versão ainda é a mais utilizada, sendo suportada pela maioria dos dispositivos BREW. Além disso, os recursos desta versão são suficientes para esta implementação.

Foi criado um módulo com apenas uma classe do tipo *applet* que executará o jogo. Através do editor de MIF, foi criado um identificador local para o *applet*. Como o programa utilizará recursos de rede, foi preciso habilitar este nível de privilégio pelo editor. Como não são utilizadas classes externas, nada mais precisou ser feito na criação do arquivo MIF.

Então, configurou-se um projeto no Visual Studio para o desenvolvimento do *applet*. Foram definidos os estados mostrados pelo diagrama da Figura 5.12. As transições de estado são feitas a partir de eventos que podem ser de três tipos: eventos do servidor, eventos internos ou eventos do jogador.

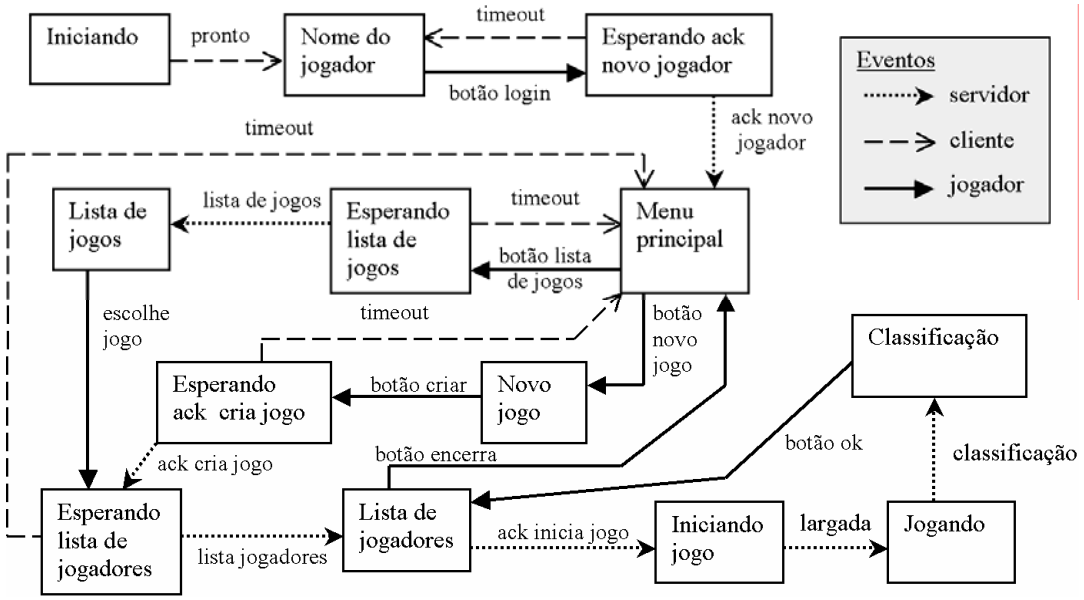

Figura 5.12: Diagrama de estados do cliente

Os eventos do servidor são as mensagens que ele envia ao cliente. Os eventos internos são gerados pelo próprio cliente, como, por exemplo, os eventos de *timeout*. Os eventos do jogador são aqueles disparados por uma ação do próprio jogador, como o acionamento de uma tecla. Nem todos os eventos estão mostrados na Figura 5.12, apenas os que implicam numa transição de estado. Se ocorrer algum evento inesperado ou algum outro problema, o cliente entra num estado de erro, que também não está mostrado pelo diagrama. Neste estado é exibida uma mensagem de erro ao jogador e o aplicativo é encerrado.

Quando o *applet* é disparado, é feita a inicialização dos recursos de tela e de rede. Então os diálogos da tela de *login* são carregados e mostrados na tela. Nos estados de espera, é exibida uma mensagem para que o jogador aguarde, mas que muitas vezes fica apenas por um curto período de tempo e não é percebida. Quando ocorre um *timeout* é exibida uma mensagem de problema na comunicação.

As telas são formadas por diálogos que estão organizados na forma de uma pilha. A tela exibida é a do diálogo que está atualmente no topo da pilha. Quando uma nova tela abre, seu diálogo é empilhado e torna-se ativo. Quando o jogador volta para uma tela anterior, o diálogo atual é desempilhado e a tela do diálogo logo abaixo é exibida.

Os diálogos para as telas foram criados com o editor de recursos do BREW. Todos os textos do jogo também foram armazenados no arquivo de recursos. As entradas de dados das telas de *login* (nome do jogador) e criação de jogo (nome do jogo) são feitas por controles de texto, como mostra a Figura 5.13(a). A tela de menu foi implementada por um controle de menu, ilustrado pela Figura 5.13(b), permitindo que as opções sejam listadas uma embaixo da outra. Já a Figura 5.13(c) demonstra o uso das teclas de *software* (encerra e fecha) da tela da lista de jogadores.

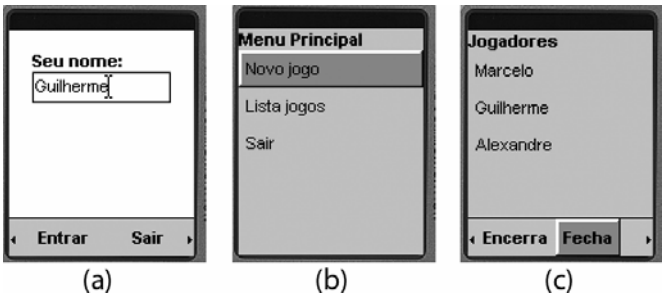

Figura 5.13: Uso de diálogos em (a) tela de login; (b) menu; (c) lista de jogadores

Quando um diálogo está ativo é ele quem recebe os eventos de pressionamento de teclas. Os controles do BREW possuem rotinas prontas que fazem esse tratamento. O controle de texto, por exemplo, gerencia a entrada de texto através do teclado numérico do celular. A navegação entre os diversos controles de uma tela também é implementada pelo próprio diálogo. Assim, o programador precisa apenas tratar os eventos de mais alto nível, como o acionamento de uma opção do menu, por exemplo.

Durante uma corrida, o programa controla a posição e o fôlego de sua tartaruga. O fôlego é representado por um número inteiro de 0 a 100, inicializado com o valor máximo. A posição da tartaruga é representada por um inteiro de 16 *bits* que, na largada, tem o valor zero. Depois de receber o sinal de largada, o programa aceita os comandos do jogador que deve alternar as teclas “1” e “3” para movimentar sua tartaruga. Quando um comando válido é verificado, a posição da tartaruga é acrescida do valor do fôlego que é reduzido em um quinto. No início, por exemplo, o fôlego é 100 e o primeiro passo coloca a tartaruga, que estava na posição zero, na posição 100 e o fôlego reduz a 80. Existe um evento que é agendado para ocorrer a cada 150 ms que faz o fôlego aumentar em 4 unidades, até o limite de 100. A cada movimento, uma mensagem é enviada ao servidor informando a nova posição da tartaruga. A Figura 5.14 mostra a tela de uma corrida com cinco jogadores.

Figura 5.14: Tela de corrida do Tartajat

Quando uma mensagem de atualização das posições chega ao cliente, ele atualiza as posições das tartarugas adversárias, porém, mantém a posição da sua própria tartaruga, evitando que ela possa voltar para um estado anterior. Neste momento é calculada a colocação atual da própria tartaruga que é exibida como referência para o jogador. Esta posição, contudo, não reflete necessariamente a realidade, pois somente o servidor possui a posição atual de todas as tartarugas.

Nesta implementação foram utilizados o módulo AEEText para o tratamento dos controles de menu, o módulo AEEImage para a geração de imagens da corrida e o módulo AEENet para a comunicação através de *sockets*, além dos módulos obrigatórios AEEAppGen, para criação de *applets*, e AEEShell para gerenciamento do aplicativo e uso dos diálogos. Todos estes módulos estão disponíveis no BREW SDK 1.1.

5.5 Resumo

Este capítulo descreveu o processo de criação de um jogo e sua implementação como um aplicativo BREW. O resultado foi um demo jogável em um emulador, inclusive com o servidor e vários clientes podendo estar computadores distintos, conectados por uma rede local comum, ou mesmo pela Internet.

Antes de passar para a etapa de testes em dispositivos reais, esta aplicação ainda precisa resolver alguns pontos práticos. Por exemplo, o local onde o servidor ficaria. Poderia estar instalado em um computador controlado pela operadora de telefonia celular ou então ser um servidor independente na Internet. Outra abordagem seria propor que o servidor ficasse em algum dos aparelhos celulares. Porém, imagina-se que isso poderia sobrecarregá-lo.

Muitas idéias de melhorias surgiram. O servidor poderia guardar um banco de dados com o registro dos jogadores e seus melhores resultados. Maneiras mais sofisticadas de se encontrar pessoas e organizar jogos poderiam ser pensadas. A implementação atual baseia-se simplesmente no uso de nomes. A corrida de tartarugas em si poderia ser incrementada. A adição de obstáculos, poderes especiais, condições de tempo, como neve e chuva certamente fariam deste um jogo mais interessante, assim como a organização de campeonatos.

Alguns métodos poderiam ser utilizados para atenuar o fato de que as tartarugas dos oponentes podem estar sendo vistas com atraso. Pode-se calcular a velocidade média de cada tartaruga com base nas duas últimas informações de posição de cada tartaruga. Então, estima-se a posição de cada tartaruga a cada instante. Pode-se estimar também qual o tempo médio que o servidor leva para informar uma nova posição da tartaruga a todos os jogadores depois que seu jogador ordenou o movimento. Com esta informação mais a velocidade média estimada das outras tartarugas, pode-se prever com relativa precisão a posição atual das tartarugas adversárias. Não ocorre acumulação de erro, pois cada vez que o servidor informa novas posições, os cálculos seriam corrigidos.

6 CONCLUSÃO

Atualmente passamos por uma era de grande expansão no mercado de telefones móveis. Muitas são as oportunidades de negócio nesta área, tanto para grandes quanto para pequenas empresas.

As grandes empresas estão investindo em serviços que vão além da comunicação por voz. A cada ano, a transmissão de dados evolui significativamente e o telefone celular deixou de ser um objeto apenas de comunicação para atender a outras necessidades humanas. Um exemplo disto é o uso destes aparelhos como relógio e despertador, o que acabou prejudicando fabricantes de relógios de pulso. Essa expansão de funções do celular gera problemas para algumas empresas, mas gera grandes oportunidades para outras.

Os aplicativos para dispositivos móveis podem ser desenvolvidos por pequenas empresas e até mesmo por apenas uma pessoa, desde que ela conheça a tecnologia. Porém, para que possam comercializar seus produtos precisam do apoio das operadoras de telefonia celular. Esta é a atual situação do mercado. A Vivo, por exemplo, aposta nestas pequenas empresas. Em 2004 ela promoveu um *workshop* para desenvolvedores. Neste evento, em parceria com a Qualcomm, foi divulgada a solução BREW, para incentivar o mercado de desenvolvimento de aplicativos e posicionar a Vivo como um espaço aberto para esses profissionais. No próprio *workshop*, a Vivo deixou explícito que prefere negociar diretamente com os desenvolvedores, sem intermediários. Analisando a situação, pode-se concluir que é mais vantajoso fazer negócio com os pequenos do que manter uma equipe de desenvolvimento própria.

A solução BREW, apesar de teoricamente não estar limitada a uma tecnologia de comunicação específica, ainda está muito ligada à tecnologia CDMA, pois surgiu neste contexto. A Qualcomm foi a empresa que desenvolveu a tecnologia CDMA e posteriormente o BREW. Imagina-se que essa raiz comum facilitou e incentivou a comunhão das duas tecnologias. A Vivo, ao adotar o CDMA, criou a tendência de seguir seu desenvolvimento através do BREW.

BREW é uma solução de tecnologia e de negócios. Uma solução completa que é interpretada de várias maneiras. Para os desenvolvedores é um conjunto de APIs, bibliotecas e ferramentas de desenvolvimento. Para as operadoras, o BREW é um modelo de negócios e de distribuição de aplicativos. O *site* do GuiaSP (2004) define o BREW para seus clientes como “uma nova tecnologia capaz de baixar aplicativos diretamente para o seu celular”. O GuiaSP é um *site* que mantém um banco de dados com informações sobre serviços, como restaurantes e cinemas, da cidade de São Paulo. Seus clientes podem acessar essas informações também pelo celular, através de um aplicativo BREW.

Este trabalho focou seu estudo nas ferramentas de desenvolvimento do BREW. A vasta documentação existente, juntamente com os programas de exemplo, facilita bastante o aprendizado desta tecnologia.

O emulador é, com certeza, uma ferramenta indispensável no desenvolvimento de aplicativos para celular. Ele facilita muito os testes com diversos tipos de dispositivos, principalmente no quesito de apresentação do software, já que existem grandes variações nos tipos de tela. A possibilidade de realizar depuração de programas utilizando conjuntamente o ambiente de programação e o emulador é um fator altamente positivo.

Um aspecto interessante nas bibliotecas do BREW é sua capacidade de adaptação ao dispositivo. Controles de diálogo, por exemplo, ajustam-se ao tamanho da tela, facilitando o desenvolvimento, pois alivia o programador no planejamento de interfaces portáveis.

Outra característica que ajuda muito no quesito portabilidade, além de organizar melhor um programa, é o uso de arquivos de recursos. Assim, imagens, textos e até controles de menus podem ser modificados sem a necessidade de alteração no código do programa. Esta abordagem permite também que vários arquivos de recursos sejam criados para um mesmo aplicativo, com textos escritos em diferentes línguas ou com imagens para dispositivos específicos, por exemplo.

O uso de programação por eventos, gerenciada pelo próprio núcleo do BREW, também facilita o desenvolvimento de aplicativos de um modo geral, além de impedir a monopolização do processador por um aplicativo.

O BREW é uma tecnologia que nasceu no mundo dos dispositivos móveis e foi elaborada para contornar as restrições existentes, possuindo esquemas de acesso genérico a recursos dos aparelhos. O uso da linguagem de programação C/C++, que é compilada e roda sem a necessidade de uma máquina virtual, garante um melhor aproveitamento dos recursos limitados de processador e memória dos dispositivos.

Um problema relativo ao desenvolvimento de aplicativos BREW é que o mesmo está vinculado ao sistema operacional Windows e à ferramenta Visual Studio. Ainda assim, iniciar o desenvolvimento em BREW requer pouco investimento, pois o BREW SDK é distribuído livremente. O emulador, apesar de sua grande utilidade, não é perfeito e testes com aparelhos reais devem ser realizados. Esses testes são caros para pequenos desenvolvedores, pois é preciso ter acesso aos aparelhos que devem estar habilitados para testes. Outra desvantagem da tecnologia BREW é o seu vínculo com a empresa que a criou e que detém todo o poder sobre ela. Os desenvolvedores devem ser registrados na Qualcomm e seus aplicativos devem passar por testes de certificação. Operadoras que comercializam aplicativos BREW também são ligadas à empresa norte americana.

Nem todos os aparelhos são compatíveis com o BREW, portanto o alcance de uma aplicação já fica limitado.

Com relação ao estilo de programação, apesar de suportar programação C++, o que favoreceria o uso de orientação a objetos, parece mais natural o estilo de programação da linguagem C, que é estrutural. Não há como fugir da programação por eventos e o uso extensivo de ponteiros, conversões de tipo e o controle de alocação de memória por parte do programador são inevitáveis. Isto torna a programação mais complicada e suscetível a *bugs*.

Se, por um lado, o custo de desenvolvimento em BREW é considerável, o fato de existir uma infraestrutura pronta para levar os aplicativos até o usuário final e de existir um modelo de cobrança bem definido faz com que o potencial de retorno deste investimento seja alto, se a aplicação for bem aceita.

Conclui-se que a solução BREW preenche várias necessidades, tanto técnicas como de negócio e tem um potencial muito bom para crescer junto com o mercado de telefonia móvel. É de fácil acesso aos desenvolvedores, e, apesar de possuir um público

ainda relativamente pequeno no Brasil, está em forte expansão através dos esforços da Vivo que detém a maior fatia de mercado de telefonia celular do país, com mais de 42% de participação em setembro de 2004 (TELECO, 2004). À medida que as pessoas forem trocando seus aparelhos por novos modelos compatíveis com esta tecnologia, novos aplicativos serão desenvolvidos e o mercado deverá ficar bastante competitivo.

BIBLIOGRAFIA

ALCATEL. Disponível em <http://www.alcatel.com.br>. Aceso em jul. de 2004.

ANATEL. Disponível em <http://www.anatel.gov.br>. Acesso em jul. de 2004.

BENDAS, D.; MYLLYAHO, M. **Games as Part of Mobile Entertainment**. Oulu, Finlândia: Department of Information Processing Science, University of Oulu, 2002.

BRASIL. **Brasil Telecom GSM**. Disponível em <http://gsm.brasiltelecom.com.br>. Acesso em: dez. de 2004.

BUCKINGHAM, S. **GSM World – What is GPRS?** Disponível em <http://www.gsmworld.com/technology/gprs/intro.shtml> Acesso em jul. de 2004.

CDG. **CDMA Technology**. Disponível em <http://www.cdg.org/technology/index.asp>. Acesso em: jul. de 2004.

CLARO. **Site da Claro**. Disponível em <http://www.claro.com.br>. Acesso em: dez. de 2004.

DEITEL, H. M.; DEITEL, P. J.; NIETO, T. R. **Internet & World Wide Web: Como Programar**. 2ª ed.. Porto Alegre, RS: Bookman, 2003.

FOO, S. M.; LEE, W. M.; WATSON, K.; WUGOFSKI, T. **Beginning WAP, WML, and WMLScript**. Birmingham: Wrox, 2000.

GSMWORLD. Disponível em <http://www.gsmworld.com>. Acesso em jul. de 2004.

GUIASP. Disponível em <http://www.guiasp.com.br/guiasp/site/expediente/brew.cfm>. Acesso em ago. de 2004.

IVANOVA, A. **Mobile Tamagotchi Game**. MSc in Information Systems, 2003.

LEVY, R. **Mobilizing with .NET – An Introduction and case study**. DEVBUZZ. Disponível em <http://www.devbuzz.com/content/zinc_dotnet_going_mobile_pg1.asp>. Acesso em: nov. de 2004.

NASCIMENTO, D. G. **Sistema de Telemetria Aplicado à Medição de Energia Elétrica**. 56 f. Projeto de Diplomação (Engenharia da Computação) – Instituto de Informática, UFRGS, Porto Alegre. 2004.

NOLOGY. **Desenvolvimento de jogos eletrônicos**. Centro de Empreendimentos – Instituto de Informática da UFRGS, Porto Alegre. 2003.

PLANETA Celular: **Global System for Mobile Communications**. Disponível em <http://www.planetacelular.com.br/gsm.htm>. Acesso em jul. de 2004.

QUALCOMM. **About QUALCOMM – History / Key Milestones**. Disponível em <http://www.qualcomm.com/about/history.html>. Acesso em ago. de 2004a.

QUALCOMM. **Manual do Usuário BREW SDK**. 2002.

QUALCOMM. **Starting with BREW**. San Diego, CA – EUA: Morehouse Drive, 2004b.

ROSA, H. **GSM – Conceitos Básicos**. WirelessBR. Disponível em <http://www.wirelessbrasil.org/wirelessbr/colaboradores/agilent_gsm>. Acesso em jul. de 2004.

SCHAFER, S. **Learning WML – Wap Basics**. Developer. Disponível em <http://www.developer.com/ws/proto/article.php/1377381>. Acesso em: jul. de 2004.

STABELL, B.; SCHOUTEN, K. R. **The Story of XPilot**. ACM Crossroads, jan. 2001.

STALLINGS, W. **Data and computer communications**. 6th ed.. Upper Saddle River, NJ – EUA: Prentice Hall, 2000.

STROUSTRUP, B. **The C++ Programming Language**. 3rd ed.. Murray Hill: AT&T Labs, 1997.

TANENBAUM, A. S. **Redes de computadores**. 4ª ed.. Rio de Janeiro, RJ: Campus, 2003.

TECHNOLOGY : 2G – cdmaOne. **CDG**. Disponível em <http://www.cdg.org/technology/2g.asp>. Acesso em: jul. de 2004.

TELECO. **Operadoras de celular no Brasil**. Disponível em <http://www.teleco.com.br/opcelular.asp>. Acesso em: dez. de 2004.

TESLA. **Master of Lightning**. Disponível em <http://www.pbs.org/tesla>. Acesso em: dez. de 2004.

TIM. **TIM Brasil**. Disponível em <http://www.tim.com.br>. Acesso em: dez. de 2004.

TUDE, E. **AMPS/TDMA(IS-136)**. Teleco. Disponível em <http://www.teleco.com.br/tutoriais/tutorialtdma>. Acesso em: jul. de 2004.

TUDE, E. **CDMA(IS-95)**. Teleco. Disponível em <http://www.teleco.com.br/tutoriais/tutorialcdma>. Acesso em: jul. de 2004.

TUDE, E. **GSM**. Teleco. Disponível em <http://www.teleco.com.br/tutoriais/tutorialgsm>. Acesso em: jul. de 2004.

VIVO. **Downloads**. Disponível em <http://www.vivo.com.br/vivodownloads>. Acesso em: dez. de 2004.