UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

JOSÉ FELIPE RABER

UM AMBIENTE DE APRENDIZAGEM BASEADO NO JOGO STOP

TRABALHO DE CONCLUSÃO DE CURSO

PATO BRANCO
2011

JOSÉ FELIPE RABER

UM AMBIENTE DE APRENDIZAGEM BASEADO NO JOGO STOP

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina de Trabalho de Diplomação, do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, da Universidade Tecnológica Federal do Paraná, Campus Pato Branco, como requisito parcial para obtenção do título de Tecnólogo.

Orientador: Profº Esp. Tarlis Tortelli Portela

PATO BRANCO

2011

ATA Nº: **188**

DEFESA PÚBLICA DO TRABALHO DE DIPLOMAÇÃO DO ALUNO **JOSé FELIPE RABER**.

Às 13:45 hrs do dia 12 de julho de 2011, Bloco S da UTFPR, Campus Pato Branco, reuniu-se a banca avaliadora composta pelos professores Tarlis Tortelli Portela (Orientador), Edilson Pontarolo (Convidado) e Richardson Ribeiro (Convidado), para avaliar o Trabalho de Diplomação do aluno José Felipe Raber, matrícula 604950, sob o título **Um Ambiente de Aprendizagem Baseado no Jogo "Stop" Usando Tecnologias Móveis**; como requisito final para a conclusão da disciplina Trabalho de Diplomação do Curso Superior de Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, Coordenação de Informática. Após a apresentação o candidato foi entrevistado pela banca examinadora, e a palavra foi aberta ao público. Em seguida, a banca reuniu-se para deliberar considerando o trabalho **APROVADO**. Às 15:40 hrs foi encerrada a sessão.

Prof. Tarlis Tortelli Portela, Esp.
Orientador

Prof. Edilson Pontarolo, Dr.
Convidado

Prof. Richardson Ribeiro, Dr.
Convidado

Prof. Omero Francisco Bertol, M.Sc.
Coordenador do Trabalho de Diplomação

Prof. Edilson Pontarolo, Dr.
Coordenador do Curso

Dedico este trabalho a minha família que sempre me apoiou nas dificuldades e nas conquistas.

AGRADECIMENTOS

Aos meus familiares que sempre me apoiaram, aos professores que, através do seu conhecimento e sabedoria, deram condições para que eu me tornasse um profissional na área em que escolhi e a UTFPR, que possibilitou que esse sonho fosse realizado.

Já que voce tem de pensar de qualquer forma, pense grande.

Donald Trump

RESUMO

JOSÉ FELIPE RABER. **Um Jogo Multiplayer para Celulares: STOP.** 2011. Monografia de Trabalho de Conclusão de Curso. Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas. Universidade Federal Tecnológica Federal do Paraná, Campus Pato Branco. Pato Branco, 2011.

O trabalho desenvolvido tem como objetivo o desenvolvimento de um jogo *multiplayer* para celulares para auxiliar no aprendizado de línguas, inovando as técnicas atualmente utilizadas, sendo que a tecnologia móvel está em alta no mercado. O *STOP* é um jogo de palavras, no qual, a cada rodada é sorteadoa uma letra, e os jogadores devem escrever uma palavra que comece com essa letra para cada tema que deverá ser previamente escolhido, isso, no menor tempo possível. Para o desenvolvimento foi utilizado o ambiente Netbeans 6.7 e a linguagem de programação JAVA ME. Para a comunicação entre os dispositivos móveis foi utilizado a tecnologia *Bluetooth*. Como resultado da execução desse trabalho foi possível obter o jogo proposto.

Palavras-chave: jogo, stop, jogadores, netbeans, Java me, bluetooth, sistemas.

LISTA DE FIGURAS

LISTA DE CÓDIGOS

LISTA DE TABELAS

LISTA DE SIGLAS

API	Application Programming Interface
CDC	Connected Device Configuration
CLDC	Connected Limited Device Configuration
CPU	Central Processing Unit
DBMS	Database Management System
SGBD	Sistema Gerenciador de Banco de Dados
IDE	Integrated Development Environment
JCP	Java Community Proccess
JRE	Java Runtime Environment
JVM	Java Virtual Machine
MIDP	Mobile Information Device Profile
MIT	Massachusetts Institute of Technology
MMA-Latin	Mobile Marketing Association na América Latina
MTA	Mobile Telephony A
MVC	Model-View-Controller
PC	Personal Computer
PDA	Personal Digital Assistants
URL	Uniform Resource Locator
WTK	Wireless Toolkit

SUMÁRIO

1 INTRODUÇÃO

Com o aumento da utilização dos dispositivos móveis, aumentou também o número de sistemas desenvolvidos para essas plataformas.

Assim como a evolução do *hardware* de dispositivos móveis, as tecnologias, metodologias e ferramentas de desenvolvimento para esses dispositivos também evoluíram e, com isso, os sistemas operacionais, aplicativos, jogos evoluíram no mesmo ritmo e como consequência, o mercado para esses produtos aumentou também.

Segundo estudo da Flurry, site especializado em tecnologia, no mês de junho de 2010 os usuários passaram mais tempo utilizando aplicativos em dispositivos móveis do que navegando na web. (FLURRY, 2011)

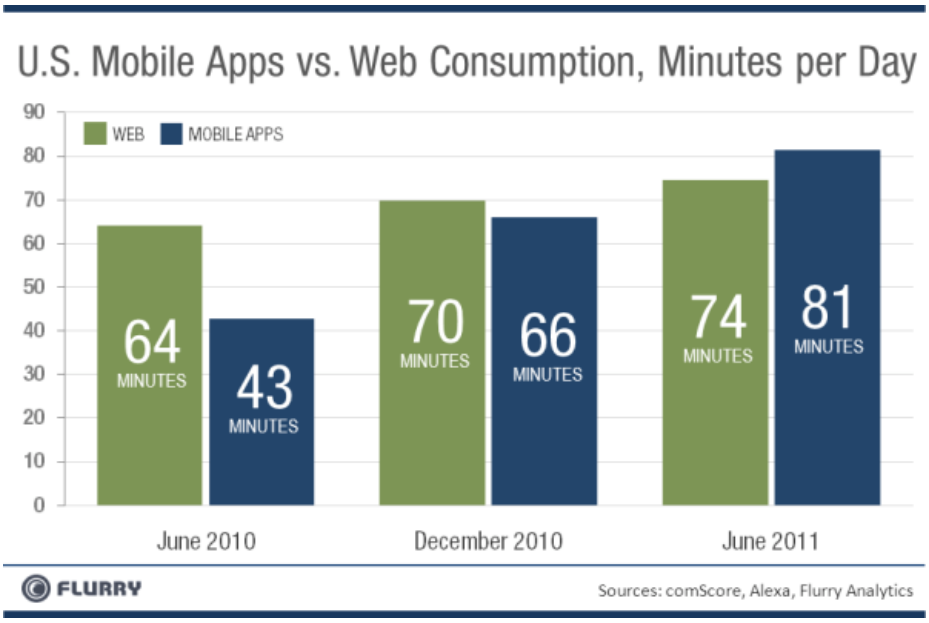

Figura 1 – Gráfico: utilização de aplicativos móveis x web

Fonte: www.flurry.com/Mobile-Apps-Put-the-Web-in-Their-Rear-view-Mirror

A Figura 1 mostra um gráfico que faz um comparativo do número médio de minutos por dia os consumidores gastam em aplicativos *mobile* em relação à web. Tendo em vista essa análise, esse trabalho propõe o desenvolvimento de um jogo para celulares.

1.1 Objetivos

O objetivo do trabalho proposto é desenvolver um jogo de cunho educativo, sendo seu enfoque auxiliar no aprendizado de línguas.

1.1.1 Objetivo Geral

Desenvolver um jogo *multiplayer* para celulares.

1.1.2 Objetivos Específicos

Os objetivos específicos consistem em:

- Gerar interação social através do entretenimento com o uso de tecnologias móveis;
- Auxiliar e maximizar os resultados no processo de aprendizado de escolas de línguas;
- Exemplificar o uso da linguagem de programação Java ME e da API JSR-82 no desenvolvimento de um aplicativo *mobile.*

1.3 Justificativa

O desenvolvimento de *softwares* para dispositivos móveis, sejam aplicativos ou games, tem aumentado praticamente em paralelo com o crescimento das tecnologias de *hardware*. Segundo Murta (2010), os celulares vão se tornar o principal meio de acesso a internet. Estudos da consultoria Gartner, empresa especializada em tecnologia, o faturamento relacionado a venda de jogos e aplicativos móveis, vai saltar de US$ 6,2 bilhões em 2010 para US$ 29,5 bilhões em 2013 e estima-se que até 80% dos aplicativos baixados da internet serão jogos.

No Brasil, as estatísticas seguem o mesmo rumo, sendo que num comparativo de junho de 2009 com junho 2010, o crescimento de vendas de *smartphones* foi de 69% e o acesso à banda larga móvel cresceu 227% no

mesmo período, de acordo com Murta (2010). “Estamos assistindo a uma substituição nos modelos de negócios”, afirma o presidente da Mobile Marketing Association na América Latina (MMA-Latam), Luiz Santucci Filho.

Esses dados exemplificam o novo comportamento social em relação as tecnologias móveis, em uma sociedade imediatista, aonde tudo precisa ser veloz e independente, aplicativos para dispositivos móveis desbravam um novo nicho de mercado produzido para uma nova geração.

No entorno desse cenário se encontra a computação móvel, que é uma área de estudo dos conceitos, ferramentas e tecnologias que envolvem o desenvolvimento de sistemas para dispositivos móveis. Segundo Nakamura (2003): “Computação móvel surge como uma quarta revolução na computação antecedida pelos grandes centros de processamento de dados da década de sessenta, o surgimento dos terminais nos anos setenta, e as redes de computadores na década de oitenta. Amplia o conceito tradicional de computação distribuída. Isso é possível graças à comunicação sem fio, o que elimina a necessidade do usuário manter-se conectado a uma estrutura em geral estática”.

Com isso em vista, o sistema que será desenvolvido nesse trabalho, tem como principal justificativa, suprir a demanda desse nicho de mercado que cresce exponencialmente.

1.4 Descrição do problema

O objeto deste trabalho trata-se de um jogo educativo para auxiliar o aprendizado de línguas, sendo utilizado dentro de salas de aula. Com isso em vista fica claro que todos os alunos e o professor devem possuir celulares para que possam jogar.

A interface do jogo para o professor, permitirá que o mesmo forneça para os alunos uma letra do alfabeto que será escolhido como base. A letra escolhida pelo professor (ou escolhida de forma aleatória pelo sistema), deverá aparecer na interface do jogo para os alunos. Na interface para os alunos também serão informados temas previamente determinados. Os alunos, assim

que puderem visualizar na tela do celular, a letra da rodada, deverão escrever via teclado, uma palavra de cada tema que comece com a letra em questão. A ideia é que cada aluno consiga terminar a rodada no menor tempo possível.

Ao terminarem de escrever as palavras, os alunos deverão enviar as respostas para o professor para que esse possa avaliar os resultados por algum critério definido por ele. O jogo, por *default*, fará essa avaliação pontuando os jogadores do mais rápido a enviar a resposta para o mais lento, sendo que o primeiro terá a maior pontuação e o último a menor pontuação.

1.5 Estrutura do Trabalho

Este texto está organizado em capítulos. O Capítulo 1 apresenta as considerações iniciais, os objetivos e a justificativa. O Capítulo 2 contém o referencial teórico do trabalho, fazendo uma abordagem do histórico dos telefones celulares e jogos e contextualização da computação móvel. No capítulo 3 são apresentados os materiais e métodos utilizados. Na sequência, o Capítulo 4 apresenta os resultados obtidos e por final as conclusões e referências bibliográficas.

2. REFERENCIAL TEÓRICO

Nesse tópico serão tratados assuntos relevantes para o desenvolvimento do jogo: computação móvel, banco de dados, RMS e bluetooth.

2.1 Jogos Eletrônicos

Os jogos eletrônicos são um fenômeno da cultura digital que estão sendo utilizados de variadas formas e nas mais diversas finalidades.

Segundo Mendes (2005), os jogos eletrônicos podem ser utilizados no ndotreinamento de habilidades motoras (por exemplo, dirigir carro, pilotar avião), reabilitação de pessoas que sofreram acidentes físicos, treinamentos médicos, instrumento pedagógico em escolas e obviamente, como artefato de entretenimento.

Apesar da falta de recursos desses jogos, os mesmos conseguiam divertir quem os jogava. Com isso em vista, as empresas do ramo começaram a investir muito dinheiro no desenvolvimento de jogos, sendo que hoje, a indústria de jogos movimenta mais de 18 bilhões de dólares na economia americana, segundo pesquisa realizada pelo iResearch. (SENA, 2011)

2.2 O surgimento dos games

Não existe um consenso entre os pesquisadores de quando foi criado o primeiro jogo eletrônico. Alguns historiadores acreditam que foi em 1958 por Willy Higinbotham, que consistia num jogo de tênis bastante simples, que era mostrado em um osciloscópio e processado por um computador analógico. Outros historiadores acreditam que primeiro jogo eletrônico foi o SpaceWar, criado pelo MIT (Massachusetts Institute of Technology). (MENDES, 2005)

“Bem depressa o jogo estava em toda a parte. Podia-se comprar uma versão especial para jogar na nossa própria televisão. O Pong era uma novidade, mas abriu caminho à chegada de outro jogo que tomaria o Japão de

assalto. Era o Space Invaders, o jogo que lançou a cultura dos jogos de vídeo". (TURKLE, 1989).

Com essa explosão dos jogos eletrônicos as empresas começaram investir mais nesse novo promissor mercado e diversas marcas surgiram, entre as mais famosas podemos citar: Atari, Nintendo, Sega, Play Stations e Xbox.

Podemos perceber dessa forma que os games da atualidade têm um potencial ilimitado, pois estimulam as crianças e adolescentes para a criatividade e imaginação, desenvolvendo habilidades cognitivas e motoras, além de interagir com um número cada vez maior de participantes nos jogos, aprendendo a tomar decisões e criar estratégias (Sena, 2011).

2.3 Games e aprendizagem

Existe um preconceito crônico em relação a jogos eletrônicos. A imprensa, educadores e psicólogos muitas vezes taxam os jogos como estimuladores da violência e viciantes.

Porém essa visão pessimista a mudar, na realidade, os jogos eletrônicos são grandes estimuladores da mente e proporcionam um engajamento cognitivo do sujeito muito maior do que o que ocorria há 30 anos atrás por meio dessas mesmas mídias. (Sena, 2011)

Pesquisas do MIT com o projeto Games-to-Teach que recebem apoio financeiro da Microsoft, começam a descobrir nos games um instrumento de aprendizagem que além de ser estimulante é divertido e motivador, que produzem capacidades e competências, revelando sua importância para a educação, a aprendizagem e o desenvolvimento cognitivo.

Segundo Sena (2011), os *games* promovem a construção ou reorganização de funções cognitivas como a memória, a atenção e a intelecção pelo qual o sujeito conhece o objeto. Essa interação com os jogos eletrônicos permite que essas funções cognitivas sejam intensificadas, favorecendo as crianças, adolescentes e adultos a descoberta de novas formas de conhecimento, que hoje ocorrem por meio da simulação de novos mundos através de games como *Age of Empires*, *Line Age 2*, *Sim City 4* e outros.

Diante dessa análise, vemos que os benefícios dos *games* para a educação são inúmeros, entre eles: desenvolvimento de habilidades motoras e de visão, capacidade investigativa, entendimento de problemas simples e complexos, capacidade do cumprimento de objetivos.

2.4 Computação Móvel

A computação móvel trata dos conceitos, ferramentas e tecnologias que permitem o desenvolvimento de aplicativos para dispositivos móveis. Segundo Nakamura (2003), é um novo paradigma computacional que permite aos usuários ter acesso a serviços independentemente de sua localização, podendo inclusive, estar em movimento.

A partir dos anos 90, pode-se notar um grande crescimento no desenvolvimento de tecnologias para comunicação celular móvel, comunicação via satélite e redes locais sem fio. A popularização dessas tecnologias tem permitido o acesso a informações remotas onde quer que se esteja, abrindo um leque muito grande de facilidades, aplicações e serviços para os usuários através da mobilidade. Esse ambiente propicia a criação do conceito de computação móvel. (NAKAMURA, 2003).

Segundo Valentino (2005), mobilidade pode ser definida como a capacidade de poder se deslocar ou ser deslocado facilmente. No contexto da computação móvel, mobilidade se refere ao uso pelas pessoas de dispositivos portáteis funcionalmente poderosos que oferecem a capacidade de realizar facilmente um conjunto de funções de aplicação, sendo também capazes de conectar-se, obter dados e fornecê-los a outros usuários, aplicações e sistemas.

2.5 RMS (Record Managment System)

Em sistemas desenvolvidos para plataforma Web ou para *desktops* existem vários SGDBs que podem ser utilizados para persistência de dados, como os que foram citados no item 2.1, assim como para o armazenamento de

dados, entre eles: CD-ROM, discos rígidos locais, unidades de rede.

Para dispositivos móveis, esse processo é feito utilizando o RMS (Record Managment System, sistema de gerenciamento de registros), um ambiente de armazenamento persistente dentro do MIDP (Mobile Information Device Profile, perfil de dispositivos de informação móvel). Uma abordagem mais detalhada sobre o MIDP será feita no item 3.12.

Segundo Muchow (2005), como uma alternativa ao uso de um sistema de arquivo, o RMS utiliza memória não-volátil para armazenar informações. Esse banco de dados orientado a registros, frequentemente referido como arquivo puro, pode ser imaginado como uma série de fileiras em uma tabela, com um identificador exclusivo para cada fileira, conforme Tabela 1.

ID de registro	Dados
1	Array de bytes
2	Array de bytes
3	Array de bytes
...	...

Tabela 1 – Tabela que representa o funcionamento do banco de dados RMS
Fonte: MUCHOW, 2005

Na Tabela 1, o ‘ID de registro’ corresponde a um registro no RMS. É um número inteiro que funciona como um identificador do registro, semelhante a uma chave primária nos banco de dados relacionais. O campo ‘Dados’ consiste em um *array de bytes* para armazenar os dados do registro.

Atualmente existem alguns *frameworks* que auxiliam na persistência de dados em dispositivos móveis, entre eles o Floggy, que será abordado na seção de materiais desse trabalho.

2.6 Bluetooth

Bluetooth é uma tecnologia que nos últimos anos vem se disseminando no mercado devido sua facilidade e baixo custo de implantação.

“O Bluetooth é uma tecnologia que permite uma comunicação simples, rápida, segura e barata entre computadores, *smartphones*, telefones celulares, mouses, teclados, fones de ouvido, impressoras e outros dispositivos, utilizando ondas de rádio no lugar de cabos. Assim, é possível fazer com que dois ou mais dispositivos comecem a trocar informações com uma simples aproximação entre eles”. (ALECRIM, 2011)

A tecnologia *Bluetooth* elimina a necessidade de cabos de interconexão e possibilita estabelecer redes entre dispositivos, possibilitando a o compartilhamento de informações. (YOUNG, 2005)

Devido ao baixo custo, facilidade de implantação essa tecnologia vem se popularizando nos últimos anos, com o principal objetivo de substituir cabos em casas e em pequenas empresas.

2.7 Banco de Dados

Um sistema de banco de dados é constituído por um conjunto de dados associados a um conjunto de programas para acesso a esses dados. O conjunto de dados, comumente chamado banco de dados, contém informações sobre o contexto no qual o sistema computacional que está utilizando esse banco de dados está inserido. (KORTH, 1999).

Em outras palavras, banco de dados, é conjunto de registros organizados e que podem ser manipulados com o principal objetivo de gerar informações, relatórios consistentes e relevantes.

Segundo CHU (1983), um banco de dados corresponde a uma reunião de arquivos de dados de toda organização em algum tipo de armazenamento, sendo manipulado por um conjunto de programas.

Esse conjunto de programas, atualmente são chamados de Sistemas Gerenciadores de Banco de Dados (SGDB) - do inglês DBMS (*Data Base Management System*). Um SGDB é um programa que tem como principal objetivo gerenciar o banco de dados, ou seja, gerenciar o acesso, manipulação e organização dos dados. Dentre os principais, podemos citar: PostgreSQL, Firebird, DB2, MySQL, Oracle, SqlServer, Access.

Existem vários modelos de banco de dados, entre eles:

- Modelo Plano: consiste de matrizes simples, bidimensionais, compostas por elementos de dados.
- Modelo em rede: neste modelo as entidades se representam como nós e suas relações são as linhas que os unem.
- Modelo relacionais: o modelo relacional é o mais utilizado atualmente. Consiste em tabelas bidimensionais, que são os elementos principais deste modelo e representam os dados e suas relações.
- Modelo orientado a objeto**:** as informações são armazenadas em forma de objetos.

3 MATERIAIS E MÉTODOS

Abaixo será feita uma abordagem das metodologias, ferramentas e tecnologias que serão utilizadas no desenvolvimento do projeto.

3.1 Materiais

Nesse item serão descritos os materiais que serão utilizados para o desenvolvimento do jogo: Netbeans, WirelessToolkit, JavaME, API SJR-82, BlueCove e Floggy.

3.1.1 Netbeans

O Netbeans IDE é uma das ferramentas para Java mais utilizadas pelos desenvolvedores de *software*. Ele fornece suporte para várias linguagens (PHP, JavaFX, C / C + +, JavaScript, etc) e *frameworks*.

É um projeto *open-source* dedicado ao fornecimento de produtos de *softwares* que atendam às necessidades de desenvolvedores, usuários e as empresas. Em Junho de 2000, foi aberto o código do NetBeans pela Sun Microsystems, que se manteve o patrocinador do projeto até janeiro de 2010, atualmente é uma subsidiária da Oracle.

Os dois produtos de base, o NetBeans IDE e o NetBeans Platform, são livres para uso comercial e não comercial. O código-fonte está disponível para todos poderem utilizar, conforme termos de uso.

O projeto do NetBeans também é uma comunidade onde pessoas de todo o mundo podem fazer perguntas, dar conselhos, fazer várias contribuições e, finalmente, compartilhar do sucesso dos produtos. Nas listas de discussões na internet, podem-se encontrar tópicos de estudantes, desenvolvedores e pessoas que querem expandir suas habilidades. (O QUE É O NETBEANS, 2011)

3.1.2 Java ME

O Java ME, também chamado de J2ME, é uma linguagem de programação destinada a dispositivos com poucos recursos de processamento, armazenamento e memória.

Segundo Muchow (2005), é uma linguagem destinada aos dispositivos consumidores com poder limitado. Muitos desses dispositivos (por exemplo, celular ou pager) não têm opção de *download* e *software* de instalação, além daquele que foi configurado durante o processo de fabricação. Com a introdução do J2ME, os dispositivos “micros” não precisam mais ter natureza “estática”. Exatamente como um navegador Web fazendo o *download* de *applets* Java, uma implementação do J2ME em um dispositivo permite a opção de nevegar, fazer *download* e instalar aplicativos Java e conteúdo.

Para suportar uma ampla variedade de produtos que se encaixam dentro do escopo do J2ME, a Sun (empresa que desenvolve o J2ME, recentemente adquirida pela Oracle) introduziu a Configuração e os Perfis. Uma Configuração define uma plataforma Java para um ampla variedade de dispositivos. Ela está intimamente vinculada a uma máquina virtual Java (JMV, Java Virtual Machine). Na verdade, uma configuração define os recursos da linguagem Java e as bibliotecas Java básicas da JVM para essa configuração em particular. (MUCHOW, 2005)

Existem duas configurações atualmente definidas: CDC e CLDC. A configuração CDC necessita no mínimo *512 kilobytes* de memória para executar o Java, *256 kilobytes* de memória para alocação de memória em tempo real e possui conectividade de rede e a configuração CLDC necessita de *128 kilobytes* de memória para executar o Java, *32 kilobytes* para alocação de memória em tempo de execução, possui interface restrita com o usuário, baixo poder e conectividade de rede. (MUCHOW, 2005)

Um perfil é uma extensão de uma configuração. Ele fornece as bibliotecas para um desenvolvedor escrever aplicativos para um tipo em particular de dispositivos. Por exemplo, o MIDP (Mobile Information Device Profile, perfil de dispositivos de informação móvel) define APIs para

componentes, entrada e tratamento de eventos de interface com o usuário, armazenamento persistente, interligação em rede e cronômetros.

3.1.3 Wireless ToolKit

O Wireless Toolkit, também conhecido por WTK, é uma ferramenta para desenvolvimento de aplicativos para celular fornecida pela Sun Microsystems. Essa ferramenta fornece um ambiente para compilação e execução de aplicativos, entretanto a mesma não possui um editor de código.

O WTK também pode ser utilizado como um emulador de celular para *desktop*, possibilitando personalizar configurações, como a velocidade da CPU e a velocidade de acesso a rede, deixando os testes no emulador os mais próximos possíveis da realidade dos celulares.

O *download* do WTK pode ser feito pela url http://java.sun.com/products/sjwtoolkit/download.html. Nessa página basta clicar no link *download*, selecionando a plataforma e a linguagem para a ferramenta. A instalação do ambiente não requer nenhuma configuração avançada. O MobilityPack do Netbeans possui o WTK integrado. (WIRELESS TOOLKIT, 2011)

3.1.4 API JSR 82

JSR 82 é uma especificação definida pelo Java Community Process para fornecer um padrão para desenvolvimento de aplicações Bluetooth em Java. É um padrão aberto e não-proprietário para desenvolvimento de aplicativos Bluetooth, que esconde a complexidade da pilha de protocolos *Bluetooth*, expondo um conjunto simples de JAVA API (JSR-82, 2011).

Utilizando a API JSR 82, é possível:

- Gerenciar as configurações de dispositivo local Bluetooth;
- Descobrir outros dispositivos na vizinhança;
- Procurar dispositivos bluetooth nos dispositivos descobertos;
- Conectar-se a qualquer um desses serviços e comunicar-se com eles;

- Registrar um serviço Bluetooth no dispositivo local, para que outros dispositivos possam se conectar a ele;
- Gerenciar e controlar as conexões de comunicação;
- Fornecer a segurança para todas as opções acima.

3.1.5 Floggy

Floggy é um *framework open-source* de persistência para aplicações J2ME/MIDP. O objetivo principal desta ferramenta é abstrair os detalhes de persistência de dados no banco de dados RMS, reduzindo o tempo de desenvolvimento e esforço de manutenção.

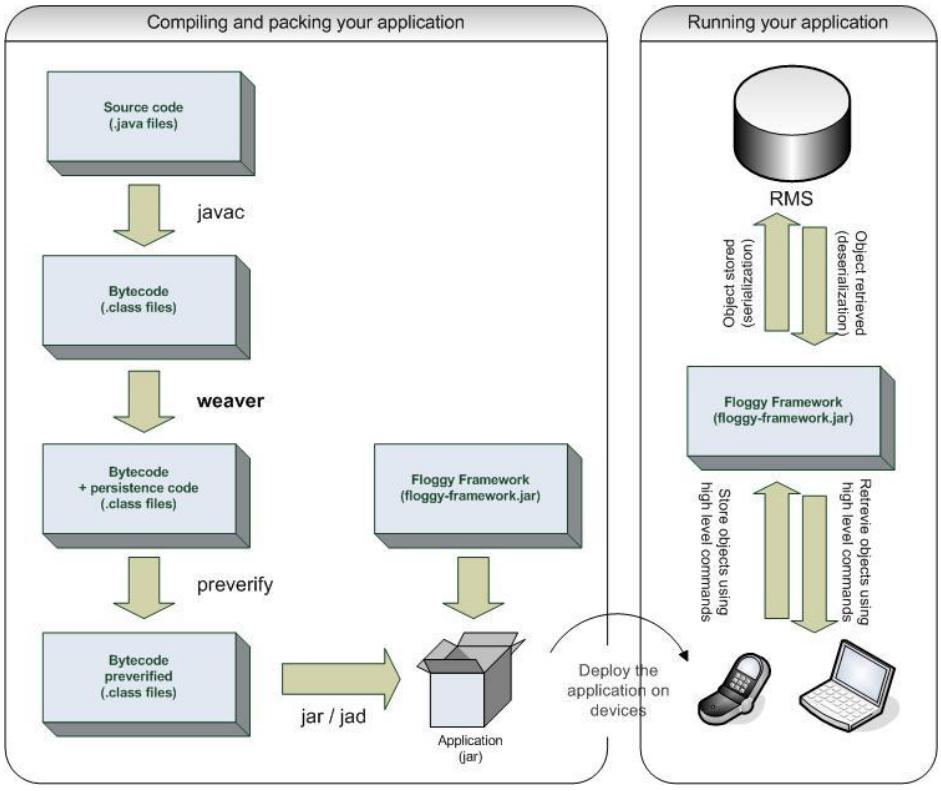

Figura 2 – Funcionamento do Floggy

Fonte: http://floggy.sourceforge.net/

A Figura 2 descreve como funciona o Floggy: o quadro a esquerda mostra como funciona a compilação e “empacotamento” da aplicação, que precisa ter o arquivo “floggy-framework.jar”, que é a biblioteca de classes do

framework Floggy, adicionado ao arquivo “.jar”. Arquivo “.jar” é semelhante ao um arquivo executável na plataforma desktop. O retângulo a esquerda mostra como funciona a interação do Floggy com o banco de dados RMS, onde os objetos são serializados para que possam ser armazenados no banco de dados num *array* de *bytes*.

Todas as operações, como salvar, carregar, excluir e encontrar objetos são feitos através de uma única classe chamada *PersistableManager*. As classes que serão persistidas devem implementar a interface *Persistable.* A Listagem 1 demonstra a classe *Persistent*, que é uma das classes que fazem parte da modelagem do jogo.

Listagem 1 – Exemplo de uma classe de persistência utilizando o framework Floggy

```
package model;

import net.sourceforge.floggy.persistence.Persistable;

public class Persistent implements Persistable {

    private int id;

    public Persistent() {
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }
}
```

3.2 Métodos

O jogo foi desenvolvido utilizando ciclos iterativos de análise e implementação, ou seja, como o trabalho se trata de um jogo simples, e não demanda de uma grande análise, a ideia é escolher pequenas partes do todo, fazer uma pequena análise de requisitos, implementar e testar. Ao terminar

esse ciclo, fazê-lo novamente num processo iterativo e, por fim, juntar essas pequenas partes para construir o produto final. Com isso, é possível ter uma visão parcial do projeto, utilizá-lo, testá-lo e assim ter mais subsídios para futuras modificações.

3.2.1 Persistência dos Dados

O banco de dados utilizado no projeto foi o RMS. Para fazer a interação com o banco de dados foi utilizado o framework Floggy, que abstrai todo o processo de configuração e operação do RMS.

A utilização do Floggy se assemelha ao uso de um banco de dados orientado a objetos. Todas as operações com o banco de dados são feitas através de métodos definidos na classe *PersistableManager* desse *framework*, entre eles:

- *save(Persistable arg0):* salva um objeto no banco de dados RMS.
- *delete(Persistable arg0):* exclui um objeto do banco de dados RMS.
- *find(Class arg0):* lista todos os objetos salvos da classe que for passada como parâmetro.

Todas as operações relacionadas ao banco de dados são feitas através de métodos definidos na classe *GenericDao* (pacote *dao*). Na Listagem 2 é possível verificar a operação de inserção de dados.

Listagem 2 – Inserção no banco de dados utilizando o Floggy

```
public void insert(Persistent persistent) {
    try {
        persistent.setId(PERSISTENT_MANAGER.save(persistent));
        edit(persistent);

    } catch (FloggyException ex) {
        ex.printStackTrace();
    }
}
```

Na Listagem 2, o método *insert()*, responsável por salvar um objeto no banco de dados recebe como parâmetro um objeto da classe *Persistent,* que é a classe pai das outras classes do jogo, conforme mostra o diagrama de classes na Figura 3.

3.2.2 Diagrama de classes

O diagrama de classes consiste nas classes criadas na modelagem do jogo. Por se tratar de um jogo simples, o diagrama de classes é compacto e consiste de apenas três classes: *Persistent, Game* e *User.* O diagrama de classes do jogo é demonstrado na Figura 3.

Cada retângulo do diagrama representa uma classe, onde os itens com o sinal de '-' representam os atributos da classe e os itens com o sinal de '+' representam os métodos da classe. Os métodos de cada classe foram representados pelo *getterMethods(),* que se refere aos métodos de acesso as atributos e pelo *setterMethods()*, que se refere aos métodos de atribuição de valores aos atributos da classe.

As classes *Game* e *User* estão ligadas a classe Persistent por uma seta com um triângulo na ponta. Isso representa que essas duas classes são filhas da classe *Persistent*, ou seja, uma instancia da classe *Game* ou da *classe User,* pode acessar os atributos e métodos da classes *Persistent* desde que os mesmos sejam públicos ou protegidos, ou seja, declarados na classe com a definição de acesso *public* ou *protected.*

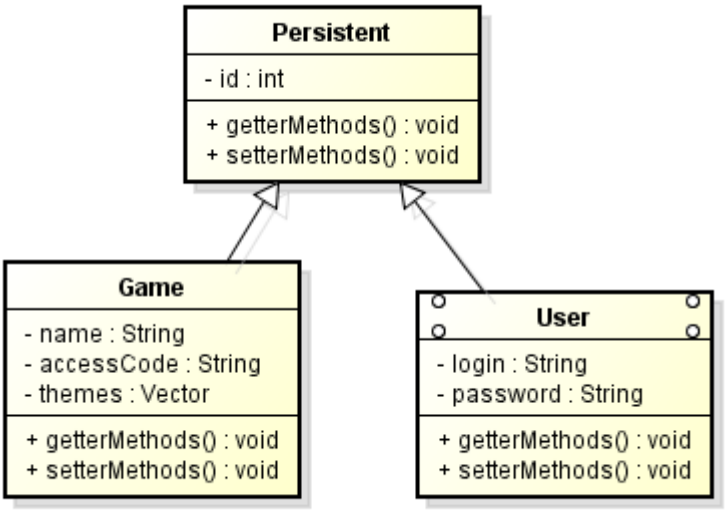

Figura 3 – Diagrama de classes da modelagem do jogo Stop

Para que seja possível manipular os dados com o Floggy, as classes que representam as tabelas do jogo, devem implementar a interface *Persistable*.

3.2.3 Diagrama de casos de uso

O diagrama de casos de uso demonstra as principais funcionalidades de um sistema do ponto de vista do usuário. Um diagrama de casos de uso é divido basicamente em três partes: atores, casos de uso e relacionamento, onde um ator representa o usuário do sistema, seja ele um ator um outro sistema computacional, o caso de uso define uma funcionalidade do sistema e o relacionamento define qual é a relação entre o ator e um caso de uso.

No contexto do jogo, o diagrama de casos de uso possui dois atores, o professor e os alunos. O relacionamento dos atores com os casos de uso do jogo é demonstrado na Figura 4.

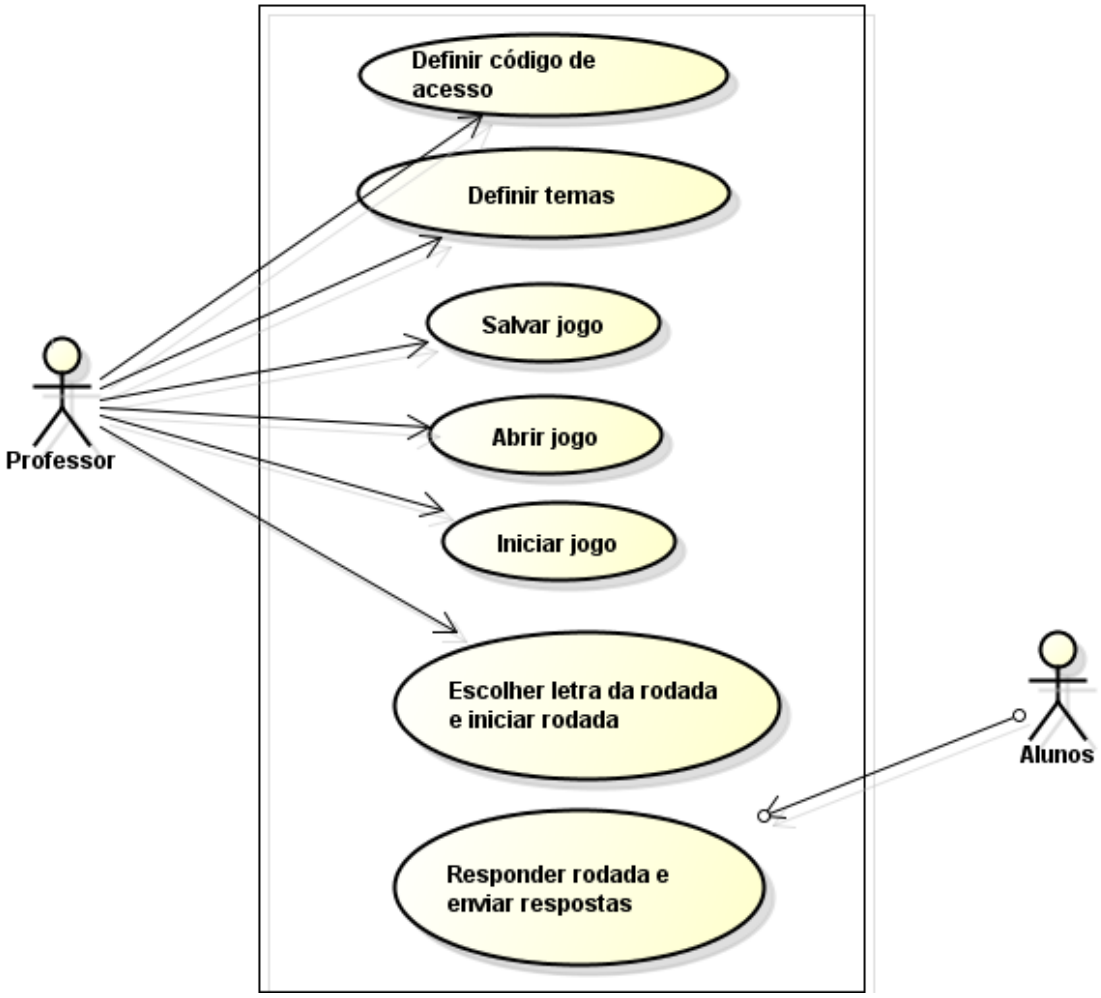

Figura 4 – Diagrama de casos de uso do jogo Stop

3.2.4 Requisitos do jogo

Como já foi visto no item 3.1.2, uma configuração define os recursos da linguagem Java e as bibliotecas Java básicas da JVM para essa configuração em particular e um perfil é uma extensão de uma configuração. (MUCHOW, 2005). O jogo STOP foi desenvolvido utilizando a configuração CLDC 1.1 e MIDP 1.1.

Para que o jogo STOP possa ser executado, o celular deve possuir suporte a plataforma Java, ou seja, deve possuir a máquina virtual Java para dispositivos móveis e suporte as a configuração e perfil definidas. Os modelos de celular utiliza*dos para teste em tempo de desenvolvimento do jogo foram o Nokia C5 e o Nokia 5530.

RESULTADOS

Na sequência serão apresentados os resultados do projeto, explicando os principais elementos do trabalho de forma sintetizada e exemplificando o uso das ferramentas, métodos e tecnologias escolhidas.

4.1 Organização dos pacotes (packages)

As *packages* foram divididas em: *midlet*, *model*, *resources*, *utils* e *dao*. A forma de organização das *packages* foi uma adaptação do padrão de arquitetura de *software* chamado MVC (*Model-View-Controller*), que tem como objetivo separar a lógica de negócio da lógica de apresentação, permitindo o desenvolvimento, teste e manutenção isolado de ambos. A Figura 5 mostra a organização das *packages* do jogo.

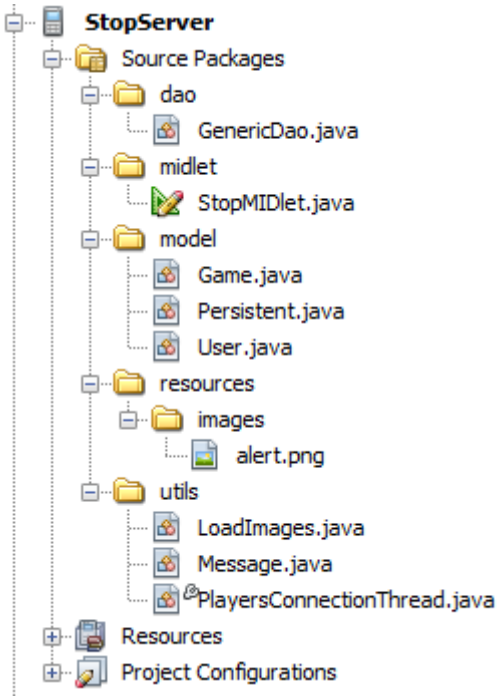

Figura 5 – Exemplo da utilização de pacotes (packages)

No pacote *midlet* se encontra o *midlet* principal da aplicação chamado *StopMidlet* para o lado servidor e *StopClient* para o lado cliente. No pacote *model,* se encontram as classes referentes a modelagem do sistema. O jogo

possui três classes: *Game*, *User* e *Persistent*. Essa última é a classe pai de todas as outras, e possui como atributo o código que referencia os objetos no banco de dados.

Já no pacote *resources,* se encontram as fontes do sistema, que no caso do jogo, são as imagens utilizadas no mesmo. No pacote *utils*, se encontram algumas classes de uso geral do jogo: *PlayerConnectionThread, LoadImages, Message.* A classe *LoadImages* tem a função de carregar as imagens que serão utilizadas no jogo e a Message tem função de fornecer mensagens que serão utilizadas no jogo. A classe *PlayerConnectionThread* será descrita no item 4.2.1.2.4.

O pacote *dao* (do inglês, *Data Access Object*), é um padrão para persistência de dados que permite separar regras de negócio das regras de acesso a banco de dados, obtendo uma melhor organização dos dados.

4.2 *StopServer* e *StopClient*

Como o jogo é *multiplayer*, ou seja, é utilizado por dois ou mais jogadores ao mesmo tempo, foram desenvolvidos dois projetos no Netbeans: o *StopServer*, que se trata do “lado servidor” do jogo, e o *StopClient,* que se trata do “lado cliente”. Abaixo segue uma análise do lado servidor e do lado cliente do jogo, respectivamente.

4.2.1 Lado servidor (*StopServer*)

O objetivo do jogo, é que seja jogado dentro de uma sala de aula, sendo que é um jogo educativo para melhorar o aprendizado de línguas. Com isso em vista fica claro que a interface do jogo para o professor é o lado servidor e a interface para os alunos é lado cliente, conforme ilustra a Figura 6.

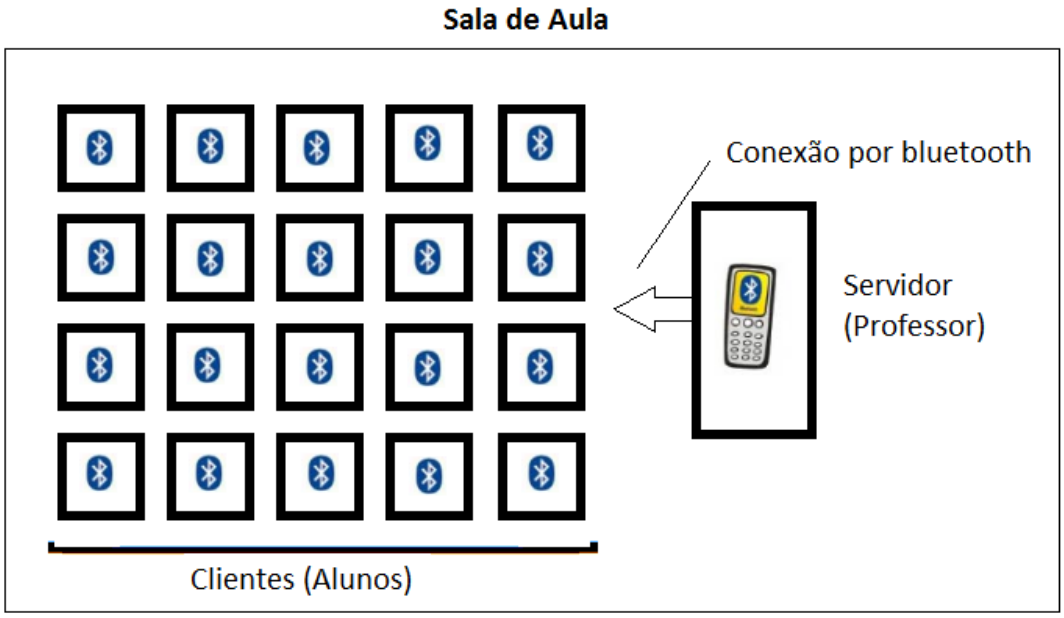

Figura 6 – Exemplo do modelo de negócio do jogo Stop

Cada uma das telas possui comandos que executam determinadas ações no jogo. Esses comandos são variáveis da classe *Command.* Para que funcionem, é necessário que o *midlet* implemente a interface *CommandListener*.

A Listagem 3 exemplifica a utilização da interface *CommandListener*. Quando o comando *cmEnterLogin*, que é a opção *Entrar* da tela de login do jogo, o código demonstrado na listagem é executado. Se a tela atual for a tela de login e a opção escolhida pelo usuário for referente ao comando de Entrar, caso o login e senha forem, respectivamente, *admin* e *adg,* o jogo será direcionado para a tela de administrador, onde é possível listar, inserir e excluir usuários, caso contrário, será feito uma verificação no banco de dados da existência do usuário e senha digitados, e caso validados, o jogo será direcionado para o menu principal.

Listagem 3 – Implementação do método *commandAction* da interface *CommandListener*

```
209         } else if (displayable == fmLogin) {
210             if (command == cmEnterLogin) {
211 
212                 if(getTxtLogin().getString().equals("admin")) {
213                     if(getTxtPassword().getString().equals("adg")) {
214                         switchDisplayable(null, getLsAdmin());
215                     } else {
216                         switchDisplayable(new Alert("Stop", "Login ou senha incorretos!",
217                                 getImages().getImgAlert(), AlertType.WARNING), getFmLogin());
218                     }
219                 } else {
220                     if(getDao().verifyUser(getTxtLogin().getString(),
221                             getTxtPassword().getString())) {
222                         switchDisplayable(null, getLsGameMenu());
223                     } else {
224                         switchDisplayable(new Alert("Stop", "Login ou senha incorretos!",
225                                 getImages().getImgAlert(), AlertType.INFO), getFmLogin());
226                     }
227 
228                     switchDisplayable(null, getLsGameMenu());
229                 }
230 
231             } else if (command == cmExitLogin) {
```

4.2.1.1 Mobile Device (Midlet principal)

Uma aplicação MIDP (*Mobile Information Device Profile*) é chamada de Midlet, ou seja, uma aplicação desenvolvida para celular utilizando CLDC/MIDP possui esse nome. O *software* de gerenciamento da aplicação (AMS) do dispositivo interage diretamente com o Midlet com os métodos de criar, iniciar, pausar e destruir o aplicativo.

A vida de um MIDlet começa quando ele é instanciado pelo AMS. Ele inicialmente entra no estado pausado. O AMS inicia o MIDlet chamando o método construtor público da classe. Se uma exceção ocorrer no construtor, o MIDlet é colocado no estado destruído e é descartado imediatamente.

Na sequência, o AMS chama o método *startApp*() do MIDlet, deixando-o no estado ativo. O MIDlet entra no estado destruído quando o AMS chama o método *destroyApp*(). Este estado também é atingido quando o método *notifyDestroyed*() retorna com sucesso para a aplicação. O MIDlet entra no estado destruído somente uma vez no seu tempo de vida. Já o estado de pausa é atingido quando o aparelho celular recebe um evento externo enquanto o aplicativo está sendo executado. Podemos citar como exemplo o recebimento de uma ligação ou de uma mensagem SMS.

Quando o método *startApp()* é chamado, o jogo vai para a tela de controle de acesso (*fmLogin*), que por sua vez será direcionada para a tela de administrador do sistema (*lsAdmin*) ou para o menu principal no jogo (*lsGameMenu*). Para entrar na tela de administrador, onde é possível fazer a listagem, inserção, edição e exclusão de usuários, é necessário digitar um *login* e senha previamente determinados pelo desenvolvedor. Na Figura 7 é possível observar a tela de controle de acesso, tela de administrador e a menu principal do sistema.

Figura 7 – Tela de controle de acesso, administrador e menu principal

4.2.1.2 Fluxo do jogo (telas)

Abaixo será feito uma abordagem de cada uma das telas do lado servidor do jogo, explicando o fluxo do jogo.

4.2.1.2.1 Menu inicial (*lsGameMenu*)

Na tela que apresenta o menu principal do jogo é possível entrar nas telas para o código de acesso, definir temas, salvar jogo, abrir jogo e iniciar o jogo. Ao acionar o comando *Selecionar*, o jogo abrirá a tela da opção que está selecionada, por exemplo, a tela de *Código de Acesso*.

4.2.1.2.2 Código de Acesso

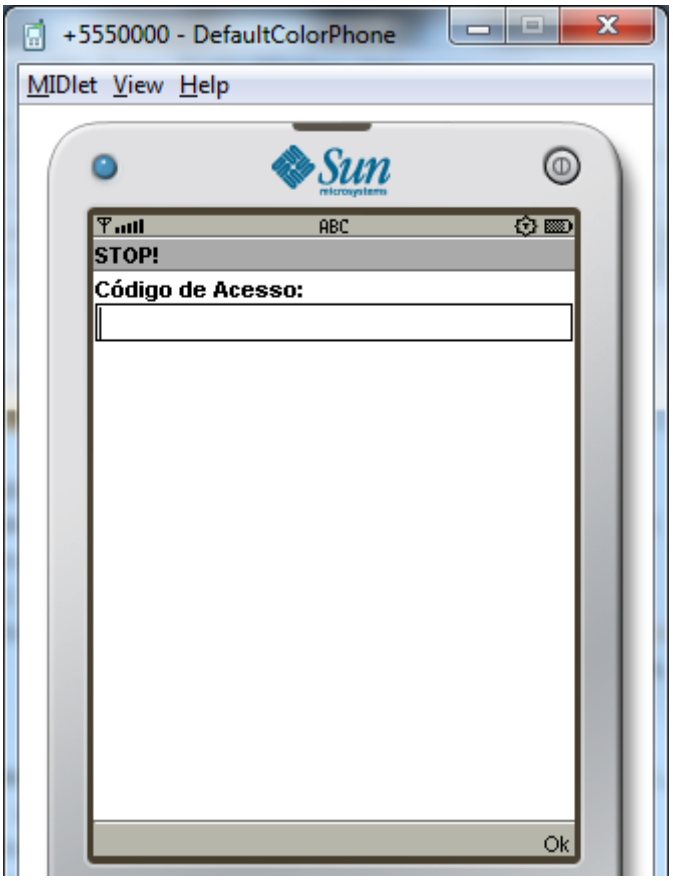

Figura 8 – Tela de definição do código de acesso

A Figura 8 mostra a tela de Código de Acesso. É necessário que o usuário do servidor defina um código de acesso para que os clientes (jogadores) que forem jogar digitem esse código e possam iniciar o jogo; se o código for inválido, o servidor enviará uma mensagem para o aluno informando, com isso, será possível fazer um jogo em uma rede fechada, sem interferência de outros.

4.2.1.2.3 Temas

A função dessa tela, é que o usuário do servidor (professor) defina quais temas serão utilizados. Por padrão o jogo inicia com os seguintes temas: nome, cidade, estado, país, fruta, profissão, cor, objeto. É possível adicionar e

remover temas como mostra a Figura 9. Também é possível, a partir dos temas adicionados, definir quais serão usados apenas marcando o *checkbox* do tema em questão.

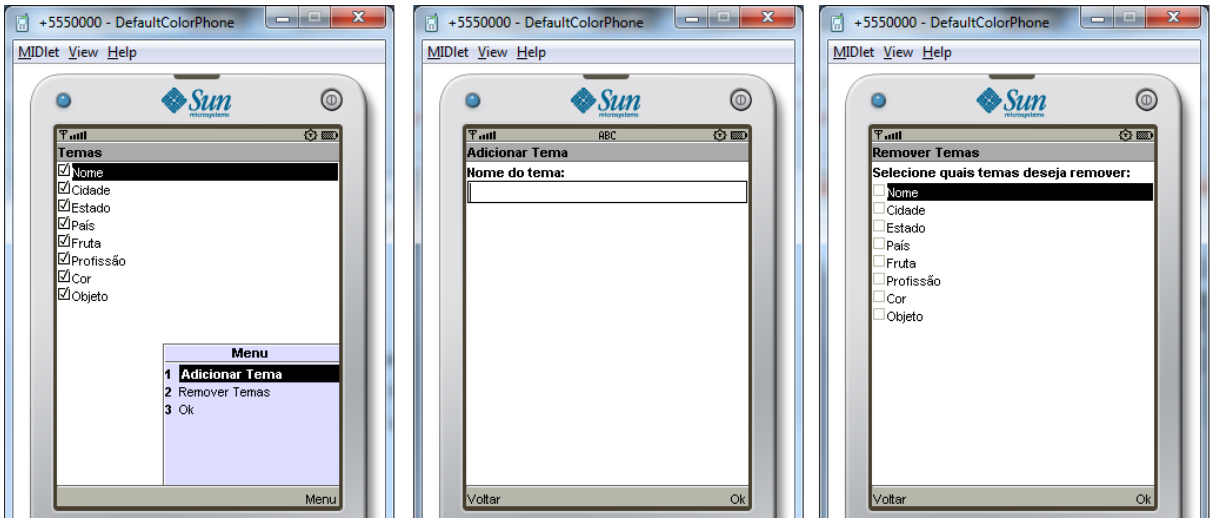

Figura 9 – Tela de listagem, inclusão e exclusão de temas

4.2.1.2.4 Salvar e abrir jogos salvos

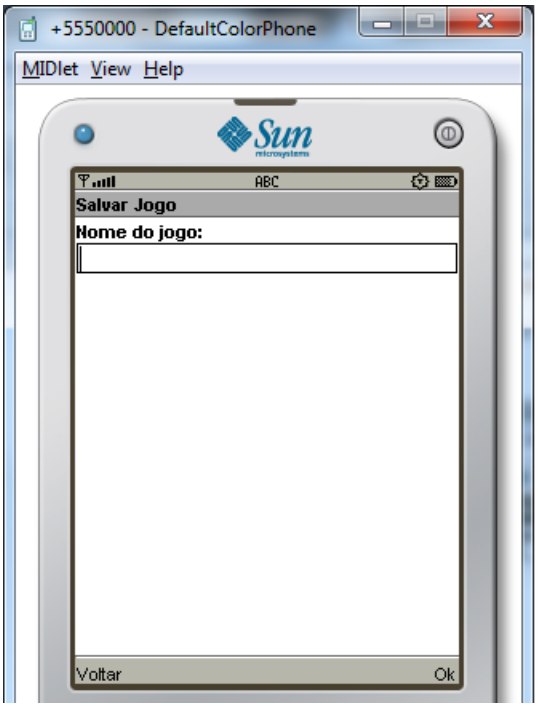

Figura 10 – Tela para salvar jogo

Ao escolher a opção *Salvar Jogo* no menu principal do jogo, a tela da Figura 10 é mostrada para o usuário. É necessário digitar um nome para o jogo e ao acionar o comando *Ok*, será salvo o jogo no banco de dados com o nome do jogo, código de acesso e temas que foram definidos. Para recuperar um jogo salvo, o usuário deverá acessar a opção do menu Abrir Jogo, como demonstra a Figura 11.

Figura 11 – Tela para abrir jogos salvos

4.2.1.2.5 Iniciar Jogo

Ao acessar a opção *Iniciar Jogo* do menu principal, o jogo executará o código mostrado na Listagem 4.

Listagem 4 – Código executado ao iniciar o jogo

```
case 5:
    if(!getTxtAccess().getString().equals("")) {
        PlayersConnectionThread player = new PlayersConnectionThread(this);

        player.start();

        getPlayersThreads().addElement(player);

        switchDisplayable(null, getLsGame());
    } else {
        switchDisplayable(new Alert("Stop", "É necessário definir um código de acesso!",
            getImages().getImgAlert(), AlertType.WARNING), getLsGameMenu());
    }

    break;
```

Em resumo, se o usuário não definiu um código de acesso, o jogo informará o mesmo disso, caso contrário, uma nova instância da classe *PlayerConnectionThread* será adicionada a uma lista, e será apresentada a tela conforme a Figura 12. A classe *PlayerConnectionThread* é uma extensão da classe Thread que implementa a interface Runnable, ou seja, ao instanciá-la, o sistema cria um novo processo que roda em paralelo, para evitar que o jogo trave.

Essa classe faz a configuração da API JSR-82 e fica aguardando que os jogadores se conectem. Quando um jogador se conecta, a própria classe cria uma nova instância dela mesma para que fique aguardando o próximo jogador a se conectar.

Figura 12 – Tela Aguardando Jogadores

O jogo (servidor) fica aguardando os jogadores (clientes) se conectarem para que se possa iniciar o jogo. Após o usuário verificar que todos os jogadores se conectaram, o mesmo deve acionar o comando *Iniciar* para que seja dado prosseguimento ao jogo, ou seja, ao acionar esse comando será apresentada a tela mostrada na Figura 13, no qual o jogador deverá escolher a letra da rodada, ou acionar o comando *Letra Aleatória* para que o jogo escolha alguma letra que ainda não foi escolhida e enviar para os jogadores.

Figura 13 – Tela de seleção da letra da rodada do jogo

Ao acionar o comando *Iniciar Rodada* a letra será enviada para todos os jogadores que deverão escrever as respostas e acionar o comando STOP. Assim que todos os jogadores acionarem o botão STOP ou tempo limite de 2 minutos for excedido, as respostas serão apresentadas no lado servidor do jogo, conforme ilustrado na Figura 14.

Figura 14 – Respostas dos jogadores

Juntamente com as respostas dos jogadores, o jogo calcula a pontuação para cada jogador, conforme Tabela 2:

1º Jogador a responder	10 pontos
2º Jogador a responder	6 pontos
3º Jogador a responder	4 pontos
4º Jogador a responder	3 pontos
5º Jogador a responder	2 pontos
6º Jogador a responder	1 ponto

Tabela 2 – Tabela de pontuação dos jogadores

Caso o jogador escolha a opção *Nova Rodada*, o jogo volta para a tela anterior para que seja efetuada nova rodada, caso o jogador escolha a opção *Pontuação,* o jogo abre uma tela mostrando a pontuação total de cada jogador.

4.2.2 Lado cliente (*StopClient*)

O fluxo (telas) do jogo no que se refere ao lado cliente, está ilustrado na Figura 15. O lado cliente possui uma complexidade menor do que o servidor, pois só precisa se conectar ao servidor e receber os dados da rodada: letra e temas. Quando o jogador recebe esses dados, escreve as respostas e manda para o servidor processá-las.

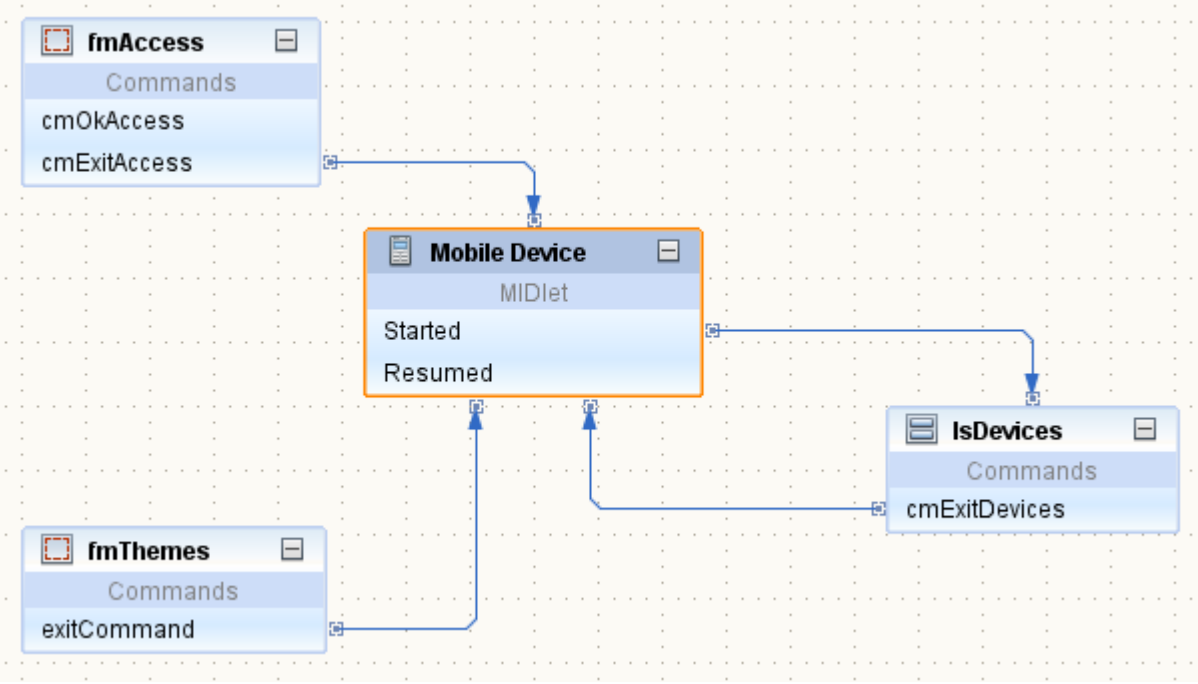

Figura 15 – Fluxo (telas) do jogo no lado cliente

4.2.2.1 Interface *DiscoveryListener*

O midlet principal do *StopClient* implementa uma interface chamada *DiscoveryListener*. A implementar essa interface, quatro métodos precisam ser obrigatoriamente implementados, como mostra a Listagem 5.

Listagem 5 – Métodos abstratos da interface *DiscoveryListener*

```
public void deviceDiscovered(RemoteDevice btDevice, DeviceClass cod) {
    throw new UnsupportedOperationException("Not supported yet.");
}

public void servicesDiscovered(int transID, ServiceRecord[] servRecord) {
    throw new UnsupportedOperationException("Not supported yet.");
}

public void serviceSearchCompleted(int transID, int respCode) {
    throw new UnsupportedOperationException("Not supported yet.");
}

public void inquiryCompleted(int discType) {
    throw new UnsupportedOperationException("Not supported yet.");
}
```

O método *deviceDiscovered* é chamado quando um dispositivo *bluetooth* é encontrado, então o mesmo é adicionado a uma lista para que possa ser manipulado posteriormente. O método *inquiryComplete* é chamado quando todos os dispositivos forem encontrados. Nesse método, a lista citada acima é percorrida e cada um dos dispositivos é adicionado numa lista na tela do celular, conforme Figura 16.

Figura 16 – Lista de dispositivos *bluetooth* encontrados

Feito isto, o jogador deve escolher qual dispositivo se refere ao servidor e selecioná-lo. Ao fazer isso, as linhas de código mostradas na Listagem 6 são executadas.

Listagem 6 – Descoberta de serviços

```
if(devices.size() != 0) {
    try {
        RemoteDevice remoteDevice = (RemoteDevice) devices
                .elementAt(getLsDevices().getSelectedIndex());

        discoveryAgent.searchServices(null, new UUID[]{uuid}, remoteDevice, this);
    } catch (Exception ex) {
        ex.printStackTrace();
    }
}
```

Nesse código, através de uma instância da classe *DiscoveryAgent* os serviços configurados no dispositivo servidor começam a ser pesquisados, e cada vez que um serviço é descoberto, o método *serviceDiscovered*, mostrado na Listagem 5, é chamado, e o que ele faz é adicionar o serviço numa lista para pode ser usado posteriormente.

Ao terminar de procurar os serviços, o método *servicesSearchComplete* é chamado, ou seja, todos os serviços já foram encontrados. Porém, apenas o serviço referente ao jogo Stop será encontrado, com isso o mesmo pode ser configurado para se comunicar com o servidor, como mostra a Listagem 7.

Listagem 7 – Configuração da JSR-82 do lado cliente

```
private void configService() {
    try {
        ServiceRecord r = (ServiceRecord) services.elementAt(0);

        String url = r.getConnectionURL(ServiceRecord.NOAUTHENTICATE_NOENCRYPT, false);

        StreamConnection con = (StreamConnection) Connector.open(url);

        input = con.openDataInputStream();
        output = con.openDataOutputStream();

    } catch (IOException ex) {
        ex.printStackTrace();
    }
}
```

4.2.2.2 Tela Inicial (Lado Cliente)

A tela inicial do *StopClient*, demonstrada na Figura 16, consiste numa lista dos dispositivos *bluetooth* disponíveis.

Após o cliente (jogador) se conectar com o servidor, o mesmo deverá definir um nome e o código de acesso definido pelo servidor e acionar o comando *Iniciar*. Caso o código de acesso esteja incorreto, o jogo informará o cliente, caso contrário o cliente ficará aguardando o servidor iniciar a rodada enviando a letra da rodada e os temas, como mostra a Figura 17.

Após o jogador escrever as respostas, deverá acionar o botão STOP e ficar aguardando a próxima rodada.

Figura 17 – Tela (lado cliente) onde o jogador deve inserir as respostas

5 CONCLUSÃO

Os professores estão começando a se aproximar das novas mídias através do uso do computador, da internet, dos cursos de educação a distância, mas ainda há muito para se descobrir e conhecer. O universo dos jogos eletrônicos é um desses locais que precisa ser mais familiar aos educadores, e os professores precisam se aproximar desse universo para se aproximar mais dos alunos.

Em relação ao projeto do jogo Stop, Fazendo um paralelo com dos objetivos iniciais com finalização do projeto, pode-se afirmar que o objetivo foi alcançado, sendo que foi possível desenvolver o jogo utilizando todos os conceitos, ferramentas e tecnologias citados nas seções desse trabalho.

Após a definição dos principais requisitos do jogo através de uma análise empírica, o próximo passo foi começar a implementar o jogo. As dificuldades encontradas foram transpostas com ajuda do orientador e de outros professores, e também com pesquisa tanto em livros quanto na internet.

O desenvolvimento do jogo Stop trouxe mais conhecimento nas áreas que foram abordadas, sendo que é um pré-requisito para projetos futuros, pois agora temos um *know-how* maior da linguagem de programação Java ME, tecnologia *bluetooth*, em geral, desenvolvimento de softwares para dispositivos móveis.

Perspectivas Futuras

Futuramente melhoramentos serão feitos no jogo, pois o mesmo foi desenvolvido utilizando componentes prontos do Java ME, perdendo a flexibilidade em relação ao design do jogo. A ideia é desenvolver componentes utilizando a tecnologia Canvas, que é uma biblioteca de classes onde é possível desenvolver componentes visuais para os aplicativos ou jogos desenvolvidos em Java ME.

Outro melhoramento que será feito é em relação a interação com a plataforma web, onde será desenvolvido um módulo para que seja possível

enviar os dados para algum servidor web e o usuário do jogo possa interagir com o jogo via *brownser* e também desenvolver um módulo que faça a interação com um console do jogo numa plataforma *desktop*.

Esses melhoramentos possibilitarão que o jogo STOP possa ser distribuído numa perspectiva comercial, que é o principal plano futuro do jogo: aproveitar o bom momento do mercado na comercialização de aplicativos e jogos *mobile*, sendo que as perspectivas de crescimento são positivas.

REFERÊNCIAS

ALECRIM, Emerson. TECNOLOGIA BLUETOOTH. Hardware, www.infowester.com.br. 2011. <http://www.infowester.com/bluetooth.php/>. Acesso: 09 jul 2011.

JSR-82: Java Bluetooth. Disponível em: < http://www.jsr82.com/jsr-82-basics>. Acesso: 23 jun. 2011.

KHOSHAFIAN, Setrag. **Banco de dados orientado a objetos.** Rio de Janeiro: Infobook, 1994. 353 p. ISBN 85-85588-38-1

LEE, Valentino; SCHNEIDER, Heather; SCHELL, Robbie. **Aplicações móveis:** arquitetura, projeto e desenvolvimento. São Paulo: Pearson Education do Brasil, 2005. 328p. ISBN 85-346-1540-3

MENDES, Cláudio Lúcio. Como os jogos eletrônicos educam? Presença pedagógica, Caiçara, BH, v. 11, p. 18-25, mar./ abr. 2005.

MUCHOW, John W. **Core J2ME:** tecnologia & MIDP. São Paulo: Makron Books, c2004. xv, 588 p. ISBN 85-346-1522-5.

MURTA, Rafael. O MERCADO DE APLICATIVOS PARA CELULAR. Notícias, bagapreta.com.br. 2010. <http://bagapreta.com.br/2010/11/13/o-mercado-de-aplicativos-para-celular-cresce-sem-parar/> Acesso: 09 jul 2011.

NAKAMURA, Freire Eduardo. **Computação móvel: novas oportunidades e novos desafios.** Disponível em: <https://portal.fucapi.br/tec/imagens/revistas/ed02_04.pdf>. Acesso: 15 mar. 2011.

Newark-French, Charles. MOBILE APPS PUT THE WEB IN THEIR REAR-VIEW MIRROR. 2011. <www.flurry.com/Mobile-Apps-Put-the-Web-in-Their-Rear-view-Mirror>. Acesso: 09 jul 2011.

O QUE É O NETBEANS. Disponível em: < http://netbeans.org/index_pt_BR.html>. Acesso: 04 ABR. 2011.

SENA, Gildeon. MOURA, Juliana. JOGOS ELETRÔNICOS E EDUCAÇÃO. <http://www.gamecultura.com.br/index.php?option=com_content&task=view&id=438&Itemid=9#axzz1TW9qmHjf>. Acesso: 24 jul 2011.

SILBERSCHATZ, Abraham; KORTH, Henry F.; SUDARSHAN, S.. **Sistema de banco de dados.** 3. ed. São Paulo: Makron, 1999. 778 p. ISBN 85-346-1073-8

TURKLE, Sherry. O segundo Eu: Os computadores e o espírito humano. 1 ª ed., Editorial Presença, Lisboa, 1989.

YOUNG, Paul H. **Técnicas de comunicação eletrônica.** São Paulo: Pearson Prentice Hall, c2006. xiii, 687 p. ISBN 8576050498.

WIRELESS TOOLKIT. Disponível em < http://www.oracle.com/technetwork/java/index-jsp-137162.html>. Acesso: 09 jul 2011.