UNIVERSIDADE FEDERAL FLUMINENSE
INSTITUTO DE COMPUTAÇÃO
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

Yuri Maximiano de Pina Ferreira

Aplicação móvel para acesso remoto às informações do quadro de horários de um departamento de ensino

Niterói
2011

YURI MAXIMIANO DE PINA FERREIRA

Aplicação móvel para acesso remoto às informações do quadro de horários de um departamento de ensino

Monografia apresentada ao Departamento de Ciência da Computação da Universidade Federal Fluminense como parte dos requisitos para obtenção do Grau de Bacharel em Ciência da Computação

Orientador: John Reed

Niterói

2011

YURI MAXIMIANO DE PINA FERREIRA

Aplicação móvel para acesso remoto às informações do quadro de horários de um departamento de ensino

Monografia apresentada ao Departamento de Ciência da Computação da Universidade Federal Fluminense como parte dos requisitos para obtenção do Grau de Bacharel em Ciência da Computação

Aprovado em Junho de 2011

BANCA EXAMINADORA

_____

John Reed, M.Sc.

Orientador

UFF

_____

Rosângela Lopes Lima, D.Sc.

UFF

_____

Leonardo Cruz da Costa, M.Sc.

UFF

Niterói

2011

RESUMO

Este trabalho tem como objetivos desenvolver e implementar uma aplicação em Java ME para dispositivo móvel capaz de de conectar-se ao servidor via rede sem fio e exibir ao usuário remoto, dentro do alcance da rede sem fio da UFF, informações do quadro de horários de um departamento de ensino, desenvolver e implementar o servidor que se comunicará com a aplicação móvel e desenvolver e implementar o servidor que enviará a aplicação móvel aos celulares dos alunos.

Palavras-Chave: Java ME, J2ME, Aplicação Móvel

ABSTRACT

This monograph deals with the construction of a mobile application in Java ME that will display information about the ongoing classes on a given semester, a server to which the mobile application will remotely connect to and a bluetooth server with the purpose of remotely pushing the mobile application to a student's mobile device.

Keywords: Java ME, J2ME, Mobiel applications

LISTA DE ABREVIATURAS, SIGLAS E SÍMBOLOS

Java ME	Java Micro Edition
J2ME	Java Micro Edition
Java SE	Java Standard Edition
UFF	Universidade Federal Fluminense
Wi-Fi	Wireless Fidelity

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

1.1 Motivação

Em algumas universidades brasileiras existe geralmente, em cada departamento de ensino de cada uma das mesmas, um quadro de horários, o qual tem a finalidade de informar aos alunos sobre as cadeiras presentes em um determinado semestre letivo. Estes quadros de horários possuem informações como o código da disciplina; o código da turma, sendo que uma mesma disciplina contém uma ou mais turmas, podendo estas serem ministradas em horários diferentes ou não; os horários nos quais as aulas de uma determinada turma serão ministradas; a sala onde será ministrada a aula de uma turma, sendo que uma mesma turma pode conter salas distintas ao longo da semana e o nome do professor que a ministrará.

Não existe porém quaisquer restrições que inviabilizem a mudança do local onde as aulas de uma turma serão ministradas, bem como o professor que as irá ministrar. Apesar de a mudança de professor não afetar sériamente os alunos de uma turma, a alteração do local das aulas pode acarretar em problemas sérios para estes. Problemas estes ocasionados pela alteração do local da sala de aula sem aviso prévio aos alunos ou, quando os alunos são avisados com antecedência, àqueles que faltaram ao dia da aula no qual o aviso foi feito.

Para remediar o problema da alteração repentina do local da sala de aula, soluções não seguras são adotas, como por exemplo a colocação de uma folha de papel na porta da sala do professor que ministra a turma em questão, ou da sala onde a aula seria realizada, contendo o aviso da mudança de sala e, caso uma nova sala já esteja disponível, o local da mesma. Outro exemplo tão comum quanto é o professor ligar para a secretaria do departamento em questão e informar sobre a alteração da sala. Estas soluções não são seguras pois a primeira depende de um objeto físico sujeito a ações de agentes externos, como por exemplo um vento levar o aviso para longe, uma pessoa arrancar o aviso da porta em um ato de vandalismo, alguém rabiscar por cima do aviso ou até mesmo o fato de o aviso ter sido mal posto e simplesmente cair ao chão e a segunda solução depende de a secretaria informar os alunos.

Ao levar em conta os problemas das soluções acima citadas, fica claro um outro problema. O quadro de horários está sujeito aos mesmos problemas do aviso em uma folha de papel. Mas por que em uma época onde há cada vez mais acesso remoto a dados armazenados em servidores, os alunos ainda precisam fazer fila para olhar o quadro de horários?

O problema da fila é porém muito pequeno quando comparado aos que serão citados a seguir. Para agravar mais ainda a situação, são raras as ocasiões nas quais os professores colocam um aviso na porta, fazendo com que nenhum dos alunos viessem a ter conhecimento da alteração da sala e ficassem a esperar pela aula na sala errada, perdendo desta forma, tanto aulas quanto provas.

Informação deve sempre estar o mais acessível quanto possível, ser fácil de ser atualizada e segura contra perdas e alterações não solicitadas. O quadro de horários não assegura nenhum desses requisitos. Estas razões levaram à idealização deste projeto.

1.2 Objetivos

Este trabalho tem como objetivos desenvolver e implementar uma aplicação em Java ME para dispositivo móvel capaz de conectar-se ao servidor, que também será desenvolvido e implementado neste trabalho, via rede sem fio e exibir ao usuário remoto, dentro do alcance da rede sem fio da UFF, informações do quadro de horários de um departamento de ensino e desenvolver e implementar o servidor que se comunicará com a aplicação móvel.

A aplicação móvel irá então requisitar ao servidor o quadro de turmas completo e o aluno poderá percorrer por toda a lista de turmas para encontrar sua turma ou simplesmente fazer uma busca com um determinado critério, como por exemplo, código da disciplina, nome da disciplina ou nome do professor. O servidor terá a finalidade de enviar a lista do quadro de turmas para a aplicação móvel quando requisitado.

Os alunos receberão a aplicação móvel via bluetooth, por meio de um servidor que também será desenvolvido e implementado neste trabalho. Este servidor executará de forma independente ao servidor pelo qual a aplicação móvel receberá as informações do quadro de horários.

1.3 Organização

Este trabalho está organizado em 4 capítulos. O segundo capítulo descreve a arquitetura da aplicação a estrutura física dos sistemas envolvidos. O terceiro capítulo descreve as tecnologias e ferramentes usadas e a implementação dos sistemas. No quarto capítulo é descrita a conclusão do trabalho.

2 ARQUITETURA

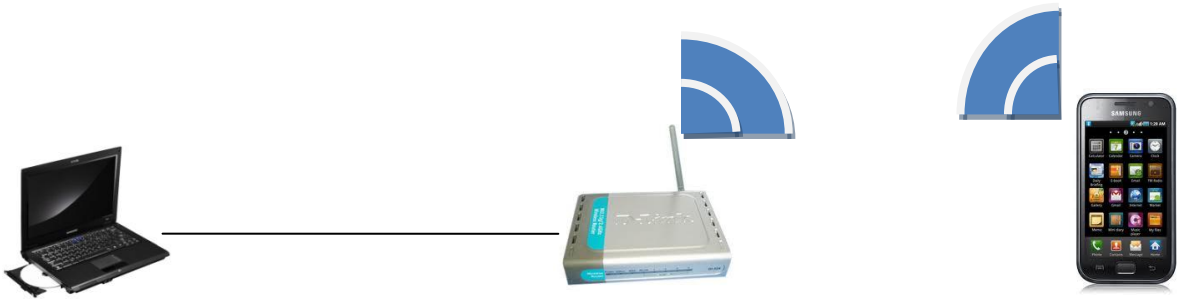
2.1 Introdução

Para melhor entender como funciona a aplicação móvel, é preciso compreender como a integração desta com os servidores HTTP e Bluetooth é feita e como o ambiente físico deve estar estruturado. As seções seguintes detalham essas questões.

2.2 Estrutura do ambiente

2.2.1 Computador

No computador estarão instalados o servidor HTTP e o servidor Bluetooth. Para que possa ser realizada a comunicação do servidor HTTP com a aplicação móvel, o computador deverá ter acesso à rede, seja via cabo ou a uma rede sem fio. Caso o computador esteja acessando uma rede sem fio, ele deverá estar conectado em um roteador com suporte a Wi-Fi conforme a figura 1.



Computador executando móvelos servidodres HTTP e Bluetooth

Roteador Wi-fi

Dispositivo

Figura 1: Estrutura física ao ambiente

Em uma das entradas USB do computador, deverá estar conectado o dongle[1] bluetooth, que fará com que o computador seja capaz de localizar dispositivos com bluetooth ligado dentro do alcance, que irá depender do produto utilizado. A Figura 2 mostra um exemplo de dongle bluetooth.

Figura 2: Dongle Bluetooth

2.2.2 Dispositivo móvel

No dispositivo móvel, estará instalada a aplicação móvel no formato JAR[2]. Para que este dispositivo possa realizar a comunicação com o servidor HTTP presente no computador, aquele deverá ter suporte à rede Wi-Fi. Para que o arquivo JAR possa ser enviado ao dispositivo móvel, este deverá também ter suporte à comunicação bluetooth e ao serviço Obex[3].

O servidor HTTP e a aplicação instalada no dispositivo móvel comunicar-se-ão seguindo o modelo cliente-servidor. Para mais detalhes sobre este modelo, consultar a seção 3.2 deste trabalho. O servidor Bluetooth irá procurar dispositivos móveis com serviço bluetooth ativo e enviará para estes, a aplicação móvel. A Figura 3 ilustra estas comunicações e a Figura 4 apresenta o fluxo de execução do servidor Bluetooth.

[1] Sem tradução direta para a língua portuguesa, é muito semelhante a um pendrive. Sua única finalidade é proporcionar serviço de comunicação bluetooth ao dispostivo ao qual estiver conectado.

[2] Formato padrão para aplicações em Java ME

[3] Ver seção 3.1.3

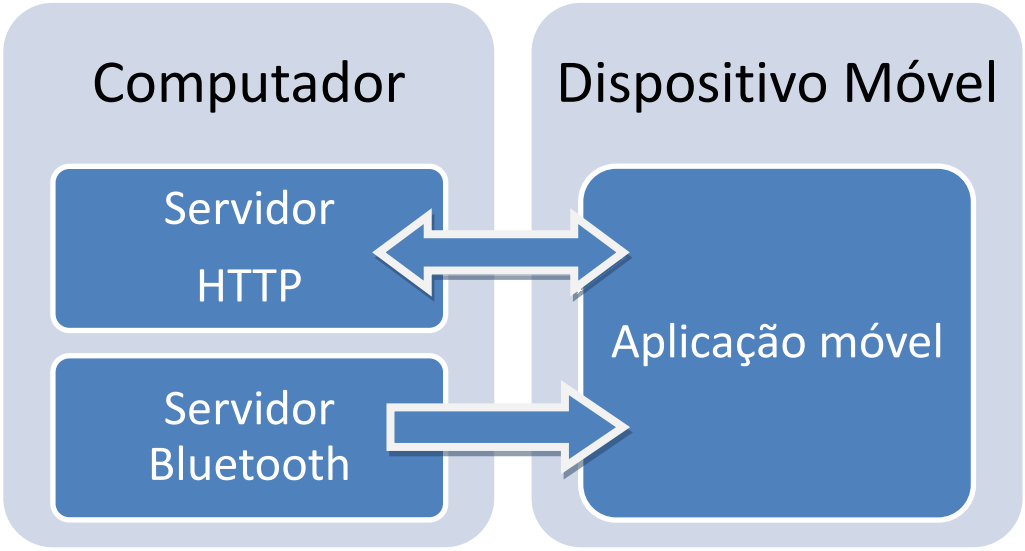


Figura 3: Esquema da comunição entre as aplicações

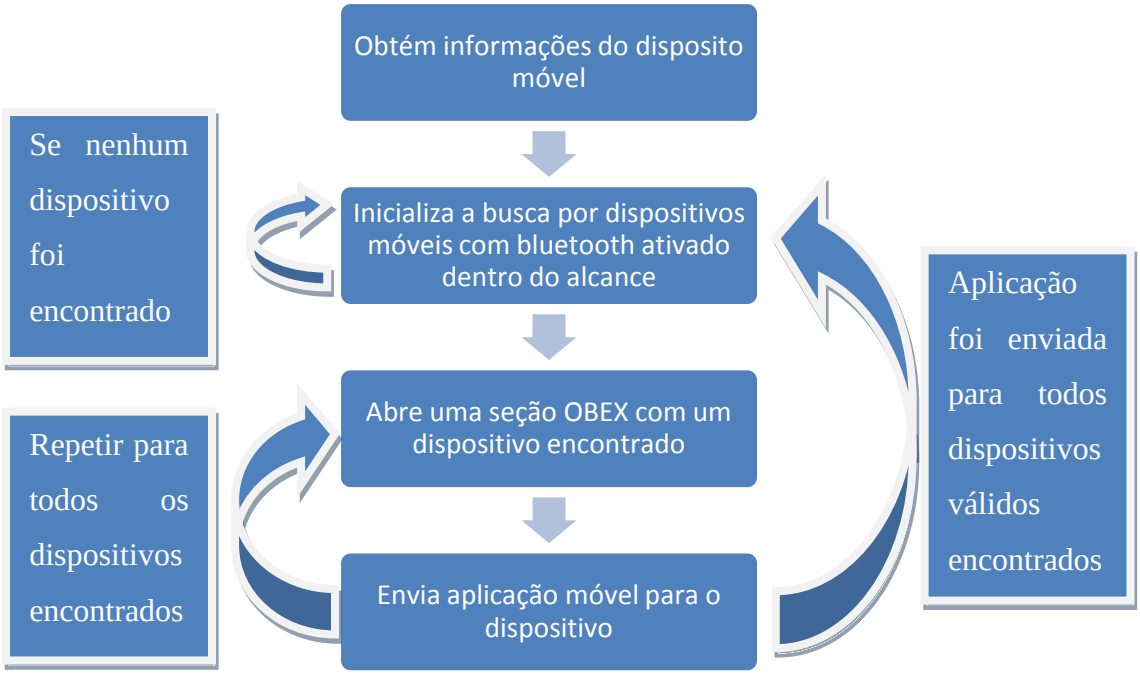


Figura 4: Fluxo de execução do servidor Bluetooth

3 IMPLEMENTAÇÃO

3.1 Tecnologias usadas

3.1.1 IEEE 802.11

IEEE 802.11 consiste em um conjunto de padrões que definem como conectar dispositivos, como computadores e celulares, à uma rede (seja ela local ou externa) sem o uso de cabos. Dentro do alcance dessa rede, esses dispositivos podem enviar e receber dados. Desse conjunto de padrões, a escolha de um padrão específico a ser usado depende de requerimentos como: segurança, interferência externa, custo, alcance, compatibilidade e velocidade de comunicação de dados. Esses padrões são: 802.11b, 802.11a e 802.11g.

O padrão 802.11b, também conhecido como Wireless Fidelity (Wi-Fi), é o mais popular dentre os demais. Seu alcance é amplo o suficiente para ser usado em escritórios e residências. Sua popularidade deu-se por seu baixo custo e velocidade de até 11Mbps dentro de um alcance de 30 metros. Contudo está sujeito à interferência de celulares móveis e dispositivos bluetooth.

O padrão 802.11a possui algumas vantagens com relação ao citado acima. Por estar operando numa banda de freqüência menos povoada (entre 5.15GHz e 5.35Ghz, enquanto a 802.11b opera na 2.4GHz), está menos sujeita à interferências. Seu pico de velocidade também é mais vantajoso, chegando à 54Mbps (porém na prática chega próximo à 25Mbps).

O padrão 802.11g consegue unir as vantagens dos dois padrões acima citados. Além de ter a velocidade do 802.11a, é compatível com 802.11b e opera na mesma banda deste. Porém a questão de interferência ainda persiste.

Em um ambiente onde os dispositivos estão ligados à rede por meios de cabos, os dispositivos ficam estruturados da seguinte forma: A um dispositivo chamado roteador, é connectado o cabo que sai do modem, o qual provê o serviço (exemplos: Virtua, Velox, GVT). Deste roteador sairão cabos para fazer com que quaisquer dispositivos a eles conectados, tenham acesso à rede. Contudo se somente um dispositivo for eventualmente conectado, o roteador pode ser descartado e o modem conectado diretamente no dispositivo.

Com essa estrutura é facil prever o acúmulo de cabos conforme mais dispositivos venham a ser conectados ao roteador.

Para que se possa ter uma rede sem fios nesse ambiente, bastaria, além de conectar o modem ao roteador, conectar um ponto de acesso sem fio (Wireless Access Point ou WAP) ao mesmo. Dessa forma, quaisquer dispositivos prontos para usar rede sem fio, poderão ter acesso à rede.

Em termos de segurança, existem os seguintes tipos disponíveis para o 802.11: WEP, 802.1x, LEAP, PEAP, WPA, TKIP e WPA2. WPA2 é a segunda geração da WPA e possui o nível de segurança mais alto entre os demais tipos. Disponibilizada em setembro de 2044 pela Wi-Fi Alliance, o método de autenticação depende do fato do ambiente ser empresarial ou doméstico. O mecanismo de encriptação usado em ambos é o AES, o qual segue os requerimentos de segurança do governo dos Estados Unidos da América.

O mecanismo de segurança WPA2 funciona da seguinte forma: quando um usuário tenta obter acesso a um AP (ponto de acesso), o AP bloqueia o acesso à rede até que as informações necessárias sejam fornecidas pelo usuário. Esse processo garante que apenas usuários autorizados ganhem acesso à rede e que o cliente está se conectando à um servidor autorizado. Assim que o servidor aceita as informações vindas do cliente, este ganha acesso à rede.

3.1.2 Protocolo de Transferência de hipertexto (HTTP)

O Protocolo de Transferência de hipertexto, cujo acrônimo em inglês é HTTP, é um protocolo de rede para sistemas distribuidos, cooperativos e de informação hipermidia [1]. O HTTP é um protocolo de “pedido-resposta” em um modelo de cliente-servidor. Para ilustrar isto, um navegador de internet representa o cliente e a aplicação que este está querendo acessar está em um computador que representa o servidor. O cliente submete uma mensagem para o servidor, o servidor então recebe a mensagem enviada e realiza as operações correspondentes de nível de aplicação. Assim que o servidor terminou este processamento, retorna ao cliente uma mensagem de resposta. Essa mensagem de resposta é muito comumente uma página de um site da Internet (FIELDING, 199). Um exemplo de mensagem enviada ao servidor, inclusive usada nesse projeto, é:

"http://192.168.0.187:8080/ServidorQuadroTurmas/abreconta?action=turmas".

3.1.3 Bluetooth

Bluetooth é um padrão de tecnologia sem fio para troca de dados através de curtas distâncias, atualmente tendo o alcance de até cem metros, variando de produto a produto. Um dispositivo bluetooth pode comunicar-se com até sete dispositivos em um piconet.[4] Os dispositivos podem alterar seus respectivos papéis a qualquer momento, fazendo com que o dispostivo mestre vire o escravo e vice-versa. (BB-ELEC, 2011)

3.1.4 Obex

Obex , Object Exchange (troca de objetos), é um protocolo de comunicação adotado pelo Bluetooth Special Interest Group para facilitar a transferência de objetos binários entre dispositivos móveis via Bluetooth. É bastante similar ao HTTP, usando o mesmo modelo cliente-servidor.(MONSON, 2011)

3.2 Ferramentas Usadas

3.2.1 Java ME

Esta seção e suas respectivas subseções são resultantes da leitura de (YUAN, 2011) e de conhecimentos adquiridos através de diverso fóruns online sobre desenvolvimento para Java ME. Nela será apresentado uma introdução breve, uma visão sobre a interface gráfica e sobre o armazenamento de dados em Java ME. Ao final desta seção, será apresentado como desenvolver aplicações móveis em Java ME.

[4] Rede ad-hoc usando tecnologia bluetooth

3.2.1.1 Introdução

Java Microedition é uma plataforma Java desenvolvida pela Sun Microsystems Inc, atualmente adquirida pela Oracle Corporation, para sistemas simples, sendo os dispositivos móveis um exemplo de tais sistemas. Java ME implementa perfis (em inglês: profiles), sendo usado neste trabalho o Mobile Information Device Profile (MIDP), cujo alvo são dispositivos móveis, como por exemplo celulares. MIDP inclui uma interface com usuário, uma API para armazenamento de dados e uma API básica para desenvolvimento de jogos. Aplicações desenvolvidas para este perfil são chamadas de MIDlets.

Desenvolver aplicação para Java ME não é diferente de desenvolver para Java SE e programadores experientes desta não irão encontrar muitos obstáculos pelo caminho. As principais diferenças em desenvolver para Java ME estão relacionadas à sistemas de armazenamentos de dados, como por exemplo arquivos e banco de dados, e na interface gráfica à primeira vista limitada e ao longo prazo, complexa. Estas diferenças serão tratadas nas seções a seguir.

3.2.1.2 Interface Gráfica

Uma aplicação Java ME interage com o usuário por meio de telas[5]. Uma tela possui itens como texto, caixa de texto editável, etc.. Como Java ME foi desenvolvido em uma época na qual não existia dispositivos com telas físicas sensíveis a toque, para que o usuário navegasse de uma tela para outra era necessário mapear comandos a botões físicos do celular. Os botões mais comumente mapeados para operações que avançassem a aplicação para a tela seguinte ou retornassem a aplicação para a tela anterior, eram os botões como indicados pelas setas azuis na Figura 5. Para navegar pelos itens da tela, como por exemplo, em uma lista de itens, como visto na Figura 6[6] ,são comumentes usados os botões ao redor do botão indicado pela seta vermelha. Este botão por último mencionado, é comumente mapeado para selecionar um item de uma lista. A palavra “comumente” foi bastante usada e isso não foi ao acaso.

[5] Consultar a seção 4.3 ajudará a entender o conceito de telas.

[6] Todas as figuras relativas a telas da aplicação estão presentes na aplicação móvel desenvolvida e implementada neste trabalho.

Estas operações são de fato o mapeamento padrão para estes botões, mas nada impede que o programador altere isso como bem desejar.

Como exemplo do que foi citado acima, na Figura 7 vê-se presente os 3 tipos mais comuns de itens de uma tela. Os retângulos são caixas de texto editáveis. Para o usuário selecionar essas caixas de texto para editar, deverá usar como mencionado acima, os botões ao redor do botão indicado pela seta vermelha.

Figura 5: Botões físicos de um dispositivo móvel

Quando todas as informações já tiverem sido digitadas, o usuário poderá apertar o botão “Buscar”, ou se desejar retornar à tela anterior, apertar o botão “Voltar”. Mas onde estão esses botões em uma tela física não sensível ao toque? Se for usado o mapeamento padrão, o botão de “Buscar” estará mapeado para o botão indicado pela seta vermelha e o botão “Voltar” para o botão indicado pela seta azul à direita.

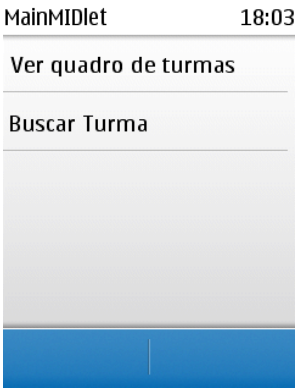


Figura 6: Lista de itens

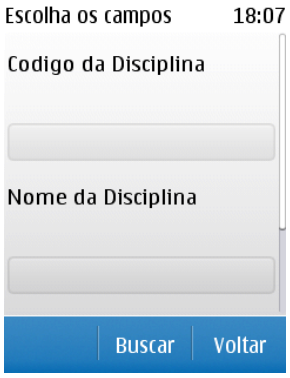


Figura 7: Itens de tela

Percebe-se então que o botão virtual[7] “Buscar” foi mapeado para o botão que por padrão, era usado para selecionar um item de uma lista. Isso aconteceu por questão de localidade. O botão associado ao botão “Buscar” está no centro do dispositivo e o botão associado ao botão “Voltar” está na extremidade direita. Se houvesse um botão virtual à esquerda do botão virtual “Buscar”, este estaria mapeado por padrão para o botão na extremidade esquerda, que estava indicado pela seta azul à esquerda.

Um programador poderia dizer que todos esses itens de tela são bem interessantes e fáceis de usar mas ele gostaria de criar itens com um design próprio e desenhar o logotipo de sua empresa na aplicação. É neste momento que o desenvolvimento de interfaces em Java ME fica complexo. Para fazer isso, o programador precisaria usar uma classe chamada Canvas e pintar pixel por pixel nas coordenadas desejadas. Isso faz com que a muitas empresas que desenvolvem aplicações para dispositivos móveis, gastem um bom tempo criando uma biblioteca gráfica que use Canvas para que ao longo prazo, o processo de desenvolvimento de outras aplicações Java ME seja acelerado.

3.2.1.3 Armazenamentos de Dados

A memória interna de um dispositivo móvel é bem menor do que a memória de um computador. Por exemplo, o celular N95 da Nokia possuia 164 Megabytes de memória interna e 64 Megabytes de RAM (NOKIA, 2011), enquanto o computador usado neste projeto possui 1.75 Terabyte de memória interna (memória secundária) e 4 Gigabytes de RAM. Devido a esta baixa memória dos dispositivos, armazenar uma base de dados na memória interna do mesmo torna-se ineficaz e portanto inviável.

Para contornar tal problema, a Sun Microsystems Inc., atualmente adquirida pela Oracle Coporation, desenvolveu uma biblioteca chamada “rms” no pacote “java.microedition”, a qual fornecia um sistema de arquivos com baixo uso de memória. Nessa biblioteca, existe uma classe chamada RecordStore que representa um arquivo. Um RecordStore possui um identificador textual a ele associado e uma lista de elementos. Um elemento possui um valor textual e identificador numérico. Pode-se fazer uma analogia entre RecordStore e uma tabela em um banco de dados, onde cada elemento do RecordStore é uma tupla da tabela. Analogamente, para representar as colunas de uma tupla, o programador

[7] Existe na interface com usuário mas não é um objeto concreto

deverá inserir no valor do elemento, um caracter que será usado para separar os valores de cada coluna. Há um exemplo de código sobre este assunto na seção 4.2.1.4.

3.2.1.4 Exemplos de código

Neste seção e em suas subseções, serão apresentados trechos de códigos pertencentes à aplicação móvel desenvolvida, ou variações destes, e implementada neste trabalho, com o intuito de mostrar as particularidades do Java ME quanto à implementação da interface gráfica e do armazenamento de dados.

Quando o usuário, através da interface do sistema operacional do dispositivo móvel, abre uma aplicação móvel, o sistema automaticamente instancia a classe MIDlet desta aplicação chamando o construtor da mesma e em seguida, chama o método startApp. A Figura 8 apresenta a estrutura básica de uma MIDlet.

```
public class MainMIDlet extends MIDlet {

    public MainMIDlet() {
    }

    public void startApp() {
    }
```

Figura 8: MIDlet

De forma a garantir a arquitetura MVC (Model-View-Controller), é aconselhável que a MIDlet não seja responsável por gerenciar interface gráfica. Ela deve apenas ser responsável por inicializar a aplicação móvel, instanciar a primeira tela da aplicação e exibir esta para o usuário. A Figura 9 apresenta um trecho de código para este exemplo.

A classe Display possui o método setCurrent que fará uma chamada ao sistema operacional do dispostivo móvel para que uma nova tela, passada como parâmetro, se torne visível ao usuário. O construtor da MIDlet instancia o display e o método startApp chama o método setCurrent daquele.

```
public class MainMIDlet extends MIDlet {
    public static Display display = null;

    public MainMIDlet() {
        display = Display.getDisplay(this);
    }

    public void startApp() {
            TelaInicio telaInicio = new TelaInicio(this);
            display.setCurrent(telaInicio);
    }
```

Figura 9: Método startApp

É necessário entender o que é uma tela em Java ME antes de tudo. Uma tela que estará visível ao usuário em um determinado momento da aplicação móvel, tenha ela interação com o usuário por meio de botões, caixas de texto, etc. ou simplesmente mensagens de texto que estarão visíveis apenas por um breve instante de tempo, pode ser de quatro tipos diferentes de tela.

Dentre eles, existe a tela de alerta (Alert), que exibirá uma mensagem para o usuário por cima da tela existente, a tela de formulário (Form) que é capaz de exibir todos os tipos de itens existentes, como textos, caixas de texto, barras de progresso, etc. e mapear botões e a tela de lista, que exibirá uma lista de itens (List), que pode ser implicita (apenas um item pode ser selecionado por vez) ou exclusiva (vários itens podem ser selecionado de uma vez).

Essas três telas, Alert, List e Form são classes Java ME que herdam métodos da classe abstrata Screen, que por sua vez os herda da classe Displayable. A Figura 10 apresenta essa hierarquia. Por questão de praticidade, a partir deste instante quando for citada a palavra "tela", deve-se generalizar o tipo dela, ou seja, se ela é uma Form, List ou Alert. Basta pensar em uma tela como uma Displayable.

Uma tela possui itens como foi acima citado. Itens podem ser de 8 tipos. Dentre este tipos estão um grupo de escolha (ChoiceGroup) onde o usuário deverá escolher qual(is)

item(ns) do grupo marcar, uma barra de progresso (Gauge), uma imagem (ImageItem), um texto (StringItem) e uma caixa de texto editável (TextField). ChoiceGroup, Gauge,

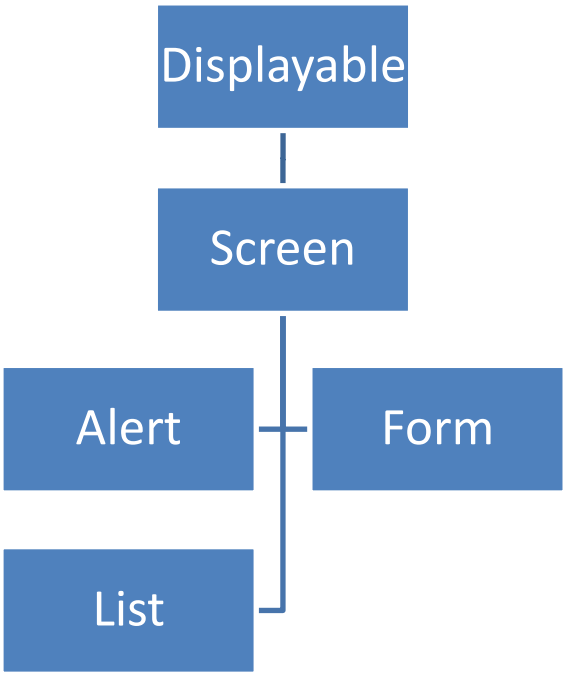


Figura 10: Hierarquia das telas

ImageItem, StringItem e TextField são classes que herdam métodos da classe abstrata Item, como mostra a Figura 11.

Talvez pareça que um item está faltando, o item de botão. Porém em Java Me, botão não é um item, pela simples razão de que, como foi explicado na seção 3.2.1.2, não é possível interagir com ele na tela da aplicação. Ao invés disso, um texto com o nome do botão é posicionado o mais perto possível do botão físico do dispositivo móvel. Portanto, o botão em Java ME é chamado de comando (Command) e a tela, seja ela Form, List,etc., possui o método addCommand para inserir comandos na tela, como mostra a Figura 12. Para instanciar um comando, é chamado o construtor da classe Command com os seguintes parâmetros em ordem: Nome do comando que será exibido ao usuário, funcionalidade padrão do comando e prioridade do comando. A prioridade do comando é apenas um termo elegante para dizer posição do comando na tela. A funcionalidade padrão do comando pode, como foi visto na seção 4.2.1.2, ser descartada para dar mais liberdade ao desenvolvedor.

Quando o usuário pressiona um botão no dispostivo móvel que foi mapeado para um comando, é chamado o método commandAction da classe que corresponde à tela na qual o

botão foi mapeado. A Figura 13 mostra como isso é feito. Para que este método seja chamado, a classe deve implementar a interface CommandListener e chamar o método setCommandListener no construtor.

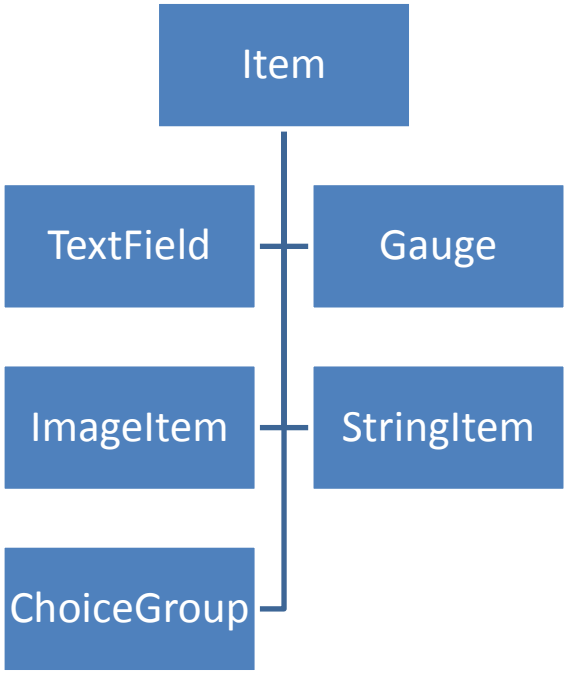


Figura 11: Hierarquia dos itens de tela

```
public class TelaInicio extends Form implements CommandListener {

    private Command start = null;

    public TelaInicio(MainMIDlet midlet){
        super(null);
        start = new Command("Iniciar", Command.OK, 0);
        addCommand(start);
        setCommandListener(this);
    }
}
```

Figura 12: Adição de comandos

O parâmetro Command representa a instância do comando que foi chamado. Logo basta comparar o parâmetro com a instância da variável correspondente. Se o comando for, por exemplo, “start”, então realiza uma sequência de passos. Nesse caso, exibe uma nova tela.

```
public class TelaInicio extends Form implements CommandListener {
    private Command start = null;
    private MainMIDlet midlet;

    public TelaInicio(MainMIDlet midlet){
        super(null);
        this.midlet = midlet;
        append("Quadro de Turmas Movel");
        start = new Command("Iniciar", Command.OK, 0);
        addCommand(start);
        setCommandListener(this);
    }
    public void commandAction(Command c, Displayable d) {
        if (c == start){
            AquisicaoIP aquisicaoIP = new AquisicaoIP(midlet);
            MainMIDlet.display.setCurrent(aquisicaoIP);
        }
    }
}
```

Figura 13: Método commandAction

O parâmetro Displayable indica em qual tela foi chamado o comando. Isso pode parecer um tanto confuso, porém é mais simples do que parece. Uma tela pode estender uma Form, mas nada impede que exista uma variável membro a esta classe do tipo Form também. Mas então quando criar uma classe nova para uma Form e quando usar internamente? Se a form possuir muitos comandos e uma lógica complexa, é sempre aconselhável torná-la uma classe.

Para melhor explicar, basta pensar em uma tela que possua um comando mapeado cuja função seja exibir uma tela de alerta. Telas de alerta são bem simples e não faria sentido criar uma classe para cada uma existente na aplicação. Inclusive, nem seria preciso torná-la uma variável membro à classe, seria suficiente que ela fosse uma variábel local ao método commandAction, como mostra a Figura 14.

```
public void commandAction(Command c, Displayable d) {
    if (c == start){
        Alert alert = new Alert(null);
        alert.setTimeout(ALERT_TIMEOUT_MS);
        alert.setTitle("Error");
        alert.setString(message);
        alert.setType(AlertType.ERROR);
        display.setCurrent(alert, displayable);
}
```

Figura 14: Tela de alerta

O armazenamento de dados em Java ME merece uma atenção especial. Como citado previamente, uma instância da classe RecordStore representa analogamente uma tabela de um banco de dados e cada elemento representa uma tupla. Conforme mostra a Figura 15, obtém-se uma instância de RecordStore pelo método openRecordStore, que possui como parâmetros o nome atribuido a ele e um valor booleano para criar um novo RecordStore caso não existe (true) ou não (false). É de responsabilidade do programador fechar um RecordStore antes de abrir outro, caso contrário, valores serão armazenadas em locais inesperados da memória. Isso é feito pelo método closeRecordStore.

Neste exemplo foi usado o caracter “;” (ponto e vírgula) para sevir de delimitador. Desta forma podemos analogamente à comparação acima, dizer que codigoDisciplina, nomeDisciplina, codigoTurma, nomeProfessor e numeroSala são colunas da tebela de nome

```
RecordStore recordStore.openRecordStore(NOME_RECORD_STORE ,
true);
String record = new String(codigoDisciplina + ";" + nomeDisciplina +
";" + codigoTurma + ";" +  nomeProfessor + ";"  + numeroSala + ";");
disciplineRecordStore.addRecord(record.getBytes());
disciplineRecordStore.closeRecordStore();
```

Figura 15: Criação de um RecordStore

igual ao nome que foi atribuído ao RecordStore. É então adicionado ao RecordStore a String convertida para bytes.

Há várias formas diferentes de se varrer um RecordStore, porém algumas não são muitos confiáveis. Existe uma classe chamada RecordEnumration que poupa ao programador o trabalho de criar toda uma lógica para enumerar os elementos do RecordStore. Contudo, quando um elemento é removido, a memória não se re-organiza e surge um “buraco” no RecordStore. A enumeração que se seguir após uma remoção poderá conter dados inválidos e o programador terá dificuldades para localizar onde ocorreu o erro.

Para contornar esse problema, foi implementado neste trabalho uma forma mais trabalhosa porém menos suscetível a erros. A Figura 16, apresentar essa implementação com uma mistura de Java ME com pseudo-código auto explicativo.

```
Vector records = turmasRecordStore.searchRecords(queryList);
for (int i = 0; i < elementos.size(); i++){
    obtém elemento na posição i
    obtém o valor do elemento
    imprime o valor na tela do dispositivo
}
```

Figura 16: Enumeração de RecordStores

Apesar de não serem utilizados neste trabalho, o código da aplicação móvel possui métodos para atualizar e remover elementos de um RecordStore. Estes métodos estão lá com

finalidade didática e o código fonte estará todo disponível no Apêndice D deste trabalho, para que fique acessível a todos.

O método de atualização é como mostra a Figura 17 , cujos parêmetros são o id do elemento no RecordStore e o novo valor a ser atualizado. O método de remoção de elementos é como mostrado na Figura 18, cujo único parâmetro é o id do lemento no RecordStore.

```
public void update(){
        recordStore.openRecordStore(NOME_RECORD_STORE ,true);
        recordStore.set(idRecord, novoDado);
        recordStore.closeRecordStore();
 }
```

Figura 17: Atualização de um elemento de um RecordStore

```
public void delete(){
        recordStore.openRecordStore(NOME_RECORD_STORE ,true);
        recordStore. deleteRecord (idRecord);
        recordStore.closeRecordStore();
 }
```

Figura 18: Remoção de um elemento de um RecordStore

3.2.2 MySQL

MySQL é um sistema gerenciador de banco de dados que foi adotado neste trabalho por oferecer uma interface com usuário intuitiva para a criação de tabelas, através do aplicativo MySQL Query Browser e por possuir driver para Java.

3.2.3 Bluecove

Biblioteca Java para realizar a comunicação entre dispositivos bluetooth e a transferência de dados entre eles.

3.3 A Aplicação Móvel.

A Aplicação móvel interage com o usuário através de telas. Uma tela representa uma interface gráfica que proporciona ao usuário navegar pelas demais telas da aplicação. Cada tela tem um objetivo específico e poderá conter um ou mais componentes de tela. Esses componentes são: textos editáveis, textos não editáveis, botões, barras de progresso, entre outros.

A cada componente pode estar associado um evento, por exemplo, o pressionar de um botão, e cada evento irá fazer com que a aplicação se comporte de um jeito específico. Por exemplo, ao pressionar o botão “Voltar”, a tela atual é desativada e a tela anterior passa a ser visível ao usuário.

A Figura 19 apresenta as telas da aplicação e a navegação entre eles dá-se como na Figura 20. A tela 1 é a tela inicial da aplicação. Ao pressionar o botão “Iniciar” o usuário é direcionado para a tela 2, a qual irá obter o IP do servidor. Após a obtenção do IP, o usuário será direcionado para a tela 3, onde as tuplas da tabela do quadro de turmas serão carregadas na memória interna do celular.O usuário será perguntado se deseja permitir acesso à internet durante este processo. Após responder “Yes” e todas as tuplas terem sido obtidas, o usuário será direcionado para a tela 4.

Nessa tela, o usuário poderá escolher entre visualizar todas as turmas de uma vez selecionando a linha com o texto “Ver quadro de turmas”, ou fazer uma busca mais específica selecionando a linha com o texto “Buscar Turma”. Ao selecionar a primeira opção. O usuário será direcionado para a tela 5 onde serão exibidas todas as tuplas do quadro de turmas. Ao clicar na segunda opção da tela 4, o usuário será direcionado para a tela 6, onde deverá escolher qual(is) critério(s) de busca usar. Os critérios são: código da disciplina, nome da disciplina e nome do professor. Após selecionar o botão “Buscar”, o usuário será direcionado para a tela 6, porém neste caso somente as tuplas que satisfizerem o(s) critério(s) serão exibidas.

3.4 O Servidor HTTP

O servidor HTTP tem o único propósito de, quando a requisição pelo quadro de horários for feita, consultar o banco de dados e enviar o resultado da consulta para o

dispositivo móvel. Foi desenvolvido em Java SE e usa a biblioteca Hibernate para consultar a base de dados. A figura 21 apresenta o código fonte da classe AbreContaServlet que é responsável por receber o pedido de consulta à base de dados enviado pelo dispositivo móvel, realizar operações a partir deste pedido e enviar como resposta as informações requisitadas.

O parâmetro request, da classe HttpServletRequest, contém o pedido feito pelo dispositivo móvel. Pelo método getParameter do request, obtém-se o valor associado à tag "action", sendo o nome desta tag escolhido aleatóriamente. Um exemplo de URL que será enviada pelo dispositivo móvel é:

"http://192.168.0.187:8080/ServidorQuadroTurmas/abreconta?action=turmas".

Se a action for "turmas", então o pedido está correto e será então enviada uma resposta para o dispositivo móvel através do parêmetro response, cuja classe é HttpServletResponse. O código fonte completo do servidor HTTP está no Apêndice E.

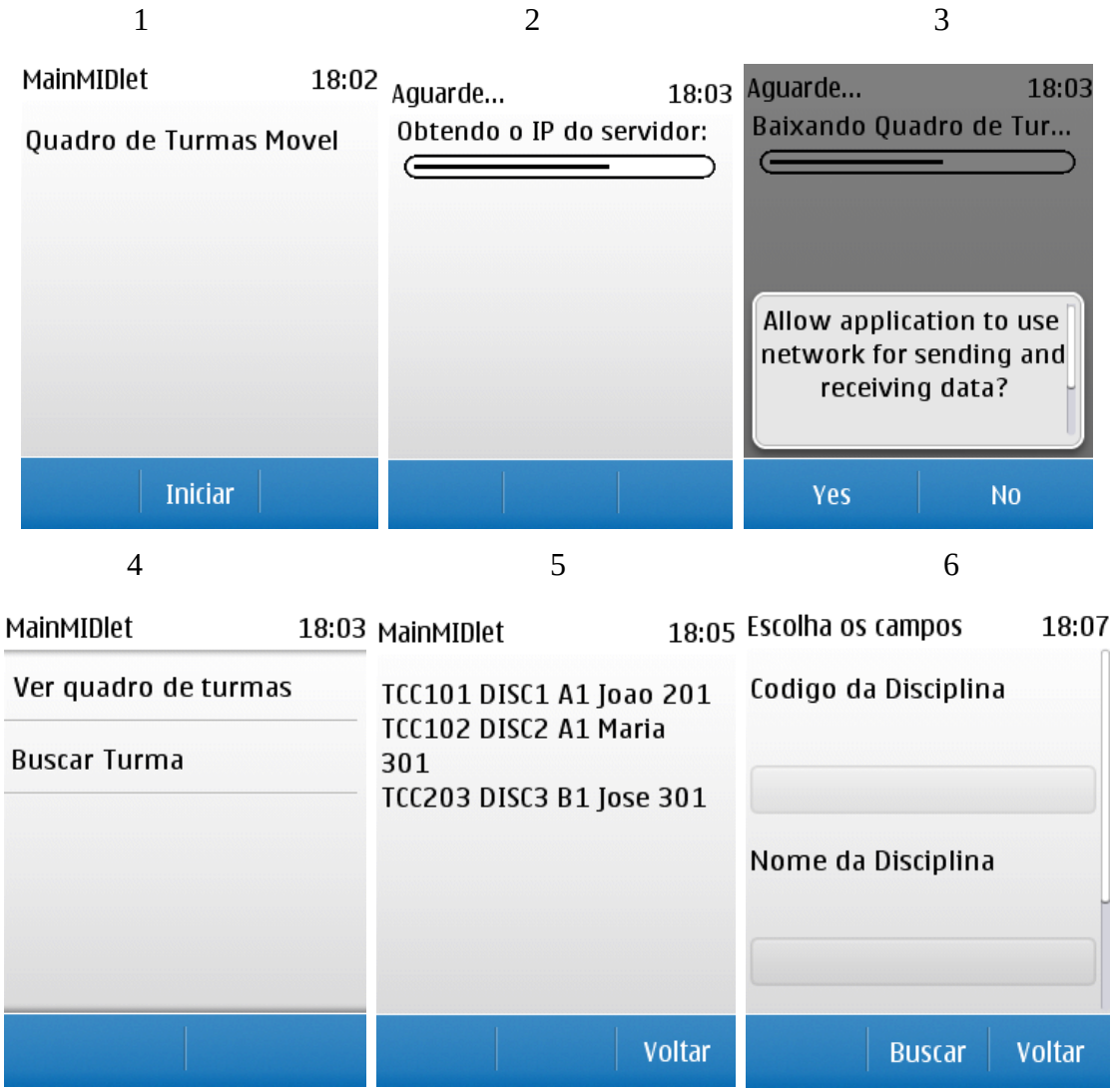


Figura 19: Telas da aplicação

Figura 20: Navegação das telas

A figura 22 apresenta a tabela “turma” da base de dados. Por ser apenas uma demonstração, dados reais não foram utilizados. A tabela possui as seguintes colunas: id, que é a chave primária; codigoDisciplina, que representa o código da disciplina (Essa coluna poderia ser a chave primária, descartando assim a coluna id, porém para simplificar os testes durante a implementação da comunicação entre o servidor HTTP e o dispositivo móvel, foi mantida desta forma); nomeDisciplina, que é o nome da disciplina; codigoTurma, que é o código da turma; nomeProfessor, que é o nome do professor e numeroSala, que é o número da sala de aula.

3.5 O Servidor Bluetooth

O servidor Bluetooth foi desenvolvido em Java SE e usa a biblioteca bluecove para realizar comunicações via bluetooth com dispositivos móveis. Boa parte do código fonte foi retirada de foruns sobre programação em Java e está disponível por completo no Apêndice F. Sua função é enviar a aplicação em Java ME para todos os dispositivos móveis que possuam bluetooth, aceitem o protocolo Obex e estejam ao alcance do dongle bluetooth conectado ao computador por uma porta USB.

Para explicar o código fonte, este será dividido nas seguintes etapas:

3.5.1 Inicialização

Etapa onde será obtido o endereço bluetooth do computador local, através do método LocalDevice.getLocalDevice(). A figura 23 apresenta o código desta etapa.

```
protected void processRequest(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        System.out.println("Entrou no processRequest");
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        String action = request.getParameter("action");
        TurmaService turmaService = null;
        if (action.compareTo("turmas") == 0){
           try {
              turmaService =
            (TurmaService) FabricaDeAppService.getAppService(TurmaService.class);
           } catch (Exception ex) {
              out.write(FAILURE);
              System.out.println("Error creating appServices" + ex.getMessage());
           }
           String listaTurmas = "";
           List<Turma> turmas = turmaService.getListTurmas();
           for (Turma turma : turmas) {
              String data = new String(turma.getCodigoDisciplina() + ";" +
                turma.getNomeDisciplina() + ";" + turma.getCodigoTurma() + ";" +
                turma.getNomeProfessor() + ";" + turma.getNumeroSala() + ";/");
              listaTurmas += data;
           }
           System.out.println("Turmas: " + listaTurmas);
           out.write(listaTurmas);
        }
     }
```

Figura 21: Servlet do servidor HTTP

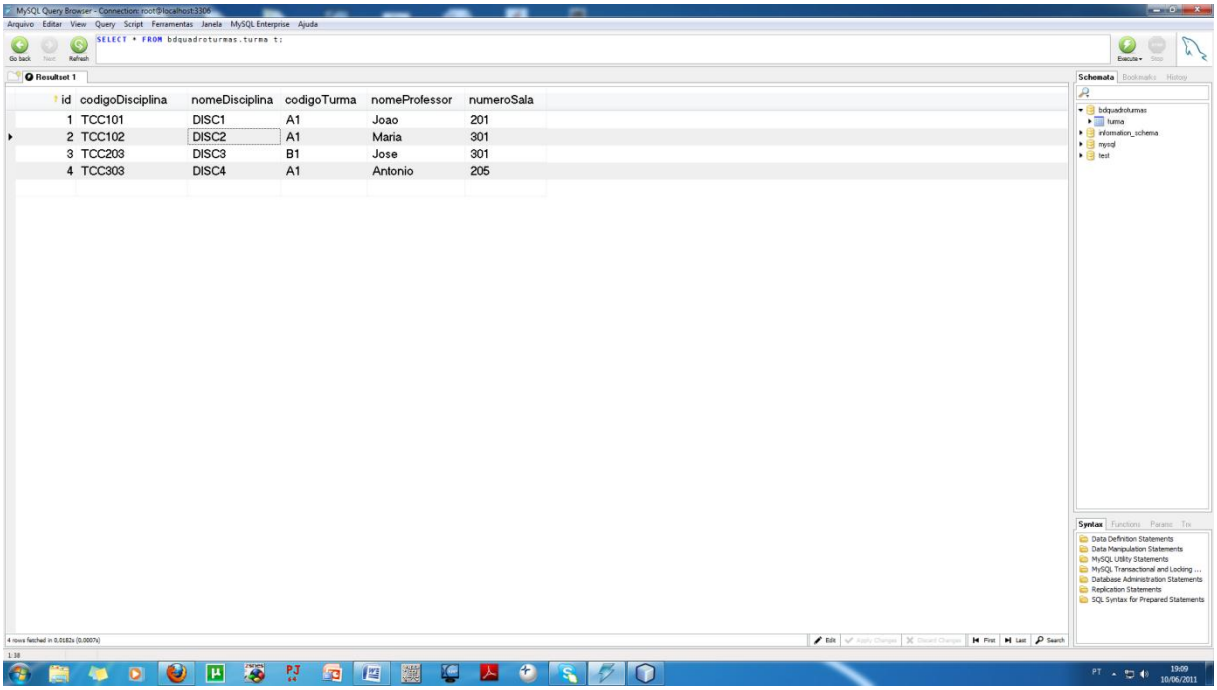


Figura 22: Tabela do quadro de horários

```
try {
        localDevice = LocalDevice.getLocalDevice();
} catch (BluetoothStateException ex) {
       // tratar exceção aqui
}
```

Figura 23: Etapa de inicialização do servidor bluetooth

3.5.2 Busca por dispositivos

Nesta etapa será realizada uma busca por dispositivos com bluetooth ativo. A busca é feita através do método startInquiry da classe DiscoveryAgent. Um DiscoveryAgent representa o agente local que irá realizar as operações de comunicação bluetooth A figura 24 apresenta o código desta etapa.

```
DiscoveryAgent agent = localDevice.getDiscoveryAgent();
try {  agent.startInquiry(DiscoveryAgent.GIAC, this);
} catch (BluetoothStateException ex) { // tratar exceção aqui
}
```

Figura 24: Etapa de busca por dispositivos móveis

3.5.3 Busca por serviços

Para cada dispositivo encontrado, será procurado o serviço de transferência de objeto binário, que usa o protocolo Obex. Para isso, primeiramente será feita a busca a partir do método searchServices do DiscoveryAgent, como mostra a figura 25.

```
for (int j = 0; j < devs.size(); j++){
        remoteDevice=(RemoteDevice)devs.elementAt(j);
        RemoteDevice remoteDevice=(RemoteDevice)devs.elementAt(j);
        UUID[] uuidSet = {new UUID("1105", true)};
        int[] attrSet = {0x0100, 0x0003, 0x0004};
        connectionURL = null;
        int transID=0;
        try {
           transID = agent.searchServices(attrSet, uuidSet, remoteDevice, this);
        } catch (BluetoothStateException ex) {
           // tratar exceção aqui
        }
```

Figura 25: Chamada da busca por serviços

Após ser chamado o método searchServices, o programa fica em um lock, até que o método serviceSearchCompleted seja automaticamente chamado quando todos os serviços suportados por um dispositivo móvel tenham sido encontrados. Para cada serviço encontrado é chamado o método servicesDiscovered.

As figuras 26 e 27 apresentam respectivamente o método servicesDiscovered que será chamado para cada serviço encontrado e o método serviceSearchCompleted quando todos os serviços já tiverem sido encontrados.

Para cada serviço, o método servicesDiscovered verifica se o nome é “OBEX Object Push”. Caso seja, adiciona-se a URL deste serviço ao vetor de serviços válidos. O método serviceSearchCompleted verifica o código de resposta recebido após a busca por serviços. Se tudo ocorreu bem, a variável de status recebe o valor SERVICE_SEARCH_COMPLETED.

```
public void servicesDiscovered(int transID, ServiceRecord[] servRecord) {
      for(int i = 0; i < servRecord.length; i++)
      {
         servs.addElement(servRecord[i].getConnectionURL(0,false));
         DataElement serviceNameElement = servRecord[i].getAttributeValue(0x0100);
         String _serviceName = (String)serviceNameElement.getValue();
         String serviceName = _serviceName.trim();
         if(serviceName.equals("OBEX Object Push"))
         {
            try
            {
               connectionURL = servRecord[i].getConnectionURL(0,false);
            } catch (Exception e){
            }
         }
      }
   }
```

Figura 26: Método servicesDiscovered

```
public void serviceSearchCompleted(int transID, int respCode) {
        String searchStatus = null;

        If (respCode ==
           DiscoveryListener.SERVICE_SEARCH_DEVICE_NOT_REACHABLE) {
           searchStatus = "SERVICE_SEARCH_DEVICE_NOT_REACHABLE\n";
        }
        else if (respCode == DiscoveryListener.SERVICE_SEARCH_NO_RECORDS) {
                    searchStatus = "SERVICE_SEARCH_NO_RECORDS\n";
             }
           else if (respCode == DiscoveryListener.SERVICE_SEARCH_COMPLETED)
                {
                     searchStatus = "SERVICE_SEARCH_COMPLETED\n";
                 }
              else if (respCode ==
           DiscoveryListener.SERVICE_SEARCH_TERMINATED)
                    {
                       searchStatus = "SERVICE_SEARCH_TERMINATED\n";
                    }
                    else if (respCode == DiscoveryListener.SERVICE_SEARCH_ERROR) {
                          searchStatus = "SERVICE_SEARCH_ERROR\n";
                        }
    }
```

Figura 27: Método serviceSearchCompleted

4 Conclusão

O que motivou este trabalho foi procurar solucionar um problema pelo qual todos os alunos passam, o da mudança repentina da sala de aula. Como solução a este problema, foram apresentadas três aplicações: uma em Java ME e duas em Java SE. O foco teórico deste trabalho nunca foi porém dissertar sobre Java SE, atualmente tão conhecida, mas sim apresentar a plataforma Java ME de forma didática e prática.

Os objetivos finais foram alcançados com sucesso e a aplicação pode ser posta em prática na Universidade Federal Fluminense para testes e então ser aplicada por todo o Brasil. Precisa-se testar a aplicação pois não foi possível realizar testes para garantir eficiência durante o uso excessivo da rede. Isto porque o emulador possui no máximo um processo executando no Windows e infelizmente não havia à disposição máquinas virtuais para contornar este problema.

4.1 Trabalhos Futuros

Apesar de atualmente existirem mais de 2 bilhões de dispositivos móveis ao redor do mundo, este número está caindo com a chegada dos smartphones e tablets com sistema Android, portanto uma versão desta aplicação para Android agregaria muito valor à aplicação como um todo.

Esta aplicação trata apenas o lado do aluno, contudo se o professor pudesse atualizar o banco de dados através de um dispositivo móvel, reduziria os riscos de os alunos não terem conhecimento de mudança de salas. Não seria necessário alterar muito a interface gráfica criada neste trabalho, bastaria fazer com que existisse uma tela que, ao invés de receber as informações do banco de dados, enviasse-as para o mesmo. O professor colocaria nesta tela as informações que desejasse alterar, como número de sala ou horário da aula.

A ferramente Java ME usada neste trabalho não precisa estar restrita à aplicações como a desenvolvida e implementada neste trabalho. É intuitivo idealizar aplicações como:

Google Wallet, que elimina a necessidade de carregar cartões de crédito na carteira, uma vez que estes estarão virtualmente presentes no dispositivo móvel (GOOGLE, 2011);

Qwerty Remote da Samsung para dispositivos com sistema Android, que serve para controlar remotamente Televisores que possuam navegadores de Internet;

Aplicações para medir o batimento cardíaco de pacientes através de uma pulseira que transmita via bluetooth as informações para o dispositivo móvel;

Aplicações voltadas para segurança, como por exemplo, quando uma residência invadida ou um automóvel for roubado, o dono possa ter conhecimento através de um rastreador no carro que envia o alerta para o dispositivo móvel do proprietário;

Nestes exemplos de aplicações citados, foram apresentadas aplicações de várias áreas diferentes, como comércio, entretenimento, saúde e segurança, cada uma tão importante quanto a outra. Oportunidades para novas aplicações móveis não faltam, basta apenas criatividade. Espera-se que este trabalho tenha servido como inspiração para novas aplicações voltadas para o mercado móvel.

REFERÊNCIAS BIBLIOGRÁFICAS

FIELDING, Roy T.; Gettys, James; Mogul, Jeffrey C.; Nielsen, Henrik Frystyk; Masinter, Larry; Leach, Paul J.; Berners-Lee. RFC 2616: Hypertext Transfer Protocol -- HTTP/1.1, 1999.

MONSON, Heidi. Bluetooth Technology and Implications. Disponível em: <sysopt.com>. Acesso em: fevereiro de 2009.

BB-ELEC, bb-elec. Disponível em : <http://www.bb-elec.com/tech_articles/80211_basics.asp>. Acesso em: junho de 2011.

ORACLE, Java ME Technology Overview. Disponível em: <http://www.oracle.com/technetwork/java/javame/java-me-overview-402920.html>.Acesso em: 6 de junho de 2011.

GOOGLE, Google Wallet. Disponível em: <http://www.google.com/wallet/>. Acesso em: 6 de junho de 2011.

TRAYLOR, Polly Schneider, 10 reasons to invest in new smartphones. Disponível em: <http://www.techrepublic.com/blog/10things/10-reasons-to-invest-in-new-smartphones/620>. Acesso em: 6 de junho de 2011.

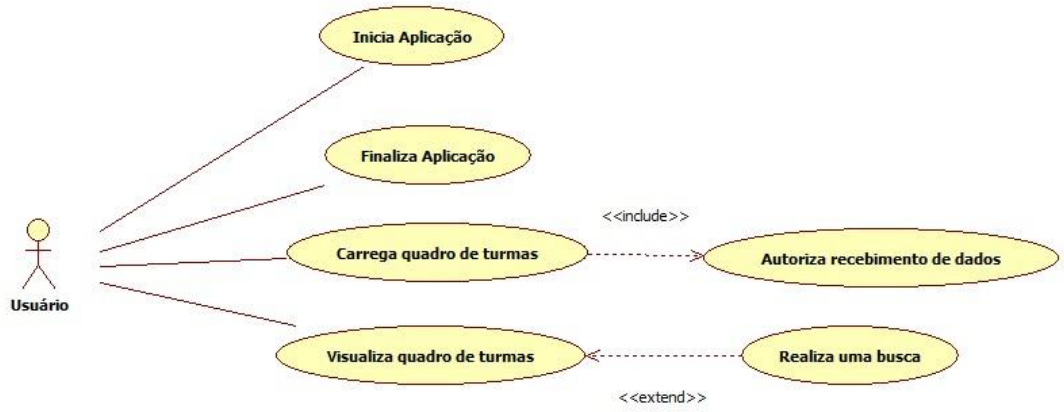
YUAN, Michael Juntao; Sharp, Kevin. Developing Scalable Series 40 Applications: A Guide for Java Developers. Estados Unidos da América: Ed. Pearson Education, 2004.

ORACLE, Overview (MID Profile). Documentação Java para dispositivos móveis. Disponível em: <http://download.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/index.html>. Acesso em: 7 de junho de 2011.

NOKIA, Nokia, Nokia N95. Especificações técnicas. Disponível em: <http://www.nokia.co.in/find-products/products/nokia-n95/technical-specifications>. Acesso em: 7 de junho de 2011.

APÊNDICE

Apêndice A – Diagram de Casos de Uso



Apêndice B – Caso de uso Visualizar Quadro de Turmas

Descrição:

suário visualiza quadro de turmas via celular

Pré-condições:

Usuário já inicializou a aplicação
IP do servidor já foi obtido pela aplicação
Aplicação já carregou o quadro de turmas na memória interna do celular

Pós-condições:

Nenhuma

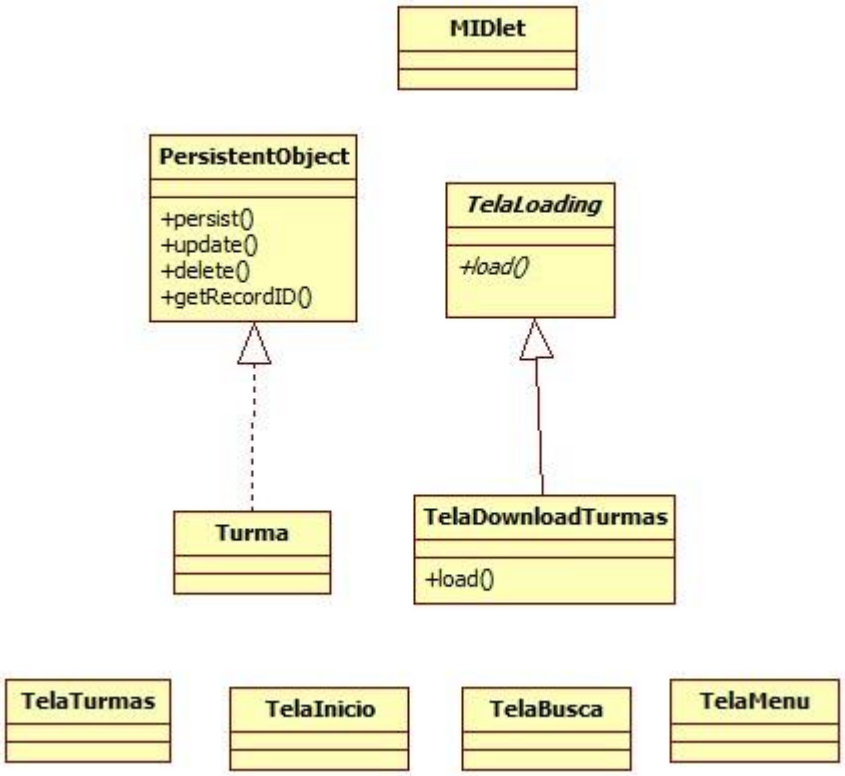
Cenário Típico:

2. Usuário autoriza o recebimento de dados via rede sem fio
3. Aplicação exibe a tela onde o usuário deverá selecionar o item da tela para exibir todo o quadro de turmas ou o item para fazer uma busca específica.
4. Usuário seleciona o item para exibir todo o quadro de turmas
5. Aplicação exibe o quadro de turmas em uma nova tela

Cenário Alternativo:

4.a.: Usuário seleciona o item para fazer uma busca específica
   1. Aplicação exibe a tela de busca onde o usuário deverá escolher qual(is) critério(s) utilizar
   2. Usuário define seus critérios e pressiona o botão de busca
   3. Aplicação exibe o quadro de turmas contendo apenas as turmas que satisfaçam o(s) critério(s) definidos pelo usuário.

Apêndice C – Diagrama de Classes da Aplicação Móvel



Apêndice D – Código Fonte da Aplicação Móvel

```
package main;

import javax.microedition.lcdui.*;
import javax.microedition.midlet.MIDlet;
import javax.microedition.rms.RecordStore;
import javax.microedition.rms.RecordStoreException;
import model.Turma;
import view.TelaInicio;

/**
 * @author yferreira
 */
public class MainMIDlet extends MIDlet {

    public static Display display = null;
    private static final int ALERT_TIMEOUT_MS = 3 * 1000;

    private Form form = null;
    private Command start = null;

    public MainMIDlet() {
        display = Display.getDisplay(this);

    }

    public void startApp() {

        try {
        RecordStore.deleteRecordStore(Turma.NOME_RECORD_STORE);
        System.out.println("Turmas Deletadas");
        } catch (RecordStoreException ex) {
```

```
            System.out.println("Nao conseguiu deletar as turmas");
            }


            TelaInicio telaInicio = new TelaInicio(this);
            display.setCurrent(telaInicio);
        }

        public void pauseApp() {
        }

        public void destroyApp(boolean unconditional) {
        }

        public void closeApp() {
            destroyApp(true);
            notifyDestroyed();
        }

        public static void showError(String message, Displayable displayable) {
            Alert alert = new Alert(null);
            alert.setTimeout(ALERT_TIMEOUT_MS);
            alert.setTitle("Error");
            alert.setString(message);
            alert.setType(AlertType.ERROR);
            showAlert(alert, displayable);
        }

        private static void showAlert(Alert alert, Displayable displayable) {
            display.setCurrent(alert, displayable);
        }
    }
```

```
package model;

/**
 *
 * @author yferreira
 */
public interface PersistentObject {

    public int getRecordID();
    public void persist();
    public void update();
    public void delete();

}

package model;

import model.PersistentObject;
import util.QueryList;
import util.QueryRecordStore;

/**
 *
 * @author Yuri
 */
public class Turma implements PersistentObject {

    public static final String NOME_RECORD_STORE = "Turmas";
    public static final int CAMPO_ID = 0;
    public static final int CAMPO_CODIGO_DISCIPLINA = 1;
    public static final int CAMPO_NOME_DISCIPLINA = 2;
    public static final int CAMPO_CODIGO_TURMA = 3;
    public static final int CAMPO_NOME_PROFESSOR = 4;
```

```
    public static final int CAMPO_NUMERO_SALA = 5;

    private String id = null;
    private String codigoDisciplina = null;
    private String nomeDisciplina = null;
    private String codigoTurma = null;
    private String nomeProfessor = null;
    private String numeroSala = null;

    public Turma(){

    }

    public Turma(String codigoDisciplina, String nomeDisciplina, String codigoTurma, String
nomeProfessor, String numeroSala){
        QueryRecordStore disciplineRecordStore = new
QueryRecordStore(NOME_RECORD_STORE);
        disciplineRecordStore.openRecordStore(true);
        id = Integer.toString(disciplineRecordStore.getNumRecords() + 1);
        disciplineRecordStore.closeRecordStore();

        this.codigoDisciplina = codigoDisciplina;
        this.nomeDisciplina = nomeDisciplina;
        this.codigoTurma = codigoTurma;
        this.nomeProfessor = nomeProfessor;
        this.numeroSala = numeroSala;
    }

    public String getClassNumber() {
        return codigoTurma;
    }

    public void setClassNumber(String classNumber) {
```

```
    this.codigoTurma = classNumber;
}

public String getCode() {
    return codigoDisciplina;
}

public void setCode(String code) {
    this.codigoDisciplina = code;
}

public String getName() {
    return nomeDisciplina;
}

public void setName(String name) {
    this.nomeDisciplina = name;
}

public String getProfessorName() {
    return nomeProfessor;
}

public void setProfessorName(String professorName) {
    this.nomeProfessor = professorName;
}

public String getNumeroSala() {
    return numeroSala;
}

public void setNumeroSala(String numeroSala) {
    this.numeroSala = numeroSala;
```

```
    }

    public int getRecordID(){
        QueryList queryList = new QueryList(1, id, 0, false);
        QueryRecordStore disciplineRecordStore = new
QueryRecordStore(Turma.NOME_RECORD_STORE);
        disciplineRecordStore.openRecordStore(true);
        disciplineRecordStore.findFirst(queryList);
        int recordID =  disciplineRecordStore.getCurrentRecordID();
        disciplineRecordStore.closeRecordStore();
        return recordID;
    }

    public void persist(){
        QueryRecordStore disciplineRecordStore = new
QueryRecordStore(Turma.NOME_RECORD_STORE);
        disciplineRecordStore.openRecordStore(true);

        //missing picture
        String record = new String(codigoDisciplina + ";" + nomeDisciplina + ";" +
codigoTurma + ";"
                +  nomeProfessor + ";"  + numeroSala + ";");
        disciplineRecordStore.addRecord(record.getBytes());
        System.out.println("***************RecordStoreSize: " +
disciplineRecordStore.getNumRecords());
        disciplineRecordStore.closeRecordStore();
    }

    public void update(){
        String disciplineData = new String(codigoDisciplina + ";" + nomeDisciplina + ";" +
codigoTurma + ";"
                +  nomeProfessor + ";" + numeroSala + ";");
        int recordId = getRecordID();
```

```
        QueryRecordStore disciplineRecordStore = new
QueryRecordStore(Turma.NOME_RECORD_STORE);
        disciplineRecordStore.openRecordStore(true);
        disciplineRecordStore.set(recordId, disciplineData);
        disciplineRecordStore.closeRecordStore();
    }

    public void delete(){
        int recordId = getRecordID();
        QueryRecordStore disciplineRecordStore = new
QueryRecordStore(Turma.NOME_RECORD_STORE);
        disciplineRecordStore.openRecordStore(true);
        disciplineRecordStore.deleteRecord(recordId);
        disciplineRecordStore.closeRecordStore();
    }
}

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package conexaoServidor;

/**
 *
 * @author Yuri
 */
import java.io.IOException;
import javax.microedition.io.Connector;
import javax.microedition.io.StreamConnection;

/*
```

```
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

/**
 *
 * @author yferreira
 */
public class ConexaoThread implements Runnable {

    private StreamConnection conn= null;
    private String serverURL= null;

    public ConexaoThread(String serverURL){
        this.serverURL = serverURL;
    }

    public StreamConnection getConnection(){
        return conn;
    }

    public void run(){
        try {
            conn = (StreamConnection) Connector.open(serverURL);
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }
}

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
```

```
 */

package conexaoServidor;

/**
 *
 * @author Yuri
 */
public class ConexaoServidorTurma extends ConexaoServidor {

    public static String getUrl(String action){
        //String lochost = ConexaoServidor.getIpAddress();
        String url = "http://" + "192.168.0.187" +
":8080/ServidorQuadroTurmas/abreconta?action=" + action;
        return url;
    }
}

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package conexaoServidor;
import java.io.DataInputStream;
import java.io.IOException;
import javax.microedition.io.HttpConnection;
import javax.microedition.rms.RecordStore;
import javax.microedition.rms.RecordStoreException;

/**
 *
 * @author yferreira
```

```
 */
public class ConexaoServidor {

    private static String ipAddress = null;

    public static String getIpAddress() {
        return ipAddress;
    }

    public static void saveIp(String ip){
        RecordStore ipRS;
        try {
            RecordStore.deleteRecordStore("IPADDRESS");
        } catch (RecordStoreException ex) {
            ex.printStackTrace();
        }
        try {
            ipRS = RecordStore.openRecordStore("IPADDRESS", true);
            ipRS.addRecord(ipAddress.getBytes(), 0, ipAddress.getBytes().length);
            ipRS.closeRecordStore();
        } catch (RecordStoreException ex) {
            ex.printStackTrace();
        }
    }

    public static void setIpAddress(String ipAddress) {
        ConexaoServidor.ipAddress = ipAddress;
    }

    public static String connect(String url){
         HttpConnection httpConn = null;
         String response = null;
            try {
```

```
   ConexaoThread connectionThread = new ConexaoThread(url);
   Thread t = new Thread(connectionThread);
   t.start();

   long initialTime = System.currentTimeMillis();
   long timeElapsed = 0;
   do{
     timeElapsed = System.currentTimeMillis() - initialTime;
     System.out.println("Tempo: " + timeElapsed);
     httpConn =  (HttpConnection) connectionThread.getConnection();
   }while (httpConn == null && timeElapsed < 15000);
   t.interrupt();

   System.out.println("Tempo: " + timeElapsed);
   if (httpConn != null){
      //httpConn = (HttpConnection) Connector.open(url);
      httpConn.setRequestMethod(HttpConnection.GET);
      httpConn.setRequestProperty("User-Agent",
            "Profile/MIDP-1.0 Confirguration/CLDC-1.0");

      StringBuffer sb = new StringBuffer();
      DataInputStream is = httpConn.openDataInputStream();
      int chr;
      while ((chr = is.read()) != -1) {
         sb.append((char) chr);
      }
      response = sb.toString();
   }else{
      return null;
   }
} catch (Exception ex){
   System.out.println(ex.getMessage());
} finally {
```

```
            try {
                httpConn.close();
            } catch (IOException ex) {
                ex.printStackTrace();
            }finally{
                return response;
            }
        }
    }
}

package conexaoServidor;

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
import conexaoServidor.ConexaoThread;
import main.MainMIDlet;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.util.Enumeration;
import java.util.Vector;
import javax.bluetooth.BluetoothStateException;
import javax.bluetooth.DataElement;
import javax.bluetooth.DeviceClass;
import javax.bluetooth.DiscoveryAgent;
import javax.bluetooth.DiscoveryListener;
import javax.bluetooth.LocalDevice;
import javax.bluetooth.RemoteDevice;
import javax.bluetooth.ServiceRecord;
import javax.bluetooth.UUID;
```

```
import javax.microedition.io.StreamConnection;
import javax.microedition.lcdui.Form;

/**
 * @author yferreira
 */
public class ConexaoBluetooth implements DiscoveryListener {

    private LocalDevice localDevice = null;
    private String serverURL = null;
    private Form form = null;
    private MainMIDlet midlet= null;
    private Vector devs = new Vector();
    private static Object lock = new Object();
    private StreamConnection conn = null;
    private DiscoveryAgent agent = null;

    /*public static final String BT_NOT_ACTIVE = "BT_NOT_ACTIVE";
    public static final String BT_SERVER_NOT_FOUND = "BT_SERVER_NOT_FOUND";
    public static final String CONNECTION_ERROR = "CONNECTION_ERROR";*/

    public static final String SERVER_ADDRESS = "0011F6050DC5";

    public ConexaoBluetooth(Form form, MainMIDlet midlet) {
        this.form = form;
        this.midlet = midlet;
    }

    public static String getBtAddress(){
        try {
            return LocalDevice.getLocalDevice().getBluetoothAddress();
        } catch (BluetoothStateException ex) {
            ex.printStackTrace();
```

```
        return null;
    }
}

public void prepare() {
    try {
        localDevice = LocalDevice.getLocalDevice();
        //form.append("\nAddress: " + localDevice.getBluetoothAddress());
    } catch (BluetoothStateException ex) {
        form.append("Por favor, ative o bluetooth no celular\ne reinicie a aplicação.");
        midlet.closeApp();
        return;
    }
    agent = localDevice.getDiscoveryAgent();
}

public boolean startDeviceSearch() {
    //form.append("\nProcurando o servidor....");
    try {
        agent.startInquiry(DiscoveryAgent.GIAC, this);
    } catch (BluetoothStateException ex) {
        ex.printStackTrace();
    }
    try {
        synchronized (lock) {
            lock.wait();
        }
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    //form.append("Procura terminada.");

    return !(devs.size() == 0);
```

```
}

public void searchServices(UUID[] uuidSet, RemoteDevice rd) {
   try {
      agent.searchServices(null, uuidSet, rd, this);
   } catch (BluetoothStateException ex) {
      ex.printStackTrace();
   }
   try {
      synchronized (lock) {
         lock.wait();
      }
   } catch (InterruptedException e) {
      e.printStackTrace();
   }
}

public void connect() {
   conn = null;
   //form.append("\nConectando...");
   ConexaoThread connectionThread = new ConexaoThread(serverURL);
   Thread t = new Thread(connectionThread);
   t.start();

   do {
      conn = connectionThread.getConnection();
   } while (conn == null);

   //acho q nem precisa disso pq o run() ja vai ter acabado
   t.interrupt();
}

public String getIpAddress() {
```

```
boolean found = false;
UUID[] uuidSet = {new UUID("1101", true)};
RemoteDevice rd = null;

//while (!found){
    prepare();
    if (!startDeviceSearch()) {
        return null;
        //form.append("\nServidor não encontrado.\nProcurando novamente...");
        //continue;
    }

    for (int i = 0; i < devs.size(); i++) {
        rd = (RemoteDevice) devs.elementAt(i);
        if (rd.getBluetoothAddress().compareTo(SERVER_ADDRESS) == 0){
            found = true;
            break;
        }
    }
    if (!found) {
        //form.append("Servidor não encontrado.\nProcurando novamente...");
        return null;
    }
// }

 //do {
    searchServices(uuidSet, rd);
    if (serverURL == null) {
        //form.append("\nFalha na conexão. Reconectando...");
        //continue;
        return null;
    } else {
        connect();
```

```
            try {
                if (conn == null) {
                    //form.append("\nFalha na conexão. Reconectando...");
                    //continue;
                     return null;
                }
                //form.append("\nConexão Bem-Sucedida");

                DataOutputStream dout = conn.openDataOutputStream();
                DataInputStream din = conn.openDataInputStream();
                String str = "ola server";
                dout.writeUTF(str);
                dout.flush();
                dout.close();

                String response = din.readUTF();
                din.close();
                //form.append("\nIP recebido: " + response);
                conn.close();

                //form.append("\nConexão Fechada");
                return response;

            } catch (IOException ex) {
            }
        }
         //break;
    //} while (true);
    return null;
}

public void pauseApp() {
}
```

```
    public void destroyApp(boolean unconditional) {
    }

    public void deviceDiscovered(RemoteDevice dev, DeviceClass arg1) {
        devs.addElement(dev);
    }

    public void servicesDiscovered(int arg0, ServiceRecord[] serv) {
        for (int i = 0; i < serv.length; i++) {
            String address = null;
            address = serv[i].getHostDevice().getBluetoothAddress();
            if (address.compareTo(SERVER_ADDRESS) == 0) {

                DataElement protocolDescriptorList = serv[i].getAttributeValue(0x0004);
                Enumeration e = (Enumeration) protocolDescriptorList.getValue(); // DATSEQ |
DATALT
                e.nextElement(); // L2CAP (ignored)
                DataElement protocolDescriptorRFCOMM = (DataElement) e.nextElement();
                e = (Enumeration) protocolDescriptorRFCOMM.getValue(); // DATSEQ
                e.nextElement(); // UUID (ignored)
                DataElement channelRFCOMM = (DataElement) e.nextElement();
                long channel;
                if (DataElement.U_INT_8 == channelRFCOMM.getDataType() ||
                        DataElement.U_INT_16 == channelRFCOMM.getDataType() ||
                        DataElement.INT_16 == channelRFCOMM.getDataType()) {
                    byte[] bytes = (byte[]) channelRFCOMM.getValue();
                    channel = bytes[0]; // Range (decimal): 1 up to 30
                } else {
                    channel = channelRFCOMM.getLong(); // U_INT_1
                }

                StringBuffer nameBuffer = new StringBuffer(5 + 3 + 12 + 1 + 2 + 19 + 14 + 13);
```

```
                nameBuffer.append("btspp");
                nameBuffer.append("://");
                nameBuffer.append(serv[i].getHostDevice().getBluetoothAddress());
                nameBuffer.append(":");
                nameBuffer.append(channel);
                nameBuffer.append(";authenticate=false");
                nameBuffer.append(";encrypt=false");
                nameBuffer.append(";master=false");

                serverURL = nameBuffer.toString();
                break;
            }
        }
    }

    public void serviceSearchCompleted(int arg0, int arg1) {
        synchronized (lock) {
            lock.notify();
        }
    }

    public void inquiryCompleted(int arg0) {
        synchronized (lock) {
            lock.notify();
        }
    }
}

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package view;
```

```
import util.QueryRecordStore;
import main.MainMIDlet;
import javax.microedition.lcdui.Command;
import conexaoServidor.*;
import javax.microedition.lcdui.Displayable;

/**
 *
 * @author Yuri
 * Tela que irá obter o IP do servidor.
 * ATENÇÂO: Prefiro que o ip sejá definido em tempo de compilação. Se o IP da Wi-Fi do nosso departamento não for fixo, alterações devem ser feitas
 */
public class AquisicaoIP extends TelaLoading{

    private Command start = null;
    //private Command finish = null;
    private MainMIDlet midlet = null;
    //private boolean finishOk = false;
    public AquisicaoIP(MainMIDlet midlet) {
        super("Obtendo o IP do servidor");
        nextScreen = new TelaDownloadTurmas(this);
        setNextScreen(nextScreen);
    }

    public void run() {
        while (!isShown()) {
            try {
                Thread.sleep(1000);
            } catch (InterruptedException ex) {
                ex.printStackTrace();
            }
```

```
        }
        if (load()) {
            MainMIDlet.display.setCurrent(nextScreen);

        } else {
            midlet.closeApp();
        }
    }

    public boolean load() {
        //ConexaoBluetooth btClient = new ConexaoBluetooth(this, midlet);
        //String ipAddress = btClient.getIpAddress();
        String ipAddress = "192.168.0.187";
        if (ipAddress == null){
            append("Conexão falhou. Reinicie a aplicação");
            return false;
        }
        ConexaoServidor.setIpAddress(ipAddress);
        ConexaoServidor.saveIp(ipAddress);
        return true;
    }
}

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package view;

import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Displayable;
```

```
import javax.microedition.lcdui.Form;
import javax.microedition.lcdui.TextField;
import main.MainMIDlet;
import model.Turma;
import util.QueryList;

/**
 *
 * @author Yuri
 * Tela onde o usuário poderá filtrar sua busca
 */
public class TelaBusca extends Form implements CommandListener {

    private Command voltar = null;
    private Command buscar = null;
    private Displayable telaAnterior = null;
    private TextField codigoDisciplinaField = null;
    private TextField nomeDisciplinaField = null;
    private TextField nomeProfessorField = null;

    public TelaBusca(Displayable telaAnterior){
        super("Escolha os campos");
        this.telaAnterior = telaAnterior;

        append("Codigo da Disciplina");
        codigoDisciplinaField = new TextField(null, null, 30, TextField.NON_PREDICTIVE);
        append(codigoDisciplinaField);
        append("Nome da Disciplina");
        nomeDisciplinaField = new TextField(null, null, 30, TextField.NON_PREDICTIVE);
        append(nomeDisciplinaField);
        append("Nome do Professor");
        nomeProfessorField = new TextField(null, null, 30, TextField.NON_PREDICTIVE);
        append(nomeProfessorField);
```

```
        voltar = new Command("Voltar", Command.BACK, 1);
        buscar = new Command("Buscar", Command.OK, 0);
        addCommand(voltar);
        addCommand(buscar);
        setCommandListener(this);

    }

    public void commandAction(Command cmnd, Displayable dsplbl) {
        if (cmnd == voltar){
            MainMIDlet.display.setCurrent(telaAnterior);
        }else{
            if (cmnd == buscar){
                QueryList queryList = null;
                if (codigoDisciplinaField.getString().compareTo("") != 0) {
                    System.out.println("********Tem codigo");
                    queryList = new QueryList(Turma.CAMPO_CODIGO_DISCIPLINA,
codigoDisciplinaField.getString(), 0, false);
                    if (nomeDisciplinaField.getString().compareTo("") != 0){
                        System.out.println("********Tem nome da disciplina");
                        queryList.addCriteria(new  QueryList(Turma.CAMPO_NOME_DISCIPLINA,
nomeDisciplinaField.getString(), 0, false));
                    }
                    if (nomeProfessorField.getString().compareTo("") != 0){
                        System.out.println("********Tem nome do professor");
                        queryList.addCriteria(new  QueryList(Turma.CAMPO_NOME_PROFESSOR,
nomeProfessorField.getString(), 0, false));
                    }
```

```
            }else{
                if (nomeDisciplinaField.getString().compareTo("") != 0){
                    System.out.println("********Tem nome da disciplina");
                    queryList = new QueryList(Turma.CAMPO_NOME_DISCIPLINA,
nomeDisciplinaField.getString(), 0, false);
                    if (nomeProfessorField.getString().compareTo("") != 0){
                        System.out.println("********Tem nome do professor");
                        queryList.addCriteria(new
QueryList(Turma.CAMPO_NOME_PROFESSOR, nomeProfessorField.getString(), 0,
false));
                    }
                }else{
                    if (nomeProfessorField.getString().compareTo("") != 0){
                        System.out.println("********Tem nome do professor");
                        queryList = new QueryList(Turma.CAMPO_NOME_PROFESSOR,
nomeProfessorField.getString(), 0, false);
                    }
                }
            }

            TelaTurmas telaTurmas = new TelaTurmas(this, queryList);
            MainMIDlet.display.setCurrent(telaTurmas);
        }
    }
  }
}

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package view;
```

```
import conexaoServidor.ConexaoServidorTurma;
import model.Turma;
import main.MainMIDlet;
import javax.microedition.lcdui.Alert;
import javax.microedition.lcdui.AlertType;
import javax.microedition.lcdui.Displayable;
import util.QueryRecordStore;

/**
 *
 * @author Yuri
 * Tela que irá esperar pelo download das informações das turmas vindas do servidor
 */
public class TelaDownloadTurmas extends TelaLoading{


    public TelaDownloadTurmas(Displayable telaAnterior){
        super("Baixando Quadro de Turmas");
        setNextScreen(new TelaMenu(telaAnterior));
        setPrevScreen(telaAnterior);
    }
    public void run() {
        while (!isShown()){
            try {
                Thread.sleep(1000);
            } catch (InterruptedException ex) {
                ex.printStackTrace();
            }
        }
        if (load()) {
            MainMIDlet.display.setCurrent(nextScreen);
```

```
        } else {
            Alert alert = new Alert(null, "Conexão falhou. Tente Novamente", null,
AlertType.ERROR);
            alert.setTimeout(3000);
            MainMIDlet.display.setCurrent(alert, prevScreen);
        }
    }

    public boolean load() {

        String url = ConexaoServidorTurma.getUrl("turmas");
        System.out.println("URL: " + url);
        String listaTurmas = ConexaoServidorTurma.connect(url);
        System.out.println("ListTurmas: " + listaTurmas);
        if (listaTurmas == null){
            return false;
        }
        String stringCorrente = listaTurmas;
        while (true) {

            String codigoDisciplina = QueryRecordStore.getField(stringCorrente,
Turma.CAMPO_CODIGO_DISCIPLINA);
            String nomeDisciplina = QueryRecordStore.getField(stringCorrente,
Turma.CAMPO_NOME_DISCIPLINA);
            String codigoTurma = QueryRecordStore.getField(stringCorrente,
Turma.CAMPO_CODIGO_TURMA);
            String nomeProfessor = QueryRecordStore.getField(stringCorrente,
Turma.CAMPO_NOME_PROFESSOR);
            String numeroSala = QueryRecordStore.getField(stringCorrente,
Turma.CAMPO_NUMERO_SALA);

            System.out.println("Codigo da Disciplina: " + codigoDisciplina);
            System.out.println("Nome da Disciplina: " + nomeDisciplina);
```

```
            System.out.println("Codigo da Turma: " + codigoTurma);
            System.out.println("Nome do Professor: " + nomeProfessor);
            System.out.println("Numero da Sala: " + numeroSala);

            Turma turma = new Turma(codigoDisciplina, nomeDisciplina, codigoTurma,
nomeProfessor, numeroSala);
            turma.persist();

            int barIndex = stringCorrente.indexOf("/");
            System.out.println("StringCorrent: " + stringCorrente.length());
            System.out.println("barIndex: " + barIndex);
            if (barIndex + 1 == stringCorrente.length()) {
                break;
            }
            stringCorrente = stringCorrente.substring(barIndex + 1);
            System.out.println("stringCorrent: " + stringCorrente);
        }
        return true;
    }
}

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package view;

import main.MainMIDlet;
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Displayable;
import javax.microedition.lcdui.Form;
```

```
import javax.microedition.midlet.MIDlet;

/**
 *
 * @author Yuri
 * Tela inicial de boas vindas
 */
public class TelaInicio extends Form implements CommandListener {

    private Command start = null;
    private MainMIDlet midlet;

    public TelaInicio(MainMIDlet midlet){
        super(null);
        this.midlet = midlet;
        append("Quadro de Turmas Movel");
        start = new Command("Iniciar", Command.OK, 0);
        addCommand(start);
        setCommandListener(this);

    }

    public void commandAction(Command c, Displayable d) {
        if (c == start){
            AquisicaoIP aquisicaoIP = new AquisicaoIP(midlet);
            MainMIDlet.display.setCurrent(aquisicaoIP);
        }
    }

}

/*
 * To change this template, choose Tools | Templates
```

```
 * and open the template in the editor.
 */

package view;

import javax.microedition.lcdui.Displayable;
import javax.microedition.lcdui.Form;
import javax.microedition.lcdui.Gauge;

/**
 *
 * @author Yuri
 * Tela que irá exibir uma espera em andamento
 */
public abstract class TelaLoading extends Form implements Runnable{

    protected Gauge autoGauge= null;
    protected Displayable prevScreen = null;
    protected Displayable nextScreen = null;
    protected Thread t= null;

    public TelaLoading(String label){
        super("Aguarde...");
        autoGauge = new Gauge(label, false, Gauge.INDEFINITE,
Gauge.CONTINUOUS_RUNNING);
        append(autoGauge);
        t = new Thread(this);
        t.start();
    }

    public Displayable getNextScreen() {
        return nextScreen;
    }
```

```
    public void setNextScreen(Displayable nextScreen) {
        this.nextScreen = nextScreen;
    }

    public Displayable getPrevScreen() {
        return prevScreen;
    }

    public void setPrevScreen(Displayable prevScreen) {
        this.prevScreen = prevScreen;
    }

    public abstract void run();

    public abstract boolean load();
}

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package view;

import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Displayable;
import javax.microedition.lcdui.List;
import main.MainMIDlet;

/**
 *
```

```
 * @author Yuri
 * Tela que irá exibir o menu principal
 */
public class TelaMenu extends List implements CommandListener{

    private Displayable telaAnterior = null;
    public static final String LISTAR_TUDO = "Ver quadro de turmas";
    public static final String BUSCAR = "Buscar Turma";

    public TelaMenu(Displayable telaAnterior){
        super(null,List.IMPLICIT);
        this.telaAnterior = telaAnterior;
        append(LISTAR_TUDO, null);
        append(BUSCAR, null);

        setCommandListener(this);
    }
    public void commandAction(Command cmnd, Displayable dsplbl) {
        String escolha = getString(getSelectedIndex());
        if (escolha.compareTo(LISTAR_TUDO) == 0){
            TelaTurmas telaTurmas = new TelaTurmas(this, null);
            MainMIDlet.display.setCurrent(telaTurmas);
        }else{
            if (escolha.compareTo(BUSCAR) == 0){
                TelaBusca telaBusca = new TelaBusca(this);
                MainMIDlet.display.setCurrent(telaBusca);
            }
        }
    }

}

/*
```

```
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package view;

import java.util.Vector;
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Displayable;
import javax.microedition.lcdui.Form;
import javax.microedition.rms.RecordEnumeration;
import main.MainMIDlet;
import model.Turma;
import util.QueryList;
import util.QueryRecordStore;

/**
 *
 * @author Yuri
 * Tela que irá exibir as turmas do semestre corrente
 * As informações estão no record store de turmas, portanto será necessário obter cada registro (turmasRecordStore.getRecord)
 */
public class TelaTurmas extends Form implements CommandListener {

    private Command voltar = null;
    private Displayable telaAnterior = null;
    private QueryList queryList;

    public TelaTurmas(Displayable telaAnterior, QueryList queryList){
        super(null);
        this.queryList = queryList;
```

```
        this.telaAnterior = telaAnterior;
        voltar = new Command("Voltar", Command.BACK, 0);
        addCommand(voltar);
        setCommandListener(this);

        QueryRecordStore turmasRecordStore = new
QueryRecordStore(Turma.NOME_RECORD_STORE);
        turmasRecordStore.openRecordStore(true);
        if (queryList == null){
           RecordEnumeration recordEnum = turmasRecordStore.enumerateRecords();
           String record = null;
           System.out.println("NumRecords: " + turmasRecordStore.getNumRecords());
           for (int i = 1; i <= turmasRecordStore.getNumRecords(); i++)
           {
              record = new String(turmasRecordStore.getRecord(i));
              System.out.println("Record :" + i + "-" + record);

              String codigoDisciplina = QueryRecordStore.getField(record,
Turma.CAMPO_CODIGO_DISCIPLINA);
              String nomeDisciplina = QueryRecordStore.getField(record,
Turma.CAMPO_NOME_DISCIPLINA);
              String codigoTurma = QueryRecordStore.getField(record,
Turma.CAMPO_CODIGO_TURMA);
              String nomeProfessor = QueryRecordStore.getField(record,
Turma.CAMPO_NOME_PROFESSOR);
              String numeroSala = QueryRecordStore.getField(record,
Turma.CAMPO_NUMERO_SALA);

              append(codigoDisciplina + " " + nomeDisciplina + " " + codigoTurma + " " +
nomeProfessor + " " + numeroSala +  "\n");
           }
           turmasRecordStore.closeRecordStore();
```

```
      }else{

         Vector records = turmasRecordStore.searchRecords(queryList);
         for (int i = 0; i < records.size(); i++){
            String record = (String)records.elementAt(i);
            System.out.println("Turma record: " + record);

            String codigoDisciplina = QueryRecordStore.getField(record,
Turma.CAMPO_CODIGO_DISCIPLINA);
            String nomeDisciplina = QueryRecordStore.getField(record,
Turma.CAMPO_NOME_DISCIPLINA);
            String codigoTurma = QueryRecordStore.getField(record,
Turma.CAMPO_CODIGO_TURMA);
            String nomeProfessor = QueryRecordStore.getField(record,
Turma.CAMPO_NOME_PROFESSOR);
            String numeroSala = QueryRecordStore.getField(record,
Turma.CAMPO_NUMERO_SALA);

            append(codigoDisciplina + " " + nomeDisciplina + " " + codigoTurma + " " +
nomeProfessor + " " + numeroSala);
         }
         turmasRecordStore.closeRecordStore();
      }
   }

   public void commandAction(Command c, Displayable d) {
     if (c == voltar){
        MainMIDlet.display.setCurrent(telaAnterior);
     }
   }

}
```

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package util;

/**
 *
 * @author Yuri
 * Lista de consultas com diferentes critérios.
 */
public class QueryList{
    private int order;
    private String refVal;
    private int relation;
    private boolean isPrefix;
    private QueryList nextQueryList;
    private QueryList tail;

    private static final int EQUALS = 0;
    private static final int LESS_THAN = 1;
    private static final int GREATER_THAN = 2;
    private static final int LESS_EQUAL = 3;
    private static final int GREATER_EQUAL = 4;
    private static final int DIFFERENT = 5;

    public QueryList(int order, String refval, int relation, boolean isPrefix){
        this.order = order;
        this.refVal = refval;
        this.relation = relation;
        nextQueryList = null;
        this.isPrefix = isPrefix;
```

```
    tail = this;
}

public void addCriteria(QueryList querylist){
    tail.nextQueryList = querylist;
    tail = nextQueryList;
}

/*public void removeLastCriteria(){
    QueryList pointer = this;
    QueryList  currentQueryList = this;
    while(pointer.nextQueryList != tail){
        pointer = currentQueryList.nextQueryList;
        currentQueryList = nextQueryList;
    }

    tail = pointer;
    pointer.nextQueryList = null;
}*/

public boolean hasPrefix(){
    return isPrefix;
}

private boolean hasNext(String record){
    if (nextQueryList != null){
        return nextQueryList.match(record);
    }else{
        return true;
    }
}

public boolean match(String record){
```

```
boolean matches = true;

while (matches){
   String currentSubstring= record;
   //procura o order-ésimo campo do record
   int begin = 0 ;
   int index = 0;

   for (int i = 0; i < order; i++){
      currentSubstring = record.substring(begin);
      index = currentSubstring.indexOf(";");
      begin += index + 1;
   }

   currentSubstring = currentSubstring.substring(0, index);

   switch (relation) {
      case EQUALS:    if (!isPrefix){
                        if (currentSubstring.compareTo(refVal) == 0){
                           return hasNext(record);
                        }else{
                           matches = false;
                           break;
                        }
                     }else{
                        if (currentSubstring.startsWith(refVal)){
                           return hasNext(record);
                        }else{
                           matches = false;
                           break;
                        }
                     }
      case LESS_THAN: if (currentSubstring.compareTo(refVal) < 0){
```

```
                    return hasNext(record);
                }else{
                    matches = false;
                    break;
                }
        case GREATER_THAN: if (currentSubstring.compareTo(refVal) > 0){
                    return hasNext(record);
                }else{
                    matches = false;
                    break;
                }
        case LESS_EQUAL:    if (currentSubstring.compareTo(refVal) <= 0){
                    return hasNext(record);
                }else{
                    matches = false;
                    break;
                }
        case GREATER_EQUAL: if (currentSubstring.compareTo(refVal) >= 0){
                    return hasNext(record);
                }else{
                    matches = false;
                    break;
                }
        case DIFFERENT: if (currentSubstring.compareTo(refVal) != 0){
                    return hasNext(record);
                }else{
                    matches = false;
                    break;
                }
      }
    }
    return false;
}
```

```
}

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package util;

import java.util.Vector;
import javax.microedition.rms.InvalidRecordIDException;
import javax.microedition.rms.RecordEnumeration;
import javax.microedition.rms.RecordListener;
import javax.microedition.rms.RecordStore;
import javax.microedition.rms.RecordStoreException;
import javax.microedition.rms.RecordStoreNotOpenException;

/**
 *
 * @author yferreira
 * Record Store aprimorado para que sejam feitas buscas nele
 */
public class QueryRecordStore implements RecordListener {

    private String recordStoreName= null;
    private RecordStore recordStore;
    private RecordEnumeration recordEnum= null;
    private int currentRecordID = 0;
    private QueryList queryList= null;

    public static final String NOT_FOUND = null;

    public QueryRecordStore(String recordStoreName){
```

```
    this.recordStoreName = recordStoreName;
    currentRecordID = 1;
}

public static String getField(String record, int fieldOrder){
    String currentSubstring = record;
    int begin = 0;
    int index = 0;

    for (int i = 0; i < fieldOrder; i++){
        currentSubstring = record.substring(begin);
        index = currentSubstring.indexOf(";");
        begin += index + 1;
    }
    currentSubstring = currentSubstring.substring(0, index);
    return currentSubstring;
}

public int getNumRecords(){
    try {
        return recordStore.getNumRecords();
    } catch (RecordStoreNotOpenException ex) {
        ex.printStackTrace();
        return -1;
    }
}

//duvido q isso seja necessario.
public RecordEnumeration enumerateRecords(){
    try {
        return recordStore.enumerateRecords(null, null, true);
    } catch (RecordStoreNotOpenException ex) {
        ex.printStackTrace();
```

```
        return null;
    }
}

public void openRecordStore(boolean create){
    try {
        recordStore = RecordStore.openRecordStore(recordStoreName, create);
        recordStore.addRecordListener(this);
    } catch (RecordStoreException ex) {
        ex.printStackTrace();
    }
}

public void closeRecordStore(){
    try {
        recordStore.closeRecordStore();
    } catch (RecordStoreNotOpenException ex) {
        ex.printStackTrace();
    } catch (RecordStoreException ex) {
        ex.printStackTrace();
    }
}

public void set(int recordID, String data){
    try {
        recordStore.setRecord(recordID, data.getBytes(), 0, data.length());
    } catch (RecordStoreNotOpenException ex) {
        ex.printStackTrace();
    } catch (InvalidRecordIDException ex) {
        ex.printStackTrace();
    } catch (RecordStoreException ex) {
        ex.printStackTrace();
    }
```

```
}

public byte[] getRecord(int ID){
   byte[] record = null;
   try {
      record =  recordStore.getRecord(ID);
   } catch (RecordStoreNotOpenException ex) {
      ex.printStackTrace();
   } catch (InvalidRecordIDException ex) {
      ex.printStackTrace();
   } catch (RecordStoreException ex) {
      ex.printStackTrace();
   }finally{
      return record;
   }
}

public void addRecord(byte[] data){
   try {
      recordStore.addRecord(data, 0, data.length);
   } catch (RecordStoreNotOpenException ex) {
      ex.printStackTrace();
   } catch (RecordStoreException ex) {
      ex.printStackTrace();
   }
}

public void deleteRecord(int recordID){
   try {
      recordStore.deleteRecord(recordID);
   } catch (RecordStoreNotOpenException ex) {
      ex.printStackTrace();
   } catch (InvalidRecordIDException ex) {
```

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


    public void updateRecord(int recordID,byte[] recordData, int fieldOrder){
        try {
            //QueryList criteria = new QueryList(1, Integer.toString(ID), 0, false);
            //int recordID = Integer.parseInt(findFirst(criteria));

            String recordString = new String(recordData);
            String recordStringPlusSemiColon = new String(recordString + ";");

            //descobrindo o offset para fazer update apenas no campo indicado
            String record = new String(recordStore.getRecord(recordID));
            String currentSubstring= null;
            int begin = 0 ;
            int index = 0;
            int offset = 0;
            for (int i = 0; i < fieldOrder; i++){
                currentSubstring = record.substring(begin);
                index = currentSubstring.indexOf(";");
                offset = begin;
                begin += index + 1;
            }

            //mas se o length for maior do que o anterior, ele irá invadir o proximo campo
            //logo:
            //copiando tudo que havia depois do campo a ser alterado
            String nextSubstring = record.substring(begin, record.length());
            //copiando tudo que havia antes do campo a ser alterado
```

```
            String prevSubstring = record.substring(0, offset);
            //juntando as strings
            String recordFinalValue = new String(prevSubstring
                  + recordStringPlusSemiColon + nextSubstring);

            //atualizando o campo
            recordStore.setRecord(recordID, recordFinalValue.getBytes(), 0,
                        recordFinalValue.getBytes().length);
         } catch (RecordStoreNotOpenException ex) {
            ex.printStackTrace();
         } catch (InvalidRecordIDException ex) {
            ex.printStackTrace();
         } catch (RecordStoreException ex) {
            ex.printStackTrace();
         }
      }

      public int getCurrentRecordID(){
         return currentRecordID;
      }

      public Vector searchRecords(QueryList queryList){
         Vector records = new Vector();
         try {
            for (currentRecordID = 1; currentRecordID < recordStore.getNumRecords();
currentRecordID++) {
               String record = new String(recordStore.getRecord(currentRecordID));
               if (queryList.match(record)){
                  records.addElement(record);
               }
            }
         } catch (RecordStoreNotOpenException ex) {
            ex.printStackTrace();
```

```
    }finally{
        return records;
    }
}

//nao estou usando nem o findFirst nem o findNext mais. Tava com muito bug
public String findFirst(QueryList queryList){
    currentRecordID = 1;
    this.queryList = queryList;
    try {
        recordEnum = recordStore.enumerateRecords(null, null, true);
    } catch (RecordStoreNotOpenException ex) {
        ex.printStackTrace();
    }
    return findNextInternal();
}

public String findNext(){
    return findNextInternal();
}

//método interno
private String findNextInternal(){
    String record= null;
    do{
        try {
            currentRecordID = recordEnum.nextRecordId();
            byte[] data = recordStore.getRecord(currentRecordID);
            record = new String(data);

            if (queryList.match(record)){
                return record;
            }
```

```
        } catch (Exception ex) {
            ex.printStackTrace();
            return NOT_FOUND;
        }
    }while (recordEnum.hasNextElement());
    return NOT_FOUND;
}

public void recordAdded(RecordStore recordStore, int recordID) {
    try {
        String currentRecordStoreName =  recordStore.getName();
        System.out.println("Added RecordStore name: " + currentRecordStoreName);
        String str = new String(recordStore.getRecord(recordID));
        System.out.println("Record content: " + str);



    } catch (RecordStoreNotOpenException ex) {
            ex.printStackTrace();
        } catch (InvalidRecordIDException ex) {
            ex.printStackTrace();
        } catch (RecordStoreException ex) {
            ex.printStackTrace();
        }
}

public void recordChanged(RecordStore recordStore, int recordID) {
    try {
        System.out.println("Changed RecordStore name: " + recordStore.getName());
        String str = new String(recordStore.getRecord(recordID));
        System.out.println("Record content: " + str);
    } catch (RecordStoreNotOpenException ex) {
            ex.printStackTrace();
```

```
            } catch (InvalidRecordIDException ex) {
                ex.printStackTrace();
            } catch (RecordStoreException ex) {
                ex.printStackTrace();
            }
    }

    public void recordDeleted(RecordStore recordStore, int recordID){
        try {
            System.out.println("Deleted RecordStore name: " + recordStore.getName());
            String str = new String(recordStore.getRecord(recordID));
            System.out.println("Record content: " + str);
        } catch (RecordStoreNotOpenException ex) {
                ex.printStackTrace();
        } catch (InvalidRecordIDException ex) {
            ex.printStackTrace();
        } catch (RecordStoreException ex) {
            ex.printStackTrace();
        }
    }
}

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package util;

import util.QueryList;
import util.QueryRecordStore;
import java.util.Vector;
import javax.microedition.lcdui.Form;
```

```
import javax.microedition.lcdui.List;

/**
 *
 * @author yferreira
 * Não está sendo usada no momento
 */
public class RecordStoreDataAppender {

    public static final int STACK_ORDER = 0;
    public static final int QUEUE_ORDER = 1;

    public static void appendRecordStore(QueryRecordStore queryRecordStore, QueryList
queryList, int fieldOrder
            , List list, int printOrder) throws IllegalArgumentException{
        if (printOrder == STACK_ORDER){
            Vector records = new Vector();
            queryRecordStore.openRecordStore(true);
            String record = queryRecordStore.findFirst(queryList);
            while(record != null){
                records.addElement(QueryRecordStore.getField(record, fieldOrder));
                record = queryRecordStore.findNext();
            }
            queryRecordStore.closeRecordStore();
            for (int i = records.size() - 1; i >= 0; i--){
                record = (String) records.elementAt(i);
                list.append(record, null);
            }
        }else{
            if (printOrder == QUEUE_ORDER){

                queryRecordStore.openRecordStore(true);
                String record = queryRecordStore.findFirst(queryList);
```

```
            while(record != null){
               list.append(QueryRecordStore.getField(record, fieldOrder), null);
               record = queryRecordStore.findNext();
            }
            queryRecordStore.closeRecordStore();

         }else{
            throw new IllegalArgumentException("printOrder must have values 0 and 1 only");
         }
      }
   }

   public static void appendRecordStore(QueryRecordStore queryRecordStore, QueryList
queryList, int fieldOrder
         , Form form, int printOrder) throws IllegalArgumentException{
      if (printOrder == STACK_ORDER){
         Vector records = new Vector();
         queryRecordStore.openRecordStore(true);
         String record = queryRecordStore.findFirst(queryList);
         while(record != null){
            records.addElement(QueryRecordStore.getField(record, fieldOrder));
            record = queryRecordStore.findNext();
         }
         queryRecordStore.closeRecordStore();
         for (int i = records.size() - 1; i >= 0; i--){
            record = (String) records.elementAt(i);
            form.append(QueryRecordStore.getField(record, 2));
         }
      }else{
         if (printOrder == QUEUE_ORDER){

            queryRecordStore.openRecordStore(true);
            String record = queryRecordStore.findFirst(queryList);
```

```
            while(record != null){
                form.append(QueryRecordStore.getField(record, fieldOrder));
                record = queryRecordStore.findNext();
            }
            queryRecordStore.closeRecordStore();

        }else{
            throw new IllegalArgumentException("printOrder must have values 0 and 1 only");
        }
    }
}

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package util;

import javax.microedition.lcdui.Form;

/**
 *
 * @author yferreira
 * Tela genérica que pode ter seus dados atualizados
 */
public abstract class RefreshableForm extends Form {

    public RefreshableForm(String message){
        super(message);
    }
```

```
    public abstract void refresh();
}

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package util;

import javax.microedition.lcdui.List;

/**
 *
 * @author yferreira
 * Tela genérica que exibe lista de itens
 */
public abstract class RefreshableList extends List {

    public RefreshableList(String message){
        super(message,List.IMPLICIT);
    }

    public abstract void refresh();
    public abstract void add(String item);
    public abstract void delete(String item);
    public abstract void update(String newValue);
}
```

Apêndice E – Código Fonte do servidor HTTP

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package servlet;

import controleTransacao.FabricaDeAppService;
import util.AplicacaoException;
import com.sun.org.apache.xerces.internal.impl.dv.util.HexBin;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import models.Turma;
import service.TurmaService;

/**
 *
 * @author dcavalcante
 */
public class AbreContaServlet extends HttpServlet {

    public static final String SUCCESS = "SUCCESS";
    public static final String FAILURE = null;

    /**
```

```
    * Processes requests for both HTTP <code>GET</code> and <code>POST</code> methods.
    * @param request servlet request
    * @param response servlet response
    * @throws ServletException if a servlet-specific error occurs
    * @throws IOException if an I/O error occurs
    */
   protected void processRequest(HttpServletRequest request, HttpServletResponse response)
           throws ServletException, IOException {

       System.out.println("Entrou no processRequest");
       response.setContentType("text/html;charset=UTF-8");
       PrintWriter out = response.getWriter();

       String action = request.getParameter("action");

       TurmaService turmaService = null;

       if (action.compareTo("turmas") == 0){
           try {
               turmaService = (TurmaService) FabricaDeAppService.getAppService(TurmaService.class);
           } catch (Exception ex) {
               out.write(FAILURE);
               System.out.println("Error creating appServices" + ex.getMessage());
           }

           String listaTurmas = "";

           List<Turma> turmas = turmaService.getListTurmas();
           for (Turma turma : turmas) {
```

```
            String data = new String(turma.getCodigoDisciplina() + ";" +
turma.getNomeDisciplina() + ";" + turma.getCodigoTurma() + ";" +
turma.getNomeProfessor() + ";" + turma.getNumeroSala() + ";/");
            listaTurmas += data;
        }
        System.out.println("Turmas: " + listaTurmas);
        out.write(listaTurmas);
    }
}// <editor-fold defaultstate="collapsed" desc="Métodos HttpServlet. Clique no sinal de + à
esquerda para editar o código.">

/**
 * Handles the HTTP <code>GET</code> method.
 * @param request servlet request
 * @param response servlet response
 * @throws ServletException if a servlet-specific error occurs
 * @throws IOException if an I/O error occurs
 */
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException,
        IOException {
    processRequest(request, response);
}

/**
 * Handles the HTTP <code>POST</code> method.
 * @param request servlet request
 * @param response servlet response
 * @throws ServletException if a servlet-specific error occurs
 * @throws IOException if an I/O error occurs
 */
@Override
```

```
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException,
            IOException {
        processRequest(request, response);
    }

    /**
     * Returns a short description of the servlet.
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>
}

package util;


import controleTransacao.ExcecaoDeAplicacao;
import java.util.List;


@ExcecaoDeAplicacao
public class AplicacaoException extends Exception
{
        private final static long serialVersionUID = 1L;

        private List<String> mensagens;

        public AplicacaoException(String msg)
        {       super(msg);
        }
```

```
	public AplicacaoException(List<String> mensagens)
	{	this.mensagens = mensagens;
	}

	public List<String> getMensagens()
	{	return mensagens;
	}
}

package util;

public class InfraestruturaException extends RuntimeException
{
	private final static long serialVersionUID = 1L;

	public InfraestruturaException(Exception e)
	{	super(e);
	}
}

package util;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityTransaction;
import javax.persistence.Persistence;

public class JPAUtil {

   private static EntityManagerFactory emf = null;
   private static final ThreadLocal<EntityManager> threadEntityManager = new
ThreadLocal<EntityManager>();
```

```
    private static final ThreadLocal<EntityTransaction> threadTransaction = new
ThreadLocal<EntityTransaction>();

    static {
        try {
            emf = Persistence.createEntityManagerFactory("bdquadroturmas");
        } catch (Throwable e) {
            System.err.println(">>>>>>>>>>>>>>>> " + e.getMessage());
            System.out.println(e.getMessage());
        }
    }

    public static EntityManager getEntityManager() {
        EntityManager s = threadEntityManager.get();
        try {
            if (s == null) {
                s = emf.createEntityManager();
                threadEntityManager.set(s);
            }
        } catch (RuntimeException ex) {
            throw new InfraestruturaException(ex);
        }
        return s;
    }

    public static void closeEntityManager() {

        try {
            EntityManager s = threadEntityManager.get();
            threadEntityManager.set(null);
            if (s != null && s.isOpen()) {
                s.close();
            }
```

```
        EntityTransaction tx = threadTransaction.get();
        if (tx != null && tx.isActive()) {
            rollbackTransaction();
            throw new RuntimeException("EntityManager sendo fechado " +
                    "com transacao ativa.");
        }
    } catch (RuntimeException ex) {
        throw new InfraestruturaException(ex);
    }
}

public static void beginTransaction() {

    EntityTransaction tx = threadTransaction.get();
    try {
        if (tx == null) {
            tx = getEntityManager().getTransaction();
            tx.begin();
            threadTransaction.set(tx);
        } else {
        }
    } catch (RuntimeException ex) {
        throw new InfraestruturaException(ex);
    }
}

public static void commitTransaction() {
    EntityTransaction tx = threadTransaction.get();
    try {
        if (tx != null && tx.isActive()) {
            tx.commit();
        }
```

```
            threadTransaction.set(null);
        } catch (RuntimeException ex) {
            try {
                rollbackTransaction();
            } catch (RuntimeException e) {
            }

            throw new InfraestruturaException(ex);
        }
    }

    public static void rollbackTransaction() {

        EntityTransaction tx = threadTransaction.get();
        try {
            threadTransaction.set(null);
            if (tx != null && tx.isActive()) {
                tx.rollback();
            }
        } catch (RuntimeException ex) {
            throw new InfraestruturaException(ex);
        } finally {
            closeEntityManager();
        }
    }
}

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package util;
```

```
import java.util.Date;

/**
 *
 * @author yferreira
 */
public class MinhaDate extends Date {


    private MinhaDate(){

    }

    private static String getMonthDayYear(String date){
        String currentSubstring = date;
        String simpleDate = "";
        int begin = 0;
        int index = 0;

        for (int i = 0; i < 6; i++){
            currentSubstring = date.substring(begin);
            index = currentSubstring.indexOf(" ");
            if (i == 1 || i == 2){
                String str = currentSubstring.substring(0, index);
                simpleDate = simpleDate.concat(str + "/");
            }else{
                if (i == 5){
                  simpleDate = simpleDate.concat(currentSubstring);
                }
            }
            begin += index + 1;
            System.out.println(simpleDate);
```

```
        }
        return simpleDate;
    }

    public static String getCurrentDate(){
        Date date = new Date(System.currentTimeMillis());
        String simpleDate = getMonthDayYear(date.toString());
        return simpleDate;
    }
}
```

```
package util;

public class ObjetoNaoEncontradoException extends Exception
{
        private final static long serialVersionUID = 1;

        public ObjetoNaoEncontradoException()
        {
        }

        public ObjetoNaoEncontradoException(String msg)
        {       super(msg);
        }
}
```

```
package util;

import controleTransacao.ExcecaoDeAplicacao;


@ExcecaoDeAplicacao(rollback=true)
public class TransacaoException extends Exception
```

```
{
        private final static long serialVersionUID = 1L;

        public TransacaoException()
        {
        }

        public TransacaoException(String msg)
        {       super(msg);
        }

        public TransacaoException(Exception e)
        {       super(e);
        }
}

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package service;

import controleDao.FabricaDeDao;
import dao.TurmaDAO;
import java.util.List;
import models.Turma;
import util.AplicacaoException;
import util.ObjetoNaoEncontradoException;

/**
 *
 * @author Yuri
```

```
 */
public class TurmaService {

  private TurmaDAO turmaDAO;

   @SuppressWarnings("unchecked")

   public TurmaService() {
      try {
         turmaDAO = FabricaDeDao.getDao(TurmaDAO.class, Turma.class);
      } catch (Exception e) {
         e.printStackTrace();
         System.exit(1);
      }
   }

   public Turma getTurma(long id) throws AplicacaoException{
      try {
         return turmaDAO.getPorId(id);
      } catch (ObjetoNaoEncontradoException ex) {
         throw new AplicacaoException("Turma nao encontrada");
      }
   }

   public List<Turma> getListTurmas(){
      return turmaDAO.getListTurmas();
   }
}

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
```

```
package models;


/**
 *
 * @author dcavalcante
 */
public interface Model{

    long getId();
}

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package models;

import java.io.Serializable;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.Table;

/**
 *
 * @author Yuri
```

```
 */


@NamedQueries(
      {
            @NamedQuery
            (     name = "Turma.getTurma",
                  query = "select t from Turma t where t.id = ?"
            ),
            @NamedQuery
            (     name = "Turma.getListTurmas",
                  query = "select t from Turma t order by t.id"
            )
      })

@Entity
@Table(name="turma")
public class Turma implements Model, Serializable{

   private long id;
   private String codigoDisciplina; //codigo da disciplina
   private String nomeDisciplina; // nome da disciplina
   private String codigoTurma; //codigo da turma
   private String nomeProfessor; //nome do professor
   private String numeroSala; //sala onde as aulas serão realizadas

   public Turma(){

   }

   @Id
   @Column(name="id")
   @GeneratedValue(strategy=GenerationType.AUTO)
```

```
public long getId() {
   return id;
}

@SuppressWarnings("unused")
public void setId(long id) {
   this.id = id;
}

@Column(name="codigoDisciplina")
public String getCodigoDisciplina() {
   return codigoDisciplina;
}

public void setCodigoDisciplina(String codigoDisciplina) {
   this.codigoDisciplina = codigoDisciplina;
}

@Column(name="nomeDisciplina")
public String getNomeDisciplina() {
   return nomeDisciplina;
}

public void setNomeDisciplina(String nomeDisciplina) {
   this.nomeDisciplina = nomeDisciplina;
}

@Column(name="codigoTurma")
public String getCodigoTurma() {
   return codigoTurma;
}

public void setCodigoTurma(String codigoTurma) {
```

```
        this.codigoTurma = codigoTurma;
    }

    @Column(name="nomeProfessor")
    public String getNomeProfessor() {
        return nomeProfessor;
    }

    public void setNomeProfessor(String nomeProfessor) {
        this.nomeProfessor = nomeProfessor;
    }


    public void getNomeProfessor(String nomeProfessor) {
        this.nomeProfessor = nomeProfessor;
    }

    @Column(name="numeroSala")
    public String getNumeroSala() {
        return numeroSala;
    }

    public void setNumeroSala(String numeroSala) {
        this.numeroSala = numeroSala;
    }
}

package dao;

import util.ObjetoNaoEncontradoException;
import java.io.Serializable;
import java.util.List;
```

```
/**
 * A interface GenericDao b�sica com os m�todos CRUD. Os m�todos
 * de busca s�o adicionados por heran�a de interface.
 *
 * Interfaces estendidas podem declarar m�todos que come�am  por
 * busca...  recuperaCnjuntoDe...  ou recupera...  Estes m�todos
 * ir�o  executar buscas pr�-configuradas que s�o localizadas em
 * fun��o do restante do nome dos m�todos.
 *
 */
public interface DaoGenerico<T, PK extends Serializable>
{
   T getPorId(PK id) throws ObjetoNaoEncontradoException;

   T getPorIdComLock(PK id) throws ObjetoNaoEncontradoException;

   T recuperaUm(PK id) throws ObjetoNaoEncontradoException;

   List<T> recuperaLista();
}

package dao;

import util.ObjetoNaoEncontradoException;
import java.lang.reflect.Method;
import java.util.List;
import java.util.Set;


public interface ExecutorDeBuscas<T>
{
```

```
    public T busca(Method method, Object[] queryArgs)
                    throws ObjetoNaoEncontradoException;

    public List<T> buscaLista(Method method, Object[] queryArgs);

    public Set<T> buscaConjunto(Method method, Object[] queryArgs);

    public T buscaUltimoOuPrimeiro (Method method, Object[] queryArgs)
                    throws ObjetoNaoEncontradoException;
}

package dao;

import util.InfraestruturaException;
import util.JPAUtil;
import util.ObjetoNaoEncontradoException;
import java.io.Serializable;
import java.lang.reflect.Method;
import java.util.LinkedHashSet;
import java.util.List;
import java.util.Set;

import javax.persistence.EntityManager;
import javax.persistence.LockModeType;
import javax.persistence.NoResultException;
import javax.persistence.Query;

import org.hibernate.HibernateException;



/**
 * A implementacao de um DAO generico para a JPA
```

```
 * Uma implementacao "typesafe" dos metodos CRUD e dos metodos de busca.
 */
public abstract class JPADaoGenerico<T, PK extends Serializable>
        implements DaoGenerico<T, PK>, ExecutorDeBuscas<T>
{
   private Class<T> tipo;

   public JPADaoGenerico(Class<T> tipo)
   {
        this.tipo = tipo;
   }

   public final T getPorId(PK id) throws ObjetoNaoEncontradoException
   {
        EntityManager em = JPAUtil.getEntityManager();
      T t = null;
      try
      {  t = (T)em.find(tipo, id);

         if (t == null)
         {         throw new ObjetoNaoEncontradoException();
         }
      }
      catch(RuntimeException e)
      {   throw new InfraestruturaException(e);
      }
      return t;
   }

   public final T getPorIdComLock(PK id) throws ObjetoNaoEncontradoException
   {
        EntityManager em = JPAUtil.getEntityManager();
      T t = null;
```

```
   try
   {
      t = (T)em.find(tipo, id);

      if (t != null)
      {   em.lock(t, LockModeType.READ);
                              em.refresh(t);
      }
      else
      {        throw new ObjetoNaoEncontradoException();
      }
   }
   catch(RuntimeException e)
   {   throw new InfraestruturaException(e);
   }

   return t;
}

@SuppressWarnings("unchecked")
      public final T busca(Method metodo, Object[] argumentos)
      throws ObjetoNaoEncontradoException
{
      EntityManager em = JPAUtil.getEntityManager();
   T t = null;
   try
   {
      String nomeDaBusca = getNomeDaBuscaPeloMetodo(metodo);
      Query namedQuery = em.createNamedQuery(nomeDaBusca);

      if (argumentos != null)
      {        for (int i = 0; i < argumentos.length; i++)
         {     Object arg = argumentos[i];
```

```
                namedQuery.setParameter(i+1, arg);  // Par?metros de buscas s?o 1-based.
            }
        }
        t = (T)namedQuery.getSingleResult();

        return t;
    }
    catch(NoResultException e)
    {   throw new ObjetoNaoEncontradoException();
    }
    catch(RuntimeException e)
    {   throw new InfraestruturaException(e);
    }
}

@SuppressWarnings("unchecked")
public final T buscaUltimoOuPrimeiro(Method metodo,
                                     Object[] argumentos)
        throws ObjetoNaoEncontradoException
{
        EntityManager em = JPAUtil.getEntityManager();
    T t = null;
    try
    {
        List lista;
        String nomeDaBusca = getNomeDaBuscaPeloMetodo(metodo);
        Query namedQuery = em.createNamedQuery(nomeDaBusca);

        if (argumentos != null)
        {
            for (int i = 0; i < argumentos.length; i++)
            {
                Object arg = argumentos[i];
```

```
            namedQuery.setParameter(i+1, arg);
          }
       }
       lista = namedQuery.getResultList();

       t = (lista.size () == 0) ? null : (T)lista.get(0) ;

       if (t == null)
       {   throw new ObjetoNaoEncontradoException();
       }

       return t;
    }
    catch(HibernateException e)
    {   throw new InfraestruturaException(e);
    }
}

@SuppressWarnings("unchecked")
       public final List<T> buscaLista(Method metodo,
                                 Object[] argumentos)
{
       EntityManager em = JPAUtil.getEntityManager();

    try
    {
       String nomeDaBusca = getNomeDaBuscaPeloMetodo(metodo);
       Query namedQuery = em.createNamedQuery(nomeDaBusca);

       if (argumentos != null)
       {
          for (int i = 0; i < argumentos.length; i++)
          {
```

```
            Object arg = argumentos[i];
            namedQuery.setParameter(i+1, arg); // Par�metros de buscas s�o 1-based.
         }
      }
      return (List<T>)namedQuery.getResultList();
   }
   catch(RuntimeException e)
   {   throw new InfraestruturaException(e);
   }
}

@SuppressWarnings("unchecked")
public final Set<T> buscaConjunto(Method metodo,
                                  Object[] argumentos)
{
      EntityManager em = JPAUtil.getEntityManager();

   try
   {
      String nomeDaBusca = getNomeDaBuscaPeloMetodo(metodo);
      Query namedQuery = em.createNamedQuery(nomeDaBusca);

      if (argumentos != null)
      {         for (int i = 0; i < argumentos.length; i++)
         {      Object arg = argumentos[i];
            namedQuery.setParameter(i+1, arg); // Par�metros de buscas s�o 1-based.
         }
      }

      List<T> lista = namedQuery.getResultList();

      return new LinkedHashSet(lista);
   }
```

```
        catch(RuntimeException e)
        {   throw new InfraestruturaException(e);
        }
    }

    public final String getNomeDaBuscaPeloMetodo(Method metodo)
    {     return tipo.getSimpleName() + "." + metodo.getName();
    }

}

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package dao;

import util.ObjetoNaoEncontradoException;
import java.util.List;
import models.Turma;

/**
 *
 * @author yferreira
 */
public interface TurmaDAO extends DaoGenerico<Turma, Long> {

    Turma getTurma(long id) throws ObjetoNaoEncontradoException;
    List<Turma> getListTurmas();
}

/*
```

```
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package dao;

import models.Turma;

/**
 *
 * @author Yuri
 */
public abstract class TurmaDAOImpl extends JPADaoGenerico<Turma, Long> implements TurmaDAO{

	public TurmaDAOImpl()
	{	super(Turma.class);
	}

}

package controleTransacao;

import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;

@Target (ElementType.TYPE)
@Retention (RetentionPolicy.RUNTIME)
public @interface ExcecaoDeAplicacao
{	boolean rollback() default true;
}
```

```
package controleTransacao;

import java.util.HashMap;

import net.sf.cglib.proxy.Enhancer;

public class FabricaDeAppService
{
	@SuppressWarnings("unchecked")
	private static HashMap<Class, Object> map = new HashMap<Class, Object>();

	@SuppressWarnings("unchecked")
	public static Object getAppService(Class classeDoBean)
		throws Exception
   {
		// classeDoBean = ContaAppService.class ou
		//		LanceAppService.class, por exemplo.
		Object appService = map.get(classeDoBean);

		if(appService == null)
		{	InterceptadorDeAppService interceptadorDeAppService = new
InterceptadorDeAppService();
			appService = Enhancer.create (classeDoBean,
interceptadorDeAppService);
			map.put(classeDoBean, appService);
		}

		return appService;

   }
}
```

```
package controleTransacao;

import util.InfraestruturaException;
import util.JPAUtil;
import java.lang.reflect.Method;

import net.sf.cglib.proxy.MethodInterceptor;
import net.sf.cglib.proxy.MethodProxy;

public class InterceptadorDeAppService implements MethodInterceptor {

   public Object intercept(Object objeto,
         Method metodo,
         Object[] args,
         MethodProxy metodoProxy)
         throws Throwable {
      try {
         Object resultado = null;

         if (isTransacional(metodo)) {
            JPAUtil.beginTransaction();

            resultado = metodoProxy.invokeSuper(objeto, args);

            JPAUtil.commitTransaction();
         } else {
            resultado = metodoProxy.invokeSuper(objeto, args);
         }
         return resultado;
      } catch (RuntimeException e) {
         try {
            JPAUtil.rollbackTransaction();
         } catch (InfraestruturaException ie) {
```

```
            }

            throw e;
        } catch (Exception e) {
            if (isTransacional(metodo)) {
                if (efetuarRollback(e)) {
                    JPAUtil.rollbackTransaction();
                } else {
                    JPAUtil.commitTransaction();
                }
            }

            throw e;
        } finally {
            JPAUtil.closeEntityManager();
        }
    }

    public boolean isTransacional(Method metodo)
            throws Exception {
        return metodo.isAnnotationPresent(Transacional.class);
    }

    public boolean efetuarRollback(Exception e) {
        if (e instanceof RuntimeException) {
            return true;
        } else {
            if (e.getClass().isAnnotationPresent(ExcecaoDeAplicacao.class)) {
                return e.getClass().getAnnotation(ExcecaoDeAplicacao.class).rollback();
            } else {
                return true;
            }
        }
```

```
    }
}

package controleTransacao;

import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;

@Target (ElementType.METHOD)
@Retention (RetentionPolicy.RUNTIME)
public @interface Transacional
{}

package controleDao;

import dao.JPADaoGenerico;
import net.sf.cglib.proxy.Enhancer;

public class FabricaDeDao
{
   //produtoDAO = FabricaDeDao.<ProdutoDAO>getDao(ProdutoDAO.class, Produto.class);

        @SuppressWarnings("unchecked")
   public static <T> T getDao(Class<T> interfaceDoDao, Class tipoDoDao)
      throws Exception
   {
      Enhancer enhancer = new Enhancer();

      enhancer.setInterfaces(new Class[] {interfaceDoDao});// Interface implementada
                                          // ProdutoDAO, por exemplo.
```

```
        enhancer.setSuperclass(JPADaoGenerico.class);    // Superclasse do DAO
        enhancer.setCallback(new InterceptadorDeDAO());  // Interceptador do DAO

        return (T) enhancer.create(new Class[] { Class.class }, new Object[] { tipoDoDao });
    }
}

package controleDao;

import dao.ExecutorDeBuscas;
import java.lang.reflect.Method;

import net.sf.cglib.proxy.MethodInterceptor;
import net.sf.cglib.proxy.MethodProxy;

public class InterceptadorDeDAO implements MethodInterceptor
{
        /* Parametros:
         *
         * objeto - "this", o objeto "enhanced", isto é, o proxy.
         *
         * metodo - o método interceptado.
         *
         * args - um array de args; tipos primitivos são empacotados.
         *
         * metodoProxy - utilizado para executar um método super.
         *
         * MethodProxy  -  Classes  geradas pela  classe Enhancer passam
         * este objeto para o objeto MethodInterceptor registrado quando
         * um método  interceptado é  executado.  Ele pode ser utilizado
         * para  invocar o  método  original,  ou  chamar o mesmo método
         * sobre um objeto diferente do mesmo tipo.
```

```
     */

    public Object intercept (Object objeto,
                             Method metodo,
                             Object[] args,
                        MethodProxy metodoDoProxy)
    throws Throwable
{
            // O s□mbolo ? representa um tipo desconhecido.
  ExecutorDeBuscas<?> daoGenerico = (ExecutorDeBuscas<?>)objeto;

  String nomeDoMetodo = metodo.getName();

  if (nomeDoMetodo.startsWith("getList"))
  {  // O m□todo recuperaLista() retorna um List
    return daoGenerico.buscaLista(metodo, args);
  }
  else if (nomeDoMetodo.startsWith("getSet"))
  {  // O m□todo recuperaConjunto() retorna um Set
    return daoGenerico.buscaConjunto(metodo, args);
  }
  else if (nomeDoMetodo.startsWith("getLast") ||
            nomeDoMetodo.startsWith("getFirst"))
  {  // O m□todo recuperaConjunto() retorna um Set
    return daoGenerico.buscaUltimoOuPrimeiro(metodo, args);
  }
  else if (nomeDoMetodo.startsWith("get"))
  {  // O m□todo recupera() retorna uma Entidade
    return daoGenerico.busca(metodo, args);
  }
  else
  {         return metodoDoProxy.invokeSuper(objeto, args);
```

```
        }
    }
}
```

Apêndice F – Código Fonte do Servidor Bluetooth

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package servidorobexbluetooth;

import java.io.IOException;
import java.util.Vector;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.bluetooth.RemoteDevice;

/**
 *
 * @author yferreira
 */
public class BTServerStarter {

    private static Vector oldDevs=new Vector();
    private static Vector failedDevs=new Vector();

    public static void main(String[] args){
        try {
            sendMessage();
        } catch (IOException ex) {
            Logger.getLogger(BTServerStarter.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    public static void sendMessage() throws IOException
```

```
    {

        BTServer server = new  BTServer();
        server.start();
        while (true){
            server.inquiryStart();
        }
    }

    public static Vector getOldDevs(){
        return oldDevs;
    }

    public static void setOldDevs(RemoteDevice dev){
        oldDevs.addElement(dev);
    }

    public static Vector getFailedDevs(){
        return failedDevs;
    }

    public static void setFailedDevs(RemoteDevice dev){
        failedDevs.addElement(dev);
    }

}

package servidorobexbluetooth;


import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
```

```
import java.io.InputStream;
import java.io.OutputStream;
import java.util.Vector;

import java.util.logging.Level;
import java.util.logging.Logger;
import javax.bluetooth.BluetoothStateException;
import javax.bluetooth.DataElement;
import javax.bluetooth.DeviceClass;
import javax.bluetooth.DiscoveryAgent;
import javax.bluetooth.DiscoveryListener;
import javax.bluetooth.LocalDevice;
import javax.bluetooth.RemoteDevice;
import javax.bluetooth.ServiceRecord;
import javax.bluetooth.UUID;
import javax.microedition.io.Connection;
import javax.microedition.io.Connector;
import javax.obex.ClientSession;
import javax.obex.HeaderSet;
import javax.obex.Operation;


/**
*
* Class that discovers all bluetooth devices in the neighbourhood,
*
* Connects to the chosen device and checks for the presence of OBEX push service in it.
* and displays their name and bluetooth address.
*
*
*/
public class BTServer implements DiscoveryListener{
```

```
    private static Object lock=new Object();
    private static Vector devs=new Vector();
    private static Vector servs=new Vector();
    private static String connectionURL=null;
    private static LocalDevice localDevice=null;
    private String serverURL= null;
    private final String filePath = "C:/Users/Yuri/Documents/NetBeansProjects/QuadroDeTurmasMobile/dist/QuadroDeTurmasMobile.jar";
    /**
    * Entry point.
    */
    public BTServer(){
    }

    public void start()
    {
        devs.removeAllElements();

        localDevice=null;
        try {
            localDevice = LocalDevice.getLocalDevice();
        } catch (BluetoothStateException ex) {
            Logger.getLogger(BTServer.class.getName()).log(Level.SEVERE, null, ex);
        }
        System.out.println("Address: "+localDevice.getBluetoothAddress());
        System.out.println("Name: "+localDevice.getFriendlyName());
    }

    public void inquiryStart(){

        DiscoveryAgent agent = localDevice.getDiscoveryAgent();
```

```
        System.out.println("Starting device inquiry...");
        try {
            //mudei aqui, era bluetoothServiceDiscovery e nao this
            agent.startInquiry(DiscoveryAgent.GIAC, this);
        } catch (BluetoothStateException ex) {
            Logger.getLogger(BTServer.class.getName()).log(Level.SEVERE, null, ex);
        }
        try {
            synchronized(lock)
            {
            lock.wait();
            }
        }
        catch (InterruptedException e) {
            System.out.println("************************Erro: " + e.getMessage());
        }

        int deviceCount=devs.size();
        if(deviceCount <= 0){
            System.out.println("No Devices Found .");
        }
        else{
            for(int i = 0; i < devs.size(); i++) {
                RemoteDevice remoteDevice=(RemoteDevice)devs.elementAt(i);
                try {
                    System.out.println((i + 1) + ". " + remoteDevice.getFriendlyName(false) + ": " +
remoteDevice.getBluetoothAddress());
                } catch (IOException ex) {
                    Logger.getLogger(BTServer.class.getName()).log(Level.SEVERE, null, ex);
                }
            }
        }
```

```
for (int j = 0; j < devs.size(); j++){

    //check for obex service
    //RemoteDevice remoteDevice=(RemoteDevice)devs.elementAt(j);
    RemoteDevice remoteDevice=(RemoteDevice)devs.elementAt(j);

    UUID[] uuidSet = {new UUID("1105", true)};

    int[] attrSet = {0x0100, 0x0003, 0x0004};
    System.out.println("\nSearching for service...");
    connectionURL = null;
    int transID=0;
    try {
       //mudei aqui, era bluetoothServiceDiscovery e nao this
       transID = agent.searchServices(attrSet, uuidSet, remoteDevice, this);
    } catch (BluetoothStateException ex) {
       Logger.getLogger(BTServer.class.getName()).log(Level.SEVERE, null, ex);
    }
    System.out.println("Service Search in Progress ("+transID+")");
    try {
       synchronized(lock)
       {
       lock.wait();
       }
    }
    catch (InterruptedException e)
    {
       System.out.println("**************************Erro: " + e.getMessage());
    }

    //System.out.println("Service Discovery Completed. ");
    System.out.println("Opening a connection with the server.... ");
    // connection creation & sending file
```

```
try
{
   Connection connection = Connector.open(connectionURL);

   System.out.println("Connection obtained");

   ClientSession cs = (ClientSession)connection;
   HeaderSet hs = cs.createHeaderSet();

   cs.connect(hs);
   System.out.println("OBEX session created");

   //System.out.println("Response code of the server after connect..."
+hs.getResponseCode());

   File file = new File(filePath);
   InputStream is = new FileInputStream(file);
   byte filebytes[] = new byte[is.available()];
   is.read(filebytes);
   is.close();

   hs = cs.createHeaderSet();
   hs.setHeader(HeaderSet.NAME, file.getName());
   hs.setHeader(HeaderSet.TYPE, "jar");
   hs.setHeader(HeaderSet.LENGTH, new Long(filebytes.length));

   Operation putOperation = cs.put(hs);
   System.out.println("Pushing file: " + file.getName());
   System.out.println("Total file size: " + filebytes.length + " bytes");

   OutputStream outputStream = putOperation.openOutputStream();
   outputStream.write(filebytes);
   System.out.println("File push complete");
```

```
            Vector failedDevs = BTServerStarter.getFailedDevs();
            if (failedDevs.contains(remoteDevice)){
               failedDevs.remove(remoteDevice);
            }

            BTServerStarter.setOldDevs(remoteDevice);

            outputStream.close();
            putOperation.close();

            cs.disconnect(null);

            connection.close();
         }
         catch(Exception e)
         {
            System.out.println(e.getMessage());
            System.out.println("connection:"+e);
            if (connectionURL != null){
               BTServerStarter.setOldDevs(remoteDevice);
            }else{
               BTServerStarter.setFailedDevs(remoteDevice);
            }
         }
      }
      devs.removeAllElements();
   }

   /**
   * Called when a bluetooth device is discovered.
   * Used for device search.
   */
```

```
public void deviceDiscovered(RemoteDevice btDevice, DeviceClass cod) {
   //add the device to the vector
   String address = btDevice.getBluetoothAddress();
   long decimal = Long.parseLong(address, 16);
   address = Long.toString(decimal);

   Vector oldDevs = BTServerStarter.getOldDevs();
   Vector failedDevs = BTServerStarter.getFailedDevs();
   if ((!devs.contains(btDevice) && !oldDevs.contains(btDevice)) ||
         (!devs.contains(btDevice) && failedDevs.contains(btDevice))){
      devs.addElement(btDevice);
   }
   /*Vector oldDevs = BTServerStarter.getOldDevs();
   Vector failedDevs = BTServerStarter.getFailedDevs();
   if((!devs.contains(btDevice) && !oldDevs.contains(btDevice)) ||
         (!devs.contains(btDevice) && failedDevs.contains(btDevice))){
      devs.addElement(btDevice);
   }*/
}

/**
* Called when the device search is over.
*/
public void inquiryCompleted(int discType) {

   switch (discType)
   {
      case DiscoveryListener.INQUIRY_COMPLETED :
         System.out.println("INQUIRY_COMPLETED");
         break;
      case DiscoveryListener.INQUIRY_TERMINATED :
         System.out.println("INQUIRY_TERMINATED");
         break;
```

```
        case DiscoveryListener.INQUIRY_ERROR :
            System.out.println("INQUIRY_ERROR");
            break;
        default :
            System.out.println("Unknown Response Code");
            break;
    }
    synchronized(lock){
        lock.notify();
    }
}//end method

/**
* Called when a bluetooth service is discovered.
* Used for service search.
*/
public void servicesDiscovered(int transID, ServiceRecord[] servRecord) {

    /*if(servRecord!=null && servRecord.length>0)
    {
    connectionURL=servRecord[0].getConnectionURL(0,false);
    System.out.println(connectionURL+"--"+servRecord);
    }*/
    for(int i = 0; i < servRecord.length; i++)
    {
        servs.addElement(servRecord[i].getConnectionURL(0,false));
        DataElement serviceNameElement = servRecord[i].getAttributeValue(0x0100);
        String _serviceName = (String)serviceNameElement.getValue();
        String serviceName = _serviceName.trim();
        System.out.println("Service Name: " + _serviceName);

        if(serviceName.equals("OBEX Object Push"))
        {
```

```
            System.out.println("[client:] A matching service has been found\n");

            try
            {
               connectionURL = servRecord[i].getConnectionURL(0,false);
               System.out.println(connectionURL+"\n");
            } catch (Exception e){
               System.out.println("[client:] oops");
            }
         }
      }
   }
   /**
   * Called when the service search is over.
   */
   public void serviceSearchCompleted(int transID, int respCode) {
      /*synchronized(lock)
      {
      lock.notify();
      }*/
      String searchStatus = null;

      if (respCode ==
DiscoveryListener.SERVICE_SEARCH_DEVICE_NOT_REACHABLE) {
         searchStatus = "SERVICE_SEARCH_DEVICE_NOT_REACHABLE\n";
      }
      else if (respCode == DiscoveryListener.SERVICE_SEARCH_NO_RECORDS) {
            searchStatus = "SERVICE_SEARCH_NO_RECORDS\n";
          }
         else if (respCode == DiscoveryListener.SERVICE_SEARCH_COMPLETED) {
               searchStatus = "SERVICE_SEARCH_COMPLETED\n";
            }
            else if (respCode == DiscoveryListener.SERVICE_SEARCH_TERMINATED) {
```

```
                searchStatus = "SERVICE_SEARCH_TERMINATED\n";
            }
            else if (respCode == DiscoveryListener.SERVICE_SEARCH_ERROR) {
                 searchStatus = "SERVICE_SEARCH_ERROR\n";
               }

    System.out.println("[client:] " + searchStatus);

    synchronized(lock)
    {
    lock.notify();
    }
  }
}//end class
```