UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO – BACHARELADO

PROTÓTIPO DE FRONT END DE CONTROLE DE ACESSO USANDO J2ME

ARNO JOSÉ SCHMITT JUNIOR

BLUMENAU
2004

2004/2-07

ARNO JOSÉ SCHMITT JUNIOR

PROTÓTIPO DE FRONT END DE CONTROLE DE ACESSO USANDO J2ME

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciência da Computação — Bacharelado.

Prof. Marcel Hugo - Orientador

BLUMENAU
2004

2004/2-07

PROTÓTIPO DE FRONT END DE CONTROLE DE ACESSO USANDO J2ME

Por

ARNO JOSÉ SCHMITT JUNIOR

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Prof. Marcel Hugo – Orientador, FURB
Membro:	Prof. Paulo de Tarso Mendes Luna - FURB
Membro:	Prof. Francisco Adell Péricas, FURB

Blumenau, 22 de novembro de 2004

Dedico este trabalho aos meus pais, a minha namorada, aos meus amigos, aos mestres da universidade, principalmente meu orientador e a todos aqueles que acreditaram em meu potencial.

Uma caminhada que chegou ao fim e rendeu bons frutos. Os conhecimentos serão aprimorados, os momentos lembrados, as amizades cultivadas e a vida continuará sendo explorada, aproveitando ao máximo cada segundo.

Arno José Schmitt Junior

AGRADECIMENTOS

Aos meus pais, Arno e Célia, por me incentivarem e por me amarem. Eles, que se esforçaram para oferecer o melhor lar e a melhor educação a seus filhos, são exemplos para mim.

Aos meus irmãos, Elaine, Saskia e Bruno, por fazerem parte da família que eu amo tanto e por estarem sempre dispostos a ajudar quando preciso. A Nadir, que também me educou e sempre é atenciosa por mim.

A minha namorada, Andressa, por ter sido compreensiva durante este trabalho, por me incentivar e por ser uma ótima companheira e receber o meu amor.

Aos meus amigos, pela felicidade que me passam e por estarem sempre me apoiando. Aos Amigos do Barney, pelas festas e comemorações que passamos juntos.

Ao meu orientador, Marcel, por me receber como seu orientado e pela ajuda e paciência dedicada neste trabalho.

RESUMO

Este trabalho apresenta a especificação e desenvolvimento de um protótipo de *front end* de controle de acesso para execução em dispositivos móveis. A portabilidade em dispositivos móveis é proposta pela utilização da edição Java 2 Micro Edition com a utilização do perfil MIDP e configuração CLDC. A linguagem de programação orientada a objetos Java, utilizando padrões de projeto, é utilizada.

Palavras chaves: Controle de Acesso; Dispositivos Móveis; J2ME.

ABSTRACT

This work presents the specification and implementation of a control access archetype front end for execution at mobile devices. The portability at mobile devices is proposed by the use of Java 2 Micro Edition with use of MIDP profile and CLDC configuration. The object-oriented language Java, using design patterns, is used.

Key-Words: Access Control; Mobile Devices; J2ME.

LISTA DE ILUSTRAÇÕES

Figura 1 - Caminho percorrido pelo código Java 17
Figura 2 - Universo J2ME 19
Figura 3 - 23 padrões de projetos descritos por Gamma 23
Figura 4 - Relação entre casos de uso e atores do sistema 28
Figura 5 - Diagrama de atividades do caso de uso “importar informações” 31
Figura 6 - Diagrama de atividades do caso de uso “verificar acesso” 31
Figura 7 - Diagrama de atividades do caso de uso “registrar acesso” 32
Figura 8 - Diagrama de atividades do caso de uso “configurar armazenamento de informações” 32
Figura 9 - Diagrama de atividades do caso de uso “configurar apresentação dos resultados” 33
Figura 10 - Diagrama de atividades do caso de uso “configurar fonte de informações” 33
Figura 11 - Diagrama de atividades do caso de uso “exportar acessos registrados” 33
Figura 12 - Diagrama de classes da numa visão de negócio do sistema 36
Figura 13 - Diagrama de classes do assunto “informações do acesso” 37
Figura 14 - Diagrama de classes do assunto “registro do acesso” 38
Figura 15 - Classes utilitárias do sistema 38
Figura 16 - Diagrama de seqüência do caso de uso “importar informações” 40
Figura 17 - Diagrama de seqüência do caso de uso “verificar acesso” 40
Figura 18 - Diagrama de seqüência do caso de uso “registrar acesso” 41
Figura 19 - Diagrama de seqüência do caso de uso ”exportar acessos registrados” 41
Figura 20 - Rastreabilidade dos requisitos e casos de uso 42
Figura 21 - Uso do padrão *Strategy* na aplicação *web* 45
Figura 22 - Tela inicial 48
Figura 23 - Menu geral 49
Figura 24 - Menu configuração 50
Figura 25 - Tela de configuração geral 51
Figura 26 - Tela configuração aplicação web 52
Figura 27 - Tela configuração apresentação das informações 53
Figura 28 - Tela de informação da importação 54
Figura 29 - Tela de confirmação do uso da rede do dispositivo 54
Figura 30 - Tela de controle do acesso 55
Figura 31 - Tela de informação importação local 56
Figura 32 - Tela de exportação dos acessos registrados localmente 57
Figura 33 - Tela validação do acesso sem histórico válido 58
Figura 34 - Tela validação do acesso sem permissão de acesso ao local 59
Figura 35 - Tela validação inválido pela faixa horária 59
Figura 36 - Tela validação identificador inválido ou sem permissão ao local 60
Figura 37 - Validação com todas as informações apresentadas 61

SUMÁRIO

1 INTRODUÇÃO 11
1.1 OBJETIVOS DO TRABALHO 12
1.2 ESTRUTURA DO TRABALHO 12
2 FUNDAMENTAÇÃO TEÓRICA 14
2.1 CONTROLE DE ACESSO 14
2.2 DISPOSITIVOS MÓVEIS 16
2.3 TECNOLOGIA JAVA 17
2.4 J2ME 18
2.4.1 Configuração CLDC 20
2.4.2 Pefil MIDP 21
2.5 PADRÕES DE PROJETO 22
2.5.1 Padrão de projeto *Factory Method* 23
2.5.2 Padrão de projeto *Singleton* 23
2.5.3 Padrão de projeto *Strategy* 24
2.6 SERVLETS JAVA 24
3 DESENVOLVIMENTO DO TRABALHO 25
3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO 25
3.2 REQUISITOS NÃO FUNCIONAIS 27
3.3 ESPECIFICAÇÃO 27
3.3.1 CASOS DE USO DO PROBLEMA 28
3.3.2 DIAGRAMA DE ATIVIDADES DOS CASOS DE USO 30
3.3.3 DIAGRAMA DE CLASSES 34
3.3.4 REALIZAÇÃO DE CASOS DE USO 39
3.3.5 FERRAMENTA UTILIZADA NA ESPECIFICAÇÃO DO TRABALHO 41
3.4 IMPLEMENTAÇÃO 43
3.4.1 TÉCNICAS E FERRAMENTAS UTILIZADAS 43
3.4.2 OPERACIONALIDADE DA IMPLEMENTAÇÃO 47
3.5 RESULTADOS E DISCUSSÃO 61
3.5.1 PROBLEMAS ENCONTRADOS DURANTE O DESENVOLVIMENTO 62
4 CONCLUSÕES 63
4.1 EXTENSÕES 64
REFERÊNCIAS BIBLIOGRÁFICAS 65

APÊNDICE A – Implementação do padrão *Strategy* no trabalho..... 66
APÊNDICE B – Implementação do padrão *Factory Method* no trabalho 67
APÊNDICE C – Implementação do padrão *Singleton* no trabalho 68

1 INTRODUÇÃO

A necessidade de segurança patrimonial é eminente nos dias atuais. Em meio às concorrências empresarias, onde as empresas protegem suas informações contra espionagem, e à desigualdade social, que acarreta violência e crimes de furto, surge a necessidade das corporações investirem em soluções de segurança do patrimônio. Entidades de ensino que desejam fortalecer o controle de presença de pessoas matriculadas em seus cursos também necessitam de uma solução, preferencialmente móvel, que possa ser aplicada em diferentes ambientes onde são ministrados os seus cursos. As tecnologias de controle diferem de acordo com a complexidade de atuação. Dentre as diversas tecnologias de controle, destaca-se a utilização de crachás para checagem do acesso.

A maioria dos controles de acesso físico atuais que se tem conhecimento são realizados por sistemas embarcados. Porém, estas soluções não oferecem o grau de flexibilidade desejado devido às restrições do poder de funcionalidades dos mesmos. Estas restrições estão relacionadas à utilização de equipamentos de hardware e software com mínimas capacidades de flexibilidade de uso. Os recursos limitam-se em coletar o identificador do crachá e a partir desta informação é verificado em processos internos, consultando banco de dados ou memória do equipamento, se aquele crachá tem permissão de acesso liberada ou bloqueada para aquele instante. A visualização de retorno desta verificação é bastante pobre de recursos gráficos e informações, pois os equipamentos que se tem conhecimento utilizam telas de dezesseis colunas por duas linhas para exibição.

Os itens negativos das soluções embarcadas podem ser minimizados com a utilização de um dispositivo que ofereça um melhor processamento, maior capacidade de armazenamento, suporte a diversificados tipos de comunicação e recursos gráficos avançados. Um *Personal Digital Assistant* (PDA) atende a estas necessidades e ainda oferece suporte a uma plataforma que provê recursos de uma linguagem orientada a objetos. Esta plataforma é a *Java 2 Plataform, Micro Edition* (J2ME), que traz uma linguagem de programação robusta e versátil para dispositivos móveis. Este trabalho visa utilizar os recursos desta plataforma para oferecer um protótipo de controle de acesso que execute em um ambiente de computação móvel e que seja flexível de acordo com situações de uso variadas. A questão de ser um dispositivo móvel possibilita a utilização desta solução em diferentes ambientes onde haja a necessidade de controlar o acesso em locais sem um controle fixo: um evento empresarial ou de ensino realizado em um auditório externo à corporação por exemplo. A flexibilidade da

aplicação poderá ser confirmada pelas formas de gravação da informação, seja em um banco de dados com a presença de recursos de rede ou em memória no dispositivo quando da ausência dos recursos de rede. A visualização da informação processada será disposta de acordo com configurações no aplicativo. As informações pessoais, tais como nome e cargo, poderão ser divulgadas no momento do acesso, assim como a foto da pessoa. Uma mensagem comunicativa registrada para a pessoa também poderá ser visualizada no momento do acesso. Essas funcionalidades e visualizações dependem das situações em que o aplicativo será exposto.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é o desenvolvimento de um protótipo de *front end* para controle de acesso, utilizando a tecnologia J2ME, para funcionamento em dispositivos móveis.

Os objetivos específicos do trabalho são:

a) oferecer um protótipo com flexibilidade de aplicação, em relação ao armazenamento de informações e interface ao usuário, adequado a diferentes situações de uso conforme configurações definidas;
b) controlar o acesso a partir de um dispositivo independente de sistema operacional desde que tenha suporte à plataforma J2ME;
c) automatizar o controle de segurança patrimonial.

1.2 ESTRUTURA DO TRABALHO

O presente trabalho está subdividido em quatro capítulos que são referidos a seguir.

O primeiro capítulo contextualiza e justifica o desenvolvimento do trabalho.

O capítulo segundo aborda a fundamentação teórica para base de conhecimento dos itens abordados no trabalho conceituando a tecnologia Java, a edição Java para dispositivos móveis, o controle de acesso e padrões de projeto.

O terceiro capítulo trata do desenvolvimento do trabalho, subdividido nos objetivos do trabalho, especificação e implementação do mesmo.

As conclusões finais do trabalho são detalhadas no quarto capítulo, que também trata as dificuldades encontradas durante o desenvolvimento e sugestões para próximos trabalhos.

2 FUNDAMENTAÇÃO TEÓRICA

A fundamentação teórica traz o conceito dos itens abordados no trabalho que são divididos nos seguintes sub-capítulos:

a) controle de acesso;
b) dispositivos móveis;
c) tecnologia Java;
d) J2ME;
e) padrões de projeto.

2.1 CONTROLE DE ACESSO

A função principal de um sistema de controle de acesso é auxiliar de forma automatizada na segurança do patrimônio, garantindo que somente pessoas autorizadas possam freqüentar lugares e/ou eventos.

Pode-se dizer que os sistemas de controle de acesso têm grande influência de sistemas de recursos humanos para controle de ponto dos funcionários. O sistema de controle do ponto de funcionários automatiza o controle de freqüência em determinados horários, o que não está muito longe da base de um sistema de controle de acesso. A história de sistemas de controle de acesso, basicamente, inicia-se pelo aperfeiçoamento do controle de freqüência de funcionários.

Segundo a linha de soluções de Senior Sistemas (2004), no início, o foco principal era controlar o acesso a colaboradores da empresa. Mais tarde surgiu a necessidade de controlar também os visitantes e colaboradores terceiros. Nos dias atuais essa atuação não se restringe a apenas meios empresariais, mas também a instituições de ensino ou organizações de eventos públicos. Consideram-se disponíveis, dois tipos de controles de acesso:

a) controles únicos, onde a pessoa recebe uma identificação e a utiliza apenas uma única vez, descartando este identificador semelhante a um único crédito sem carga de reabastecimento desse crédito;
b) controles permanentes, onde a pessoa recebe um identificador e utiliza o mesmo periodicamente.

Um sistema de controle do acesso é composto por duas principais ações:

a) identificação lógica;

b) bloqueio físico.

Há uma série de tecnologias para identificação lógica do acesso no mercado atual. Entre elas, pode-se citar:

a) código identificador por pessoa;
b) crachá identificador com código de barras;
c) crachá identificador de proximidade;
d) crachá identificador com tecnologia *smart card*;
e) reconhecimento de íris;
f) reconhecimento de geometria da mão;
g) biométrica através do dedo;
h) utilização de *smart tags*.

A verificação lógica pode ser configurada para utilizar mais de uma tecnologia de verificação. Por exemplo, para controlar o acesso de um determinado local restrito em uma empresa, pode-se utilizar a tecnologia de crachá identificador em paralelo com a checagem da verificação biométrica da pessoa, aumentando assim a eficiência do controle.

Após a ação de identificação lógica, ocorre o bloqueio físico que depende do resultado da identificação. O bloqueio físico pode ser um hardware que recebe instruções do sistema após verificado o acesso ou um recurso humano que interpreta o resultado do sistema e toma as ações necessárias, permitindo ou não o acesso. Entre os hardwares utilizados para o bloqueio físico pode-se destacar:

a) catracas;
b) torniquetes;
c) cancelas;
d) fechaduras magnéticas.

Conforme Telemática Sistemas (2004), a verificação lógica comumente está agregada a um dispositivo de bloqueio físico de difícil mobilidade como os hardwares citados acima. Para tornar um sistema de controle de acesso um recurso independente de localização específica, podendo facilmente ser transferido e utilizado em outros locais, é necessário que a verificação lógica do acesso seja realizada em um dispositivo sem dependência ou sem estar acoplada ao hardware. Um dispositivo móvel do tipo PDA que faça a verificação lógica tornaria um sistema de controle de acesso móvel e independente de localização específica.

O avanço dos sistemas de controle de acesso oferecem verificações especiais, que automatizam cada vez mais a segurança patrimonial. Pode-se citar como verificações especiais a utilização de validade do identificador do acesso, verificação de situação de afastamento ou demissão do colaborador, checagem de pessoa não grata no recinto, controle de créditos de acesso, controle de faixas horárias de acesso e controle de nível do local do acesso.

O controle de faixas horárias de acesso é a verificação especial mais importante abordada neste trabalho. Este controle consiste da verificação do horário da tentativa do acesso com os horários permitidos para tal dia da semana. Pode-se imaginar o seguinte, para um funcionário informa-se que o mesmo tem acesso à empresa apenas durante o seu horário normal de trabalho, fora deste horário o funcionário não tem permissão de acesso. Ao tentar entrar no recinto em uma determinada hora fora do seu horário normal de trabalho, o sistema deve bloquear o acesso do funcionário pois a faixa horária não é válida para aquele determinado instante.

2.2 DISPOSITIVOS MÓVEIS

Pode-se considerar como dispositivo móvel, um dispositivo com capacidade de processamento em um ambiente de rede sem fio. Por se tratar de um ambiente de rede sem fio e por terem um pequeno volume, os dispositivos móveis têm alguns limites em relação à taxa de comunicação e processamento. Alguns dispositivos móveis encontrados no mercado hoje são:

a) telefones celulares;
b) *pagers*;
c) *personal digital assistants* (PDAs);
d) dispositivos embarcados;
e) computadores de bordo automotivo;
f) *handheld*.

A utilização de dispositivos móveis, para lazer ou trabalho, é algo que vem crescendo nos últimos anos no Brasil. Esse avanço acaba, por conseqüência, ampliando a oferta de novos serviços móveis como afirma TELECOMWEB (2005), que acrescenta que a taxa anual de crescimento dos serviços móveis no mundo deve chegar à 58%. De acordo com CAMARA-E (2005), no Brasil existem hoje mais de 40 milhões de usuários de comunicação

móvel. Esses números crescentes demonstram a forte tendência do uso de dispositivos móveis para soluções já existentes no mercado e até para novas soluções.

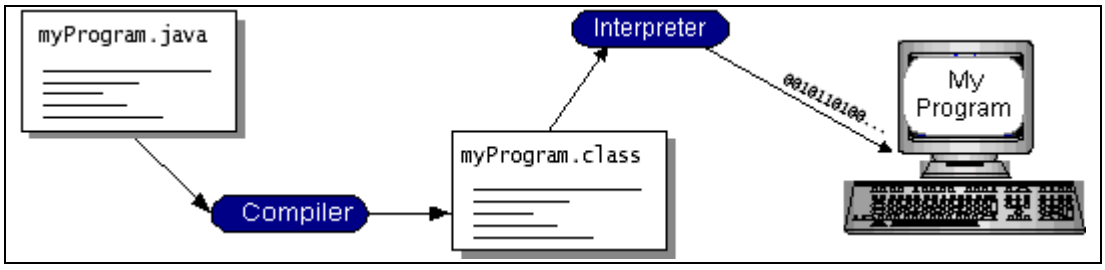
2.3 TECNOLOGIA JAVA

Uma linguagem de programação e uma plataforma (SUN JAVA, 2004): é assim que pode ser considerada a tecnologia Java.

Dentre as principais características da linguagem de programação Java pode-se citar:

a) orientada a objetos;
b) interpretada;
c) portável em plataformas distintas.

Uma das grandes vantagens da tecnologia Java é a utilização do recurso da máquina virtual. Com a máquina virtual Java é possível codificar o programa uma vez e rodá-lo em todos os sistemas operacionais que suportam uma máquina virtual Java independente de modificações no código. Isto somente é possível pela utilização do recurso de Java *bytecodes*, que é o código Java compilado para uma máquina virtual Java. O Java *bytecode* é posteriormente interpretado por uma máquina virtual que roda o programa independente da plataforma. A Figura 1 exemplifica o caminho do código Java até chegar à plataforma onde será rodada a aplicação.



Fonte: Sun Java (2004).

Figura 1- Caminho percorrido pelo código Java

Uma plataforma é um ambiente de hardware ou software no qual os sistemas são executados. A plataforma Java é uma plataforma de software que trabalha sobre uma plataforma baseada em hardware. Segundo Sun Java (2004), a plataforma Java é composta por dois componentes:

a) máquina virtual (*Virtual Machine* - VM);
b) interface de programação de aplicação (*Java Aplication Program Interface* - API).

De acordo com Sun Java (2004), no início dos anos 90, a visão de que o mercado de dispositivos eletrônicos inteligentes seria um nicho de tecnologia bastante procurado pelos consumidores finais, fez com que a Sun Microsystems financiasse um projeto interno chamado Green. O projeto Green resultou numa linguagem de programação chamada Oak, que era baseada nas linguagens C e C++. Alguns anos mais tarde a tal linguagem Oak criada pela Sun passou a se chamar Java por já existir uma outra linguagem com tal nome. O projeto não estava muito bem, pois o mercado de eletrônicos inteligentes não havia se desenvolvido conforme a Sun esperava, e o projeto estava para ser abortado, quando em 1993 a utilização da *World Wide Web* fez com que a Sun reaproveitasse a linguagem Java para disponibilizar conteúdo dinâmico nas páginas *Web*. Mais tarde, com a criação da Java 2 Plataform, a Sun achou necessário subdividir Java em 3 partes, as consideradas edições de Java.

Uma edição da plataforma Java é uma versão definitiva e concordada da plataforma que provê funcionalidades para um segmento de mercado e aplicação. Cada edição é composta por dois tipos de conjuntos de funções (API):

a) conjunto de funções do núcleo (*Core*);
b) conjunto de funções opcionais.

A plataforma Java contém 3 diferentes edições (SUN JAVA, 2004):

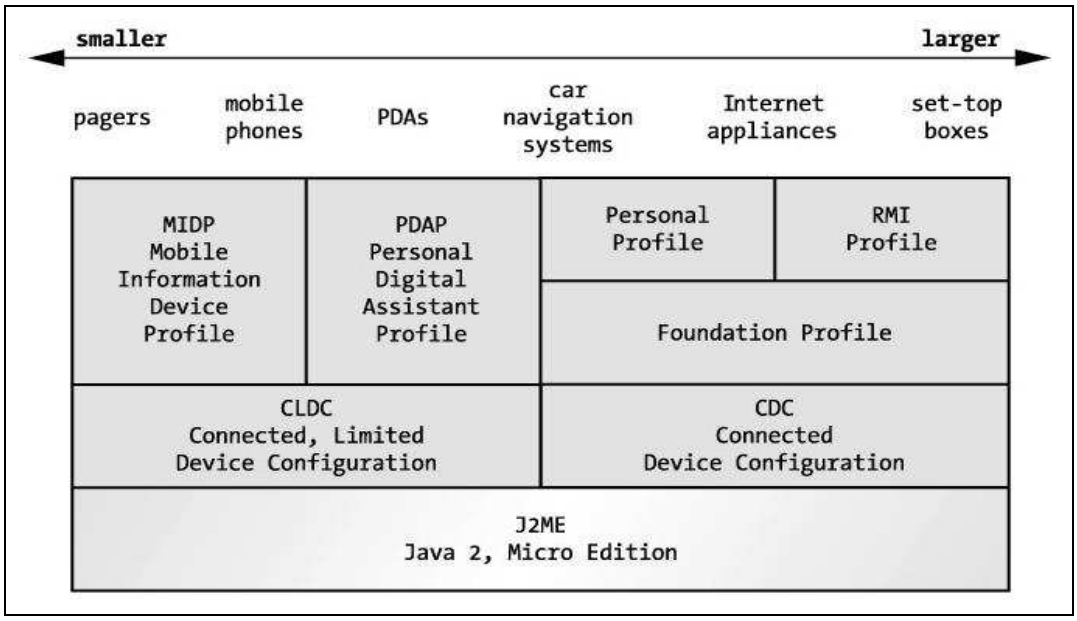
a) *Java 2 Plataform, Enterprise Edition* (J2EE): plataforma para construção de aplicações servidoras;
b) *Java 2 Plataform, Micro Edition* (J2ME): plataforma para pequenos dispositivos como telefones celulares, *handhelds*, *PDAs*, etc.;
c) *Java 2 Plataform, Standard Edition* (J2SE): plataforma para construção de aplicações cliente.

2.4 J2ME

Conforme história da linguagem Java (SUN JAVA, 2004), criada para atender ao nicho de dispositivos eletrônicos inteligentes, pode-se dizer que a necessidade de atender recursos computacionais aos dispositivos eletrônicos móveis, o qual é foco da J2ME, deu início a tecnologia Java.

A edição micro da plataforma Java propõe soluções para dispositivos móveis como telefone celular, *pager*, assistente pessoal digital (PDAs), *handheld*, dispositivo embarcado,

computador de bordo automotivo entre outros. Estes dispositivos se assemelham pela baixa capacidade de processamento, baixo volume de memória virtual e persistente, mínima interface gráfica e poucos recursos de conectividade. Porém, cada dispositivo é construído com seu propósito e tem suas características diferenciadas para atender a este propósito. Em conseqüência disto, as aplicações que atendem a estes propósitos são diferenciadas umas das outras, utilizando-se de recursos do dispositivo que poderiam não estar disponíveis em outros ambientes como conectividade, processamento, interface gráfica ou memória. Para atender a estas diferenças, J2ME dispõe um conjunto de componentes como perfis, configurações e pacotes opcionais conforme demonstrado na Figura 2.



Fonte: Knudsen (2003).

Figura 2 - Universo J2ME

As configurações do J2ME definem uma plataforma Java para uma determinada faixa de dispositivos definidos por uma série de características como:

a) informações sobre a memória do dispositivo;
b) tipo e velocidade do processador;
c) conectividade do aparelho.

Uma configuração obrigatoriamente representa as mínimas características de uma plataforma para um dispositivo, não podendo definir características opcionais (TOPLEY, 2002). Com isso, os fabricantes são obrigados a implementar por total a especificação de uma

configuração almejada para tal dispositivo. Isso garante que programadores possam desenvolver aplicações sem se preocupar em qual dispositivo específico elas irão rodar, apenas precisam da certeza que o dispositivo implementa a configuração escolhida. Há duas configurações disponíveis para a utilização do J2ME, a *Connected Device Configuration* (CDC), voltada para dispositivos com maior poder operacional e recursos de memória e conectividade, e a *Connected Limited Device Configuration* (CLDC), voltada para dispositivos com recursos limitados e que é a configuração escolhida neste trabalho.

De acordo com Topley (2002), os perfis complementam as configurações com a adição de classes que provêm características específicas para um determinado tipo de dispositivo ou um determinado segmento de aplicação, como recursos de interface gráfica, armazenamento persistente, segurança e conectividade. Cada configuração da J2ME, tem um ou mais perfis associados. Segundo White (2002), alguns perfis associados à configuração CDC são:

a) *Foundation Profile*;
b) *Personal Profile*;
c) *Remote Method Invocation* (RMI) *Profile*;
d) *Personal Basis Profile*.

White (2002) descreve também que os seguintes perfis são referentes a configuração CLDC:

a) *Mobile Information Device Profile* (MIDP);
b) *Personal Digital Assistant Profile* (PDAP).

Os perfis *Multimedia*, *Gaming* e *Telephony Profile* atendem tanto a configuração CDC como CLDC.

2.4.1 Configuração CLDC

A *Connected Limited Device Configuration* (CLDC) é destinada para os dispositivos com as mais limitadas configurações do mercado, como telefones celulares, *pagers*, simples PDAs, entre outros dispositivos, que são caracterizados pelos baixos recursos de memória, processamento e conectividade. A CLDC é a configuração que provê funções de núcleo que são usadas como base para alguns perfis.

Como características principais, os dispositivos que suportam a CLDC devem ter:

a) no mínimo 192 kb de memória total disponível para a plataforma Java;

b) um processador de 16 ou 32 bits;
c) conectividade a algum tipo de rede por meio de tecnologia sem fio, conexão intermitente e com banda limitada.

Cada configuração do J2ME consiste de um conjunto de bibliotecas e de uma máquina virtual Java. A CLDC define uma máquina virtual que pode ser considerada uma versão simplificada da *Java Virtual Machine* (JVM) utilizada na edição J2SE. A Sun especifica como deve trabalhar uma máquina virtual para a configuração CLDC e também oferece uma máquina virtual junto ao kit de desenvolvimento para J2ME chamada *Kilobyte Virtual Machine* (KVM). Porém, a utilização da KVM em aplicações J2ME não é obrigatória, podendo assim cada fabricante de dispositivos móveis implementar a sua máquina virtual que atenda à especificação.

Pela simplificação da máquina virtual na CLDC, alguns dos recursos da linguagem Java e funções da máquina virtual usadas na J2SE não são suportados. Muitos dispositivos, pela baixa disponibilidade de memória, não suportam operações com ponto flutuante, e por padrão a utilização do ponto flutuante não é suportada pela CLDC. O baixo recurso de memória nos dispositivos é a principal causa de algumas omissões na linguagem Java da configuração CLDC como inviabilidade de recursos de reflexão, não implementação de finalização de objetos, restrições na utilização de *threads*, não suporte de várias classes de erros e exceções e omissão do recurso de interface nativa.

2.4.2 Pefil MIDP

O perfil *Mobile Information Device Profile* (MIDP) trabalha sobre a configuração CLDC e habilita recursos de conectividade, armazenamento de dados local e interface gráfica com usuário. Por padrão, o perfil MIDP carece de recursos avançados de segurança e interface gráfica, porém os fabricantes dos dispositivos podem oferecer recursos opcionais que customizam essas carências. Uma aplicação que implementa o perfil MIDP é chamada de MIDlet e roda sobre a máquina virtual KVM proposta pela configuração CLDC. Um MIDlet pode utilizar tanto funções oferecidas no perfil MIDP como funções que o MIDP herda da CLDC.

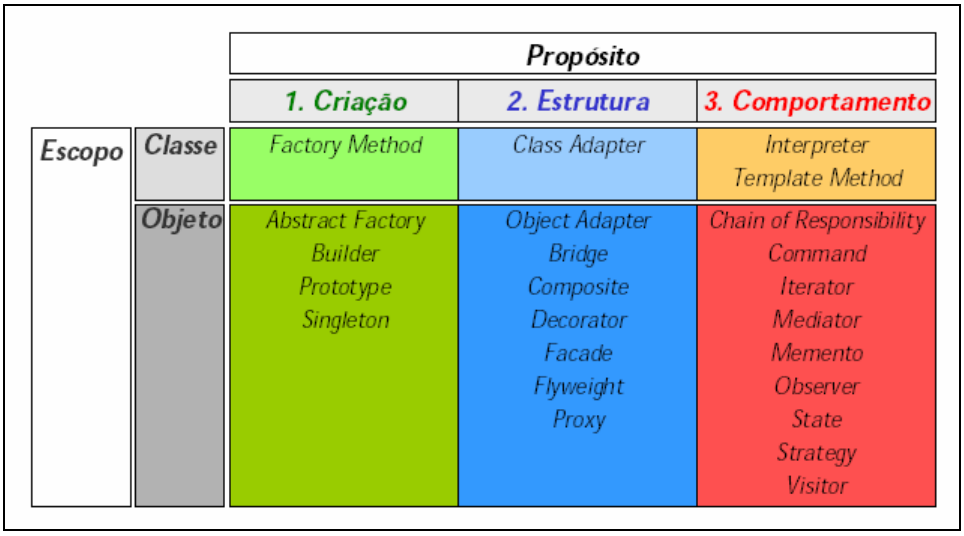
Os dispositivos atendidos pelo perfil MIDP são normalmente telefones celulares e PDAs. De acordo com a especificação deste perfil, o dispositivo deve ter:

a) 128 kb de memória não volátil necessária para armazenamento da implementação;
b) 32 kb de memória volátil para a pilha Java;
c) 8 kb de memória não volátil necessária para armazenamento local persistente;
d) visor de pelo menos 96 x 54 pixels;
e) algum recursos para entrada de dados como teclado, tela por toque ou área de teclas conforme telefones celulares;
f) possibilidade de conexão a algum tipo de rede.

Uma aplicação MIDP contém no mínimo uma classe MIDlet. Porém pode haver casos em que uma aplicação contém mais de um MIDlet, e neste caso a aplicação é chamada de *MIDlet Suite*. Ao ser carregada no dispositivo, a *MIDlet Suite*, faz a varredura de todos os MIDlets existentes e dispõe ao usuário escolher qual MIDlet lançar. A aplicação MIDP, para ser instalada no dispositivo, precisa ser empacotada, construindo um *Java Archive* (JAR). O JAR obrigatoriamente contém um arquivo de manifesto que engloba informações sobre o MIDlet ou conjunto de MIDlets contidos no pacote JAR. O JAR trabalha em conjunto com um *Java Application Descriptor* (JAD). O descritor contém informações sobre a aplicação, principalmente sobre a configuração e perfil utilizados pela mesma.

2.5 PADRÕES DE PROJETO

Segundo Argonavis (2004), os padrões de projeto são aplicados na engenharia de software para alcançar objetivos usando classes e métodos em linguagens orientadas a objeto. O emprego de padrões de projeto ajuda o desenvolvimento de sistemas, como na documentação, na manutenção e na qualidade do produto. Gamma (2000) descreve 23 padrões de projeto clássicos que são divididos em propósito e escopo. Os 23 padrões de projeto podem ser vistos na Figura 3.



Fonte: Argonavis (2004).

Figura 3 - 23 padrões de projetos descritos por Gamma

Cada padrão de projeto é caracterizado pelos seguintes elementos:

a) nome;
b) problema a ser resolvido;
c) solução aplicável para resolver o problema;
d) conseqüências da aplicação do padrão.

Apesar de serem descritos 23 padrões ao todo, serão detalhados a seguir apenas os padrões *Factory Method*, *Singleton* e *Strategy*, pois foram estes os padrões aplicados no projeto que resultaram em soluções mais eficientes e elegantes para concluir alguns objetivos do trabalho.

2.5.1 Padrão de projeto *Factory Method*

Segundo Gamma (2000), a finalidade do padrão *Factory Method* é definir uma interface para criar um objeto, mas deixar que subclasses decidam que classe instanciar. Este padrão pode garantir uma flexibilidade da aplicação. Para esse caso é necessário pelo menos duas classes concretas que implementam a classe abstrata de diferentes formas. A criação das classes concretas depende de uma definição que distingue umas das outras, podendo assim criar um objeto que difere de outro na execução de determinado método.

2.5.2 Padrão de projeto *Singleton*

Este padrão tem como finalidade garantir que uma classe seja instanciada apenas uma vez em todo o sistema (GAMMA, 2000). Normalmente utilizado para garantir controles

centrais e únicos. Trabalha a visibilidade do método construtor da classe de forma privada, não podendo ser criada por classes externas. Usa um método público que retorna a instância única do objeto.

2.5.3 Padrão de projeto *Strategy*

Gamma (2000) define que a finalidade do padrão *Strategy* é definir uma família de algoritmos, encapsular cada um, e fazê-los intercambiáveis. O padrão *Strategy* pode substituir, em forma de classes executoras, um algoritmo que verifica algum tipo de parâmetro e executa uma ação comum, porém em uma classe concreta diferenciada para aquele parâmetro. Para tal controle há uma interface que define os métodos da estratégia. As classes concretas precisam implementar esses métodos e o cliente que utilizar a interface pode passar um parâmetro na criação da mesma que, por exemplo, justifique a criação de uma ou outra classe concreta que implementa a interface de estratégia. Comumente usado quando é necessário diversas soluções para um mesmo algoritmo ou quando classes relacionadas diferem apenas no seu comportamento.

2.6 SERVLETS JAVA

Servlet Java é uma aplicação Java para ser executada num servidor e atender a requisições *web*. Semelhante a uma página de internet, o *servlet* recebe a requisição com seus parâmetros e trata a requisição de formas diversas, podendo consultar o banco de dados ou até mesmo utilizar uma requisição para outro *servlet*.

A chamada da requisição ao *servlet* pode ser realizada através de um endereço *Hypertext Transfer Protocol* (HTTP) executado ou através de um formulário *web*. Os parâmetros podem ser passados no corpo do formulário ou concatenados na *Universal Resource Locator* (URL) executada. Cabe ao *servlet* receber e identificar esses parâmetros e realizar um processo que atende tal ação chamada. O resultado desse processamento é dado em informações literais, podendo ser uma página Hypertext Transfer Markup Language (HTML) ou dados no formato Extensible Markup Language (XML). A aplicação que solicitou o serviço do *servlet* deve saber tratar o resultado obtido.

3 DESENVOLVIMENTO DO TRABALHO

O protótipo de *front end* resultante deste trabalho é um software para controle de acesso voltado à execução em dispositivos móveis e uma aplicação *web* que trata e responde requisições do sistema móvel. Pela portabilidade da edição J2ME, a aplicação do dispositivo móvel é independente de plataforma, sendo necessário apenas que o dispositivo atenda às necessidades do perfil MIDP e à configuração CLDC escolhidos para tal solução.

Para desenvolvimento do trabalho foi necessário um levantamento e análise dos requisitos que definem as características que o sistema deve ter. Uma especificação se fez necessária para expressar por intermédio de diagramas como os requisitos serão tratados no trabalho. A análise dos requisitos e especificação são tratados a seguir.

A experiência profissional do autor na área de sistemas de controle de acesso teve grande influência no desenvolvimento do trabalho. Trabalhando há mais de três anos no desenvolvimento do sistema Ronda Acesso e Segurança da Senior Sistemas, o autor adquiriu conhecimento para tratar das regras de negócio do protótipo resultante deste trabalho. Por trabalhar na área de desenvolvimento de sistema o autor também tem conhecimento sobre alguns dispositivos usados atualmente nos controles de acesso do mercado.

3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

Os requisitos funcionais declaram e detalham quais são as principais características da camada de negócio do sistema. Características de ambiente, performance ou disposição são tratados como requisitos não funcionais. São requisitos funcionais do sistema:

a) leitura do identificador do acesso;
b) verificação de liberação ou bloqueio do acesso;
c) gravação das informações do acesso;
d) configuração para armazenamento das informações do acesso;
e) configuração para visualização das informações do acesso e da pessoa;
f) possibilidade de controle de faixa horária do acesso;
g) importação de identificadores do acesso liberados;
h) configuração da fonte de informações do acesso;
i) exportação dos acessos registrados.

O sistema fará a leitura do identificador do acesso através de uma tela inicial onde o identificador deve ser informado (digitado). Não é função do sistema fazer a leitura do identificador através de uma leitora de cartões.

Após realizar a leitura do identificador do acesso, o sistema deve realizar a verificação de liberação ou bloqueio. A verificação pode checar as informações em um banco de dados ou em memória do aplicativo.

O sistema deve gravar as informações da tentativa de acesso, sendo ela liberada ou bloqueada. A informação pode ser gravada em memória ou em banco de dados de acordo com a configuração para armazenamento das informações do acesso. Para atender a gravação do acesso, o sistema deve prever uma configuração de como as informações do acesso devem ser gravadas.

É necessário prever um controle para escolha de quais informações serão mostradas ao usuário no momento da verificação do acesso. As seguintes informações podem ser apresentadas:

a) identificador do acesso;
b) nome da pessoa;
c) situação do acesso da pessoa;
d) data e hora do acesso.

O sistema deve prever de forma opcional o controle de faixa horária na verificação do acesso da pessoa.

Para formar uma lista de identificadores com acesso ao recinto controlado pelo dispositivo, o sistema deve prever uma rotina de importação de dados que carregue no dispositivo uma lista de identificadores e pessoas que tem acesso liberado.

O sistema deve prever uma configuração da fonte de informações que poderão ser importadas ou que serão consultadas no sistema. Neste momento será considerado como fonte de informações apenas um serviço web. Em outra situação caberia ter como fonte de informações um arquivo XML ou outro formato/definição qualquer.

Se faz necessário um recurso para exportação dos acessos registrados pelo sistema, que será usado no caso da configuração estar como armazenamento local. A exportação deverá ser realizada através da aplicação *web* configurada.

3.2 REQUISITOS NÃO FUNCIONAIS

Requisitos de performance e disposição são tratados como não-funcionais, mas nem por isso sua importância é descartada. São requisitos não funcionais do sistema:

a) portabilidade em dispositivos móveis usando a configuração CLDC;
b) compatibilidade com recursos do perfil MIDP;
c) resposta de validação do acesso em intervalo inferior ou igual a dois segundos.

3.3 ESPECIFICAÇÃO

A *Unified Modeling Language* (UML) é uma linguagem unificada para modelagem de sistemas orientados a objeto que, de acordo com OMG (2004), define doze tipos de diagramas divididos em três categorias:

a) categoria estrutural, onde se aplicam os diagramas de classe, objetos, componentes, implantação;
b) categoria comportamental, onde se aplicam os diagramas de caso de uso, seqüência, atividade, colaboração e estado;
c) categoria de gerenciamento de modelos, onde se aplicam os diagramas de pacote, subsistema e modelos.

A modelagem ágil (AMBLER, 2003) é uma metodologia baseada na prática para modelagem efetiva de sistemas usando a linguagem UML. A prática de uma modelagem ágil agiliza o processo de modelagem, fazendo com que o projeto seja produtivo e se invista tempo apenas em funções que são vitais para o projeto, buscando a simplicidade e objetivo do projeto ao máximo. Seguiu-se no trabalho apenas alguns princípios da modelagem ágil, e não toda a metodologia. Os princípios utilizados foram a simplicidade dos modelos, porém resultando num detalhamento elegante, atendendo aos seus propósitos e oferecendo um fácil entendimento.

A seguir serão abordados os casos de uso do problema, diagrama de atividades de casos de uso, diagrama de classes e diagrama de realização de casos de uso. Também é abordada a ferramenta utilizada na especificação.

3.3.1 CASOS DE USO DO PROBLEMA

Os casos de uso representam a interação do usuário com o sistema, destacando as ações que serão realizadas. No protótipo do trabalho foram definidos sete casos de uso:

a) importar informações;
b) verificar acesso;
c) registrar acesso;
d) configurar armazenamento de informações;
e) configurar apresentação dos resultados;
f) configurar fonte de informações;
g) exportar acessos registrados.

A seguir os casos de uso do sistema são detalhados. A Figura 4 demonstra o diagrama dos casos de uso do sistema.

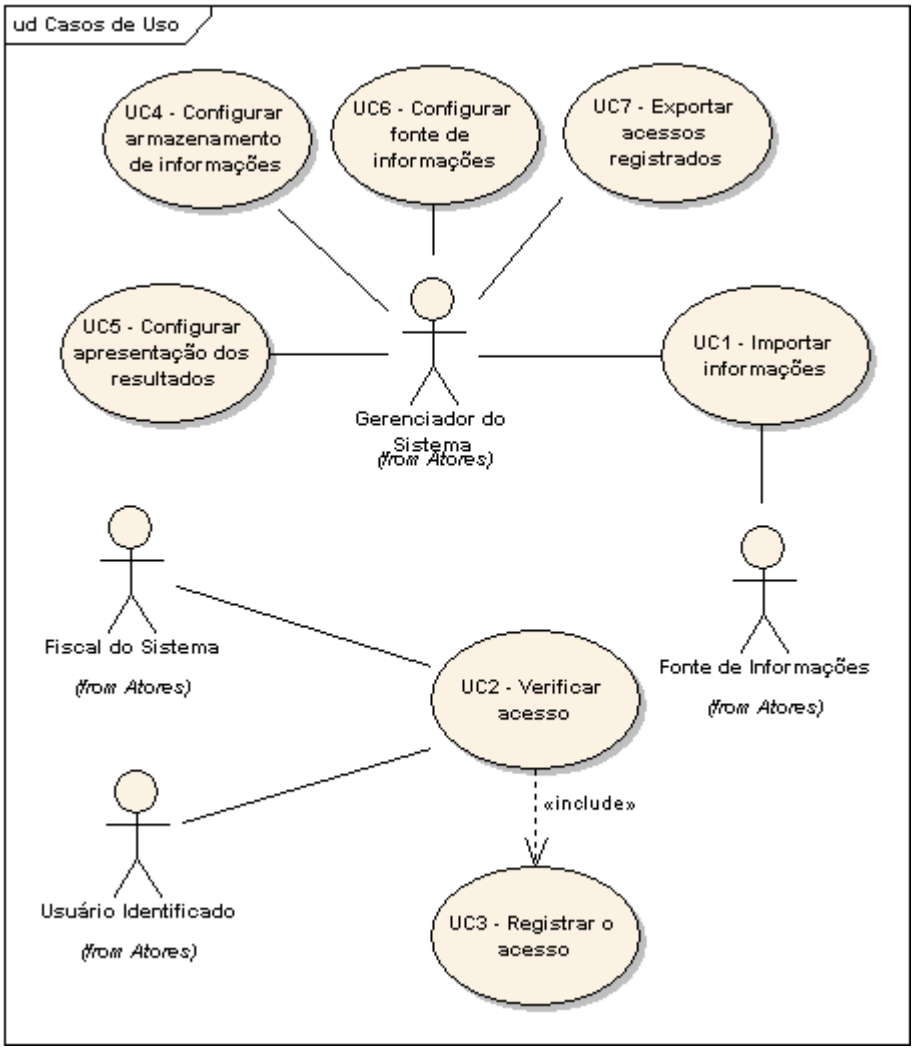


Figura 4 - Relação entre casos de uso e atores do sistema

3.3.1.1 Caso de uso: importar informações

Há um sistema de recursos humanos ou controle de pessoas e/ou funcionários que tem informações de pessoas. Cada pessoa pode ser responsável por um componente de identificação de acesso que contenha as seguintes informações:

a) identificador;
b) período de validade;
c) status (liberado ou bloqueado);
d) faixa horária, contendo os intervalos de tempo para os dias da semana.

O ator gerenciador do sistema é responsável por fazer a importação manual das informações do acesso. No momento da importação, o sistema deve importar apenas identificadores com período válido e que tenham permissão de acesso para o dispositivo em questão.

3.3.1.2 Caso de uso: verificar acesso

O usuário, ao entrar no recinto que é controlado pelo sistema, informa ao mesmo qual o seu identificador de acesso. O sistema verifica de forma *on-line* no banco de dados por intermédio de uma aplicação *web* ou em listas de dados persistentes no dispositivo se este identificador de acesso tem permissão para acessar o local controlado. Se o identificador tiver permissão de acesso, a faixa horária de acesso é verificada caso o dispositivo esteja configurado para verificar faixa horária. Independente da situação, o sistema registra o acesso. O fiscal do sistema interpreta essa informação verificada e toma as ações necessárias, liberando ou bloqueando fisicamente o usuário com identificador.

3.3.1.3 Caso de uso: registrar acesso

Após verificado o acesso, o sistema busca qual a forma de armazenamento de informações, podendo ser no banco de dados ou persistente no dispositivo, e faz o registro do acesso com as informações do identificador, data e hora do acesso, situação do acesso e local do acesso.

3.3.1.4 Caso de uso: configurar armazenamento de informações

O Gerenciador do sistema escolhe o local de armazenamento do registro do acesso que pode ser em memória persistente do dispositivo ou em banco de dados. A opção de banco de dados requer configuração da aplicação *web*.

3.3.1.5 Caso de uso: configurar apresentação dos resultados

O gerenciador do sistema define quais informações serão apresentadas ao usuário identificado após o sistema verificar o acesso. São opções apresentar nome, identificador, data e hora do acesso e situação do acesso.

3.3.1.6 Caso de uso: configurar fonte de informações

O gerenciador do sistema deve configurar uma fonte de informações de onde as informações do acesso serão consultadas ou importadas. A fonte de informações deve ser considerada como uma aplicação *web*. Essa aplicação *web* trata de realizar uma conexão com um banco de dados. Cabe ao gerenciador do sistema definir conexões de acesso ao banco de dados na aplicação *web* e configurar no dispositivo a URL de conexão com a aplicação *web* através do protocolo HTTP.

3.3.1.7 Caso de uso: exportar acessos registrados

Após verificados, os acessos são registrados. Os acessos armazenados localmente, conforme configuração escolhida, precisam em algum momento ser exportados ao banco de dados. O sistema deve prever uma rotina de exportação dessas informações, que serão enviadas a aplicação *web* e esta se encarregará de armazená-los no banco de dados.

3.3.2 DIAGRAMA DE ATIVIDADES DOS CASOS DE USO

Um diagrama de atividades descreve o fluxo de funções ou passos que são realizados desde o início do caso de uso até seu encerramento. Para cada caso de uso do trabalho criou-se um diagrama de atividades ilustrando o fluxo de atividades. A seguir, podem-se acompanhar os diagramas de atividades para cada caso de uso do sistema.

A Figura 5 demonstra o diagrama de atividades do caso de uso “importar informações do acesso”.

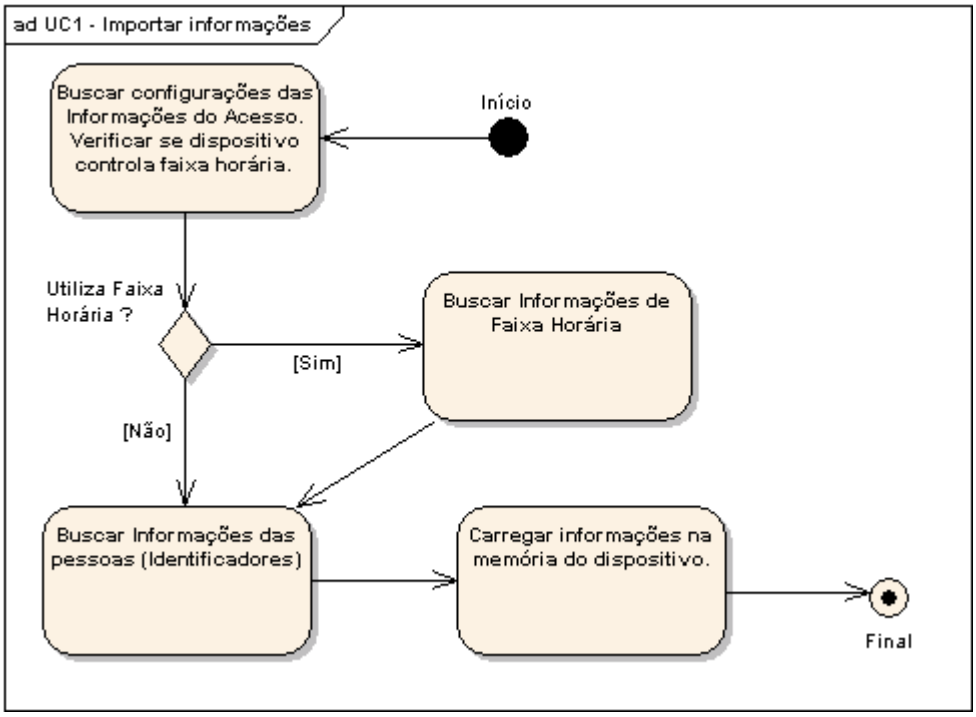


Figura 5 - Diagrama de atividades do caso de uso “importar informações”

A Figura 6 detalha o diagrama de atividades do caso de uso “verificar acesso”.

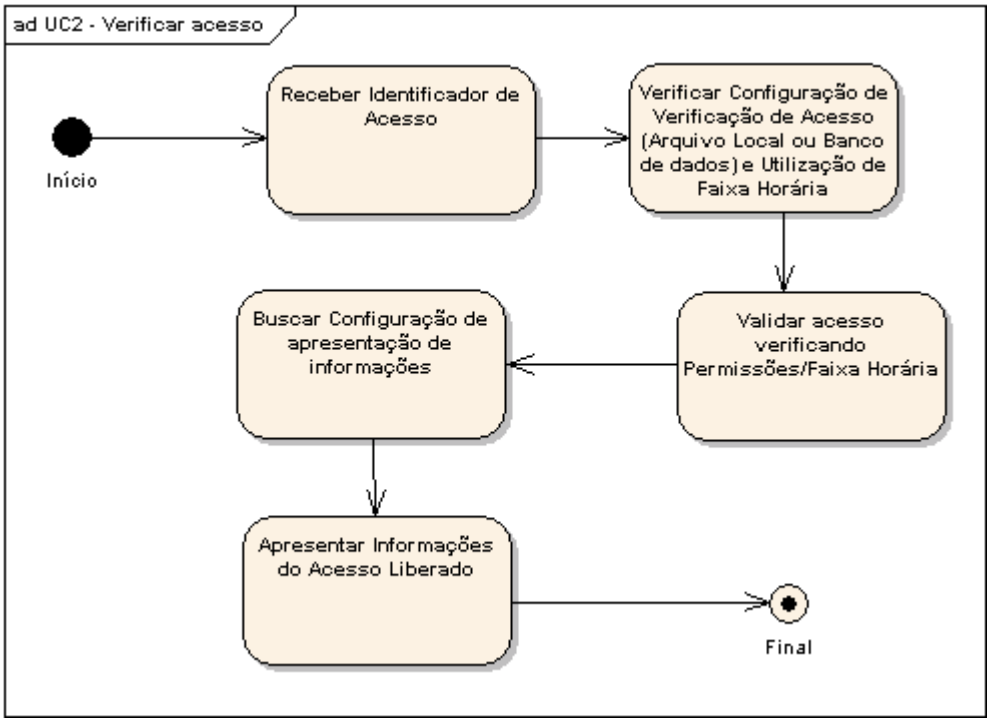


Figura 6 - Diagrama de atividades do caso de uso “verificar acesso”

O diagrama de atividades do caso de uso “registrar acesso” é demonstrado na Figura 7.

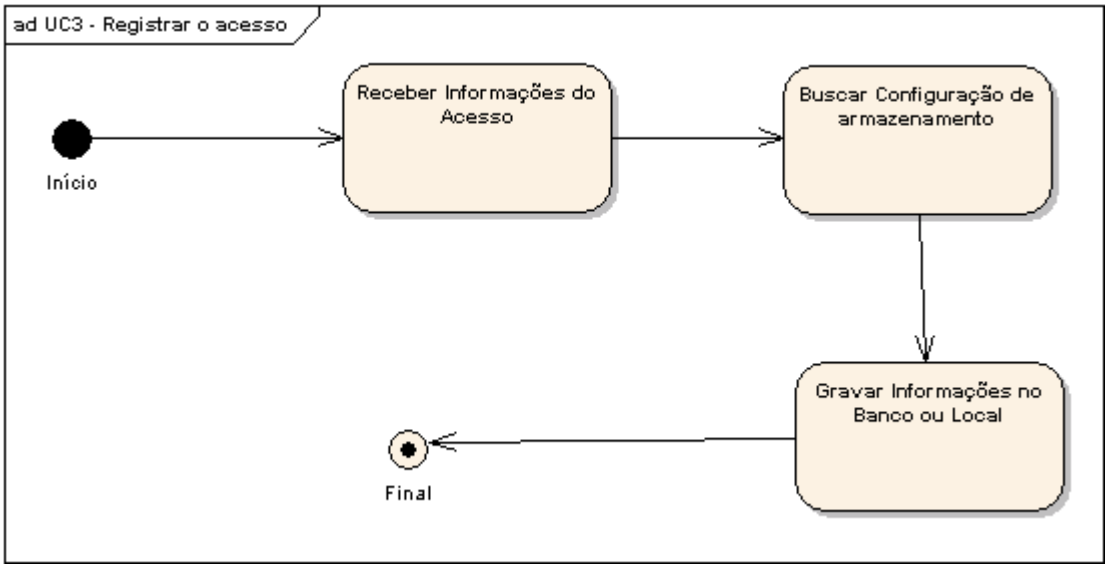


Figura 7 - Diagrama de atividades do caso de uso “registrar acesso”

O diagrama de atividades dos casos de uso “configurar armazenamento de informações”, “configurar apresentação dos resultados” e “configurar fonte de informações” podem ser vistos nas Figura 8, Figura 9, Figura 10 respectivamente.

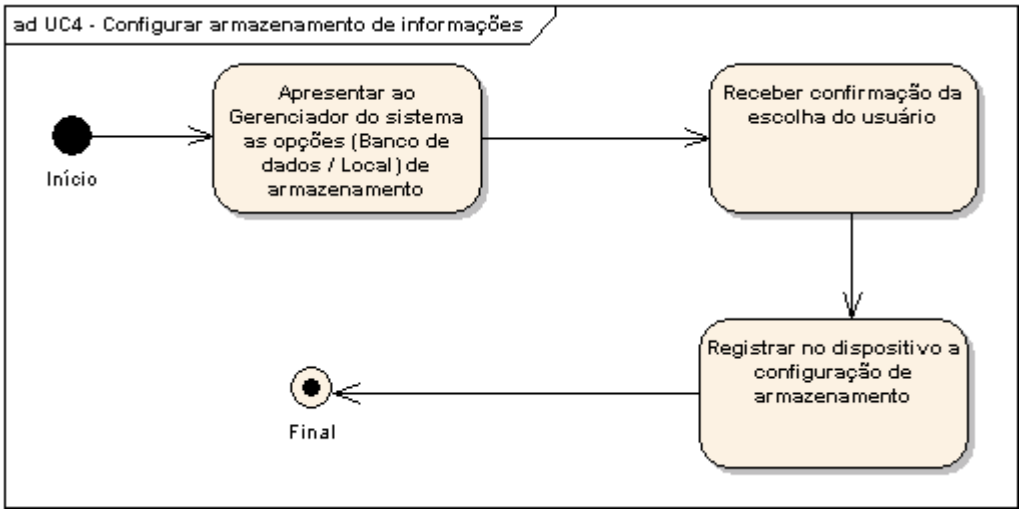


Figura 8 - Diagrama de atividades do caso de uso “configurar armazenamento de informações”

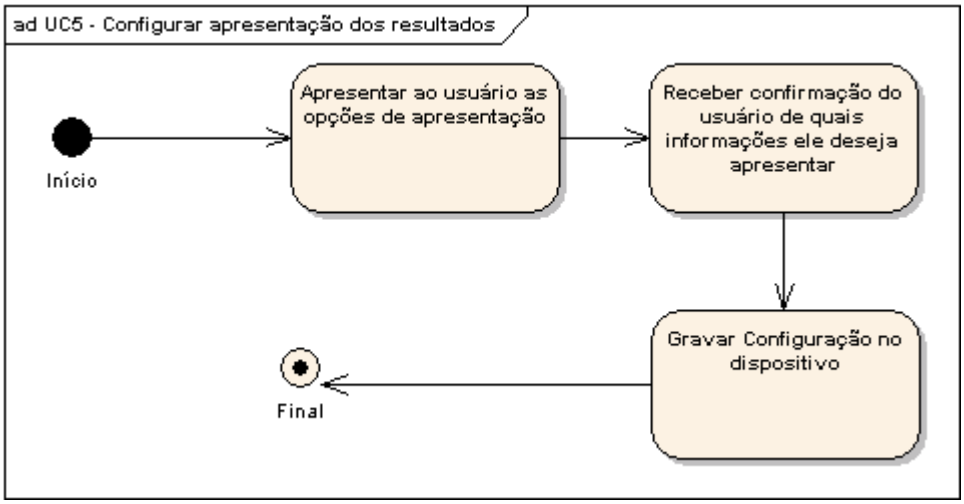


Figura 9 - Diagrama de atividades do caso de uso “configurar apresentação dos resultados”

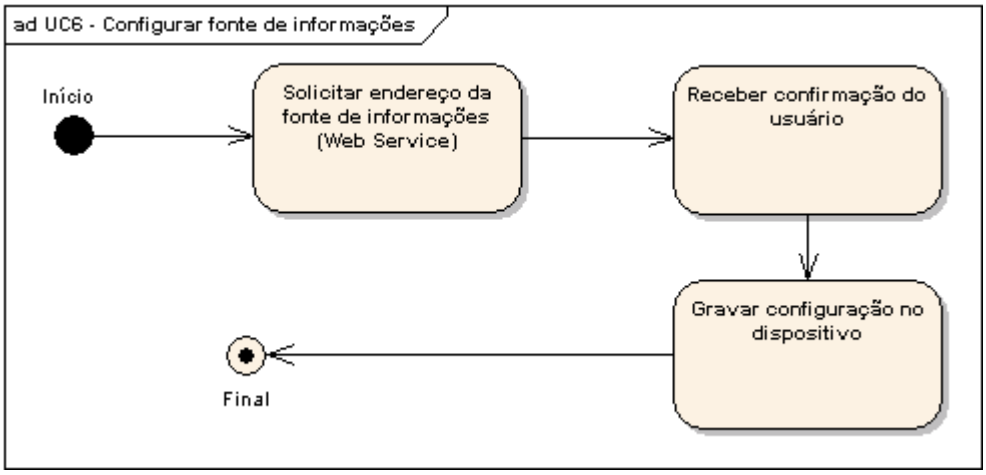


Figura 10 - Diagrama de atividades do caso de uso “configurar fonte de informações”

O caso de uso “exportar acessos registrados” tem seu diagrama de atividade demonstrado na Figura 11.

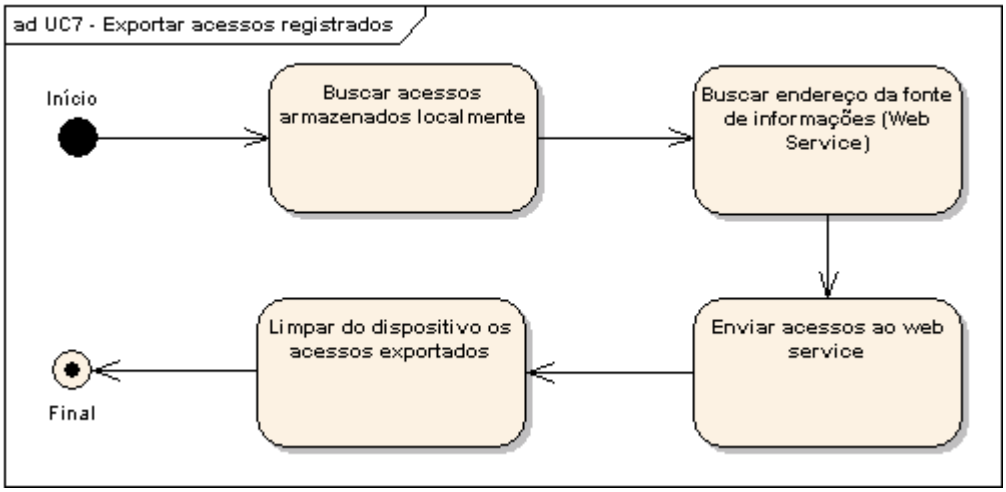


Figura 11- Diagrama de atividades do caso de uso “exportar acessos registrados”

3.3.3 DIAGRAMA DE CLASSES

Em um sistema com programação orientada a objetos, as classes podem ser consideradas o menor conjunto de informações que formam o grande sistema. O diagrama de classes demonstra uma visão lógica de como as classes do sistema estão associadas, podendo também identificar o sentido da visibilidade entre elas. Também é função do diagrama de classes demonstrar os atributos e métodos das classes do sistema.

A Figura 12 demonstra o diagrama de classes do sistema numa camada voltada ao negócio da solução, abstraindo classes supérfluas para a camada de negócio.

A classe “Dispositivo” é a principal classe do sistema. Só permite uma instância em todo sistema, pois é ela que centraliza os demais controles. Praticamente todas as ações do sistema passam por uma chamada do objeto “Dispositivo”. É responsável por armazenar a informação do código de permissão de acesso local e a utilização ou não da verificação de faixa horária.

A classe “ConexaoWebService” trata de armazenar o endereço da aplicação *web* e define um método genérico para envio de requisições à aplicação *web*.

A classe “ApresentacaoInformacoes” é responsável por armazenar se as informações de situação do acesso, nome da pessoa, identificador de acesso e data e hora do acesso serão apresentadas após a validação do acesso.

A classe “InformacoesAcesso” é responsável por armazenar e manipular as informações do acesso importadas da base de dados. Define métodos abstratos que as subclasses devem implementar para garantir a importação das informações e validação dos identificadores.

A classe “IdentificadorAcesso” é o controle principal de uma informação do acesso. Por intermédio desta classe consegue-se localizar a pessoa e a faixa horária da mesma. Controla o código do identificador do acesso e validade do histórico de identificador do acesso.

A classe “FaixaHoraria” possui uma lista de intervalos que determinam a faixa horário de acesso do identificador e contém métodos para controle de sua classe interna

“IntervaloTempo”. A classe “IntervaloTempo” controla os intervalos de tempo possíveis na semana que o identificador tem acesso ao local controlado.

Referenciada pela classe “IdentificadorAcesso” a classe “Pessoa” tem o objetivo de controlar as informações da pessoa responsável pelo identificador de acesso. No trabalho é apenas controlado o nome da pessoa.

A classe “RegistroAcesso” é responsável por armazenar e manipular os acessos registrados no dispositivo. Define um método abstrato para registro do acesso que deve ser implementado pelas subclasses. É responsável pela função de exportar os acessos registrados para a aplicação web.

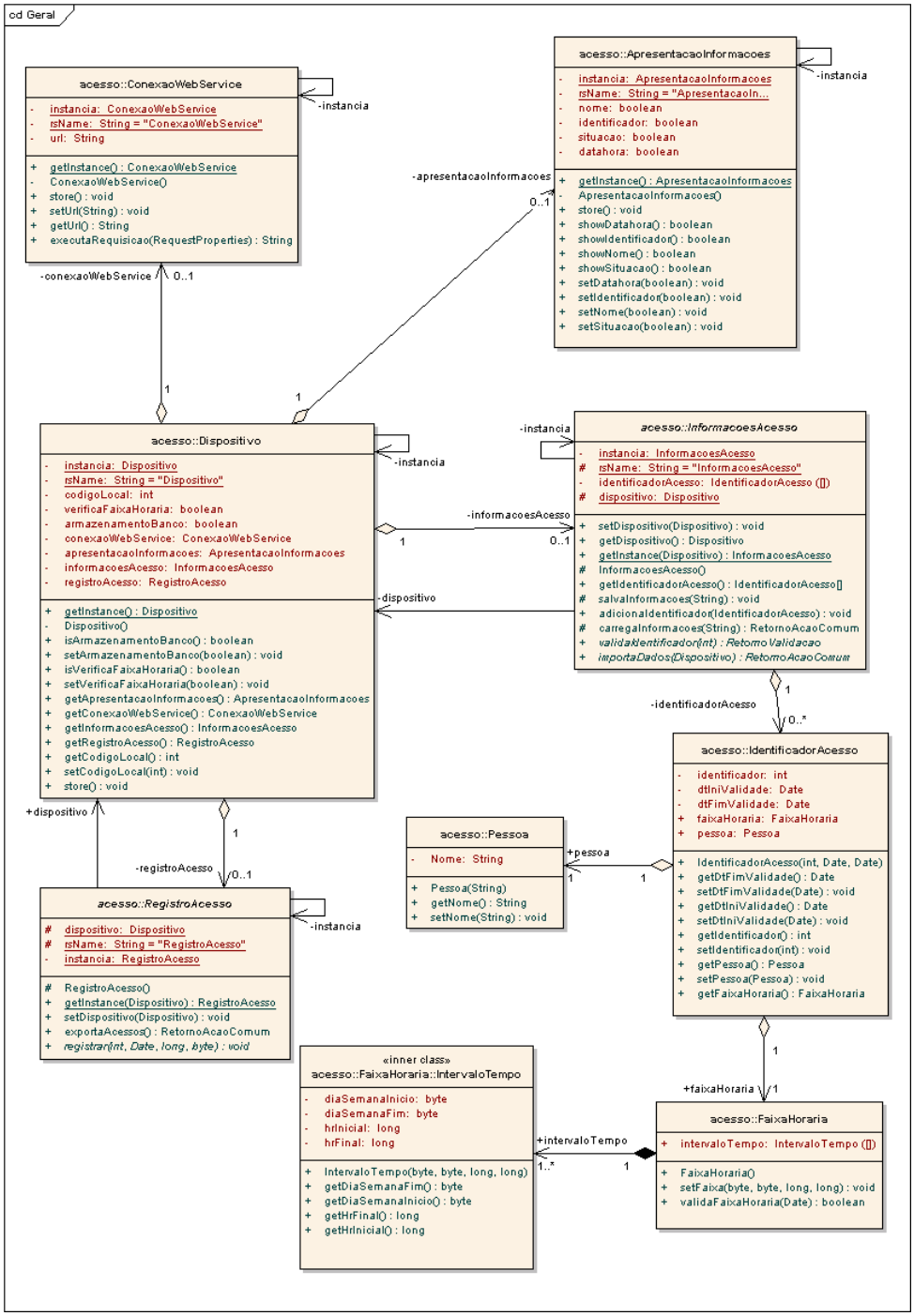


Figura 12 - Diagrama de classes da numa visão de negócio do sistema

Além das classes demonstradas na camada de negócio utilizaram-se classes que supriram algumas necessidades para se chegar ao resultado final. Na Figura 13 pode-se notar um diagrama de classes do assunto “informações do acesso”, demonstrando duas classes que implementam os métodos abstratos da classe herdada e duas que foram utilizadas no *parser* das informações recebidos por XML.

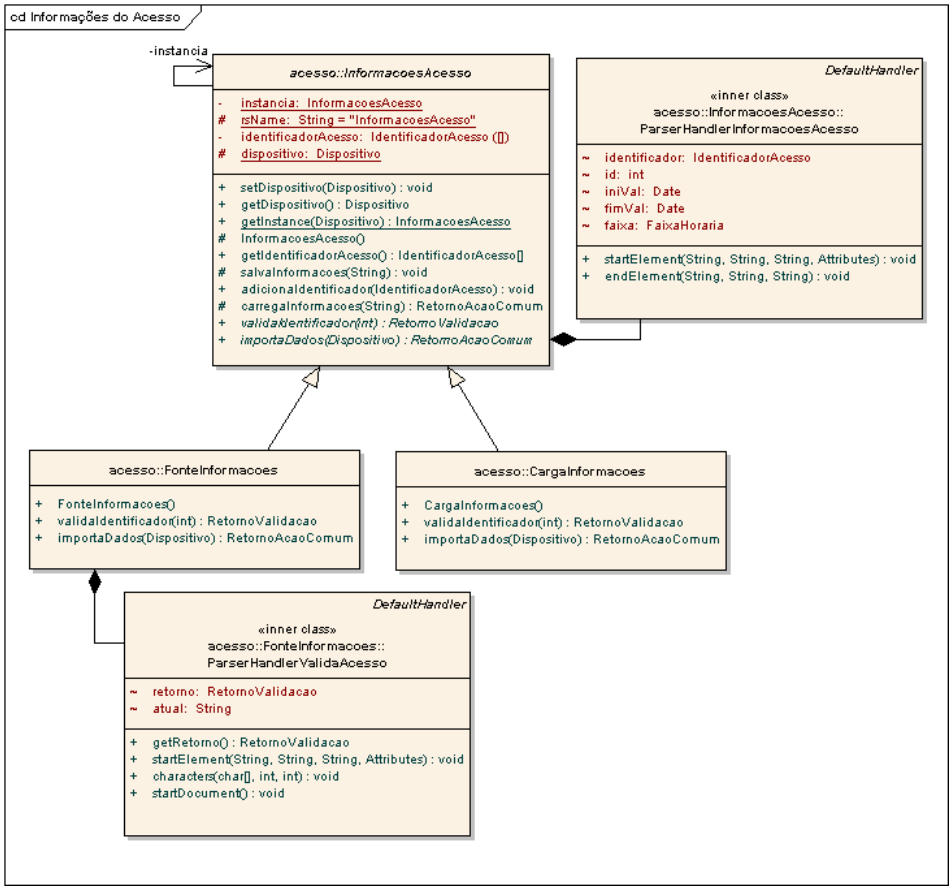


Figura 13 - Diagrama de classes do assunto “informações do acesso”

A classe “ParserHandlerInformacoesAcesso” é uma classe interna de “InformacoesAcesso” responsável pelo *parser* do XML resultante da função de importação do acesso.

As classes “FonteInformacoes” e “CargaInformacoes” são subclasses de “InformacoesAcesso” responsáveis pela implementação dos métodos abstratos que exercem as funções de validação do identificador e importação das informações do acesso. “FonteInformacoes” comunica-se com a aplicação *web* agindo de forma a utilizar o banco de dados on-line para concluir as ações implementadas. “CargaInformacoes” conclui as ações utilizando apenas as informações carregadas no dispositivo.

A classe “ParserHandlerValidaAcesso” é uma classe interna de “FonteInformacoes” responsável pelo *parser* do XML resultante da função de validação do identificador.

No assunto “registro do acesso” também foi necessário trabalhar a herança da classe principal do assunto. A Figura 14 demonstra as duas classes dessa herança e também uma classe utilizada no *parser* da informação XML.

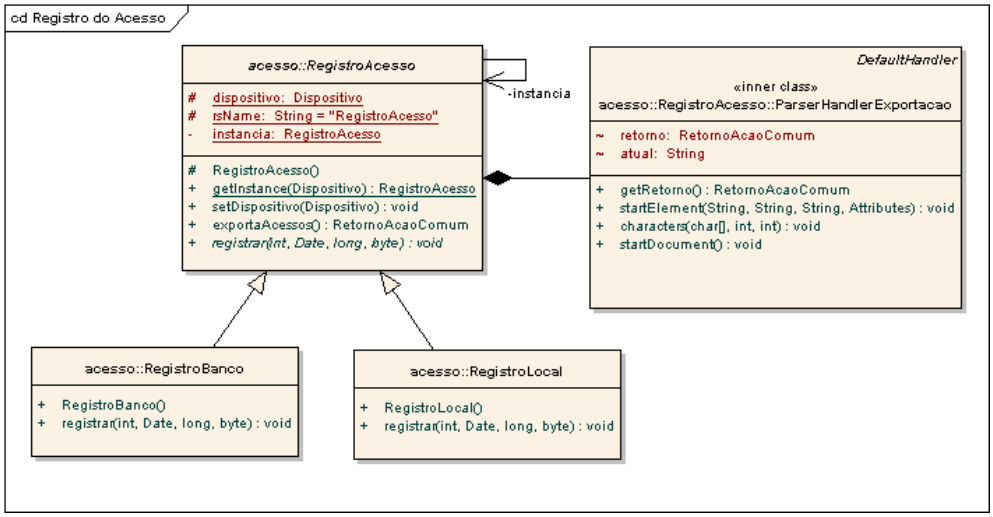


Figura 14 - Diagrama de classes do assunto “registro do acesso”

As classes “RegistroBanco” e “RegistroLocal” implementam a função abstrata de registrar o acesso da classe “RegistroAcesso”. “RegistroBanco” comunica-se com a aplicação *web* e solicita que o registro seja armazenado em banco de dados, já “RegistroLocal” armazena o registro do acesso localmente no dispositivo.

“ParserHandlerExportacao” é uma classe interna de “RegistroAcesso” responsável pelo *parser* do XML resultante da exportação dos acessos registros para o banco de dados através da aplicação *web*.

Mais algumas classes utilitárias completaram o pacote de funções do acesso, na Figura 15 elas são apresentadas.

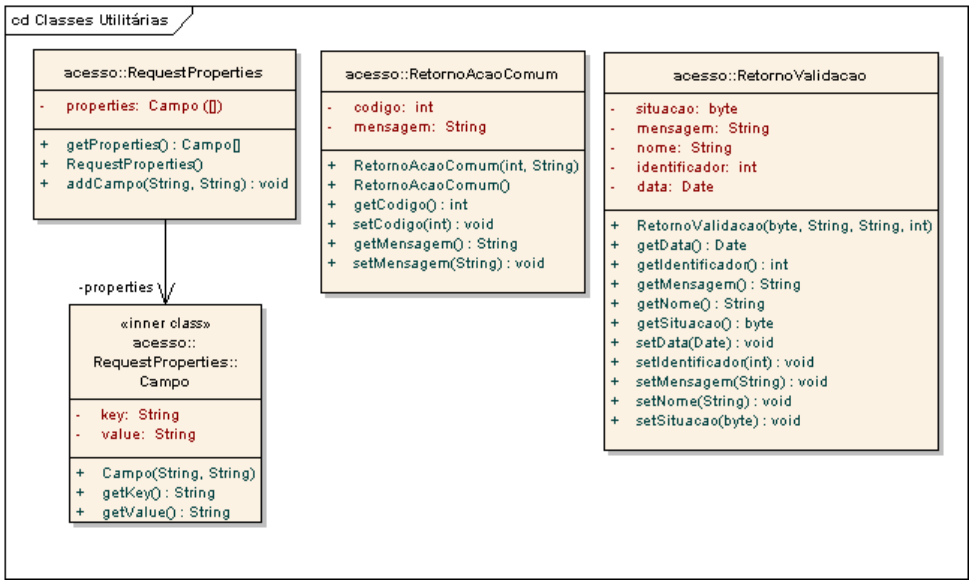


Figura 15 - Classes utilitárias do sistema

A classe “RequestProperties” é uma classe utilitária para armazenar os valores que são enviados à aplicação web. Contém uma lista de campos, controlados pela classe interna “Campo” que contém atributos para armazenar a chave e o valor da informação que se deseja enviar à aplicação *web*.

A classe “RetornoAcaoComum” é utilitária. Usada para retorno de funções que necessitam retorno de um código e uma mensagem como as funções de importação das informações do acesso e registro dos acessos validados.

Uma instância da classe “RetornoValidacao” é o resultado da função de validação do acesso. Controla as informações necessárias na apresentação das informações logo após a validação do acesso.

3.3.4 REALIZAÇÃO DE CASOS DE USO

O diagrama de seqüência definido pela UML, também conhecido por realização de caso de uso, tem a função de demonstrar como as instâncias de classe interagem entre si relatando a troca de mensagem entre elas. Este tipo de diagrama faz parte da visão dinâmica do projeto.

Por influência da modelagem ágil, procurando trazer resultados positivos e não supérfluos, aplicou-se o diagrama de seqüência apenas nos principais casos de uso:

a) importar informações do acesso;
b) verificar acesso;
c) registrar acesso;
d) exportar acessos registrados.

A Figura 16 demonstra um diagrama de seqüência com as trocas de mensagem para o caso de uso “importar informações do acesso”.

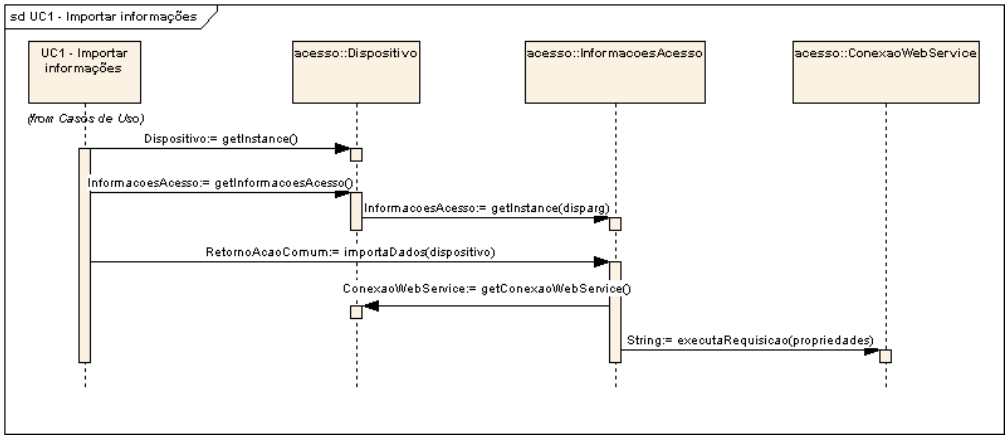


Figura 16 - Diagrama de seqüência do caso de uso “importar informações”

A Figura 17 detalha a troca de mensagens do caso de uso “verificar acesso”.

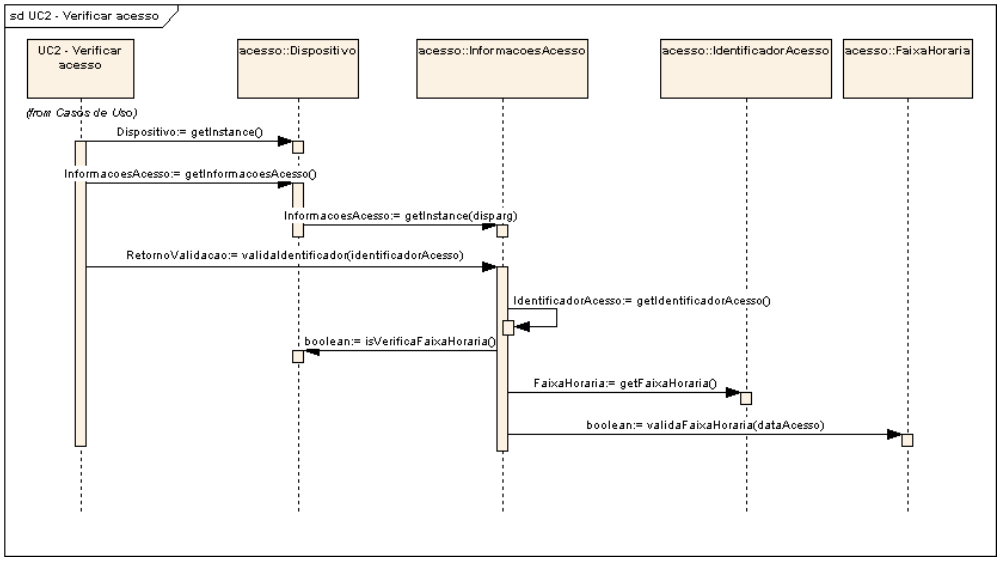


Figura 17 - Diagrama de seqüência do caso de uso “verificar acesso”

A realização dos casos de uso “registrar acesso” e “exportar acessos registrados” são demonstrados na Figura 18 e Figura 19 respectivamente.

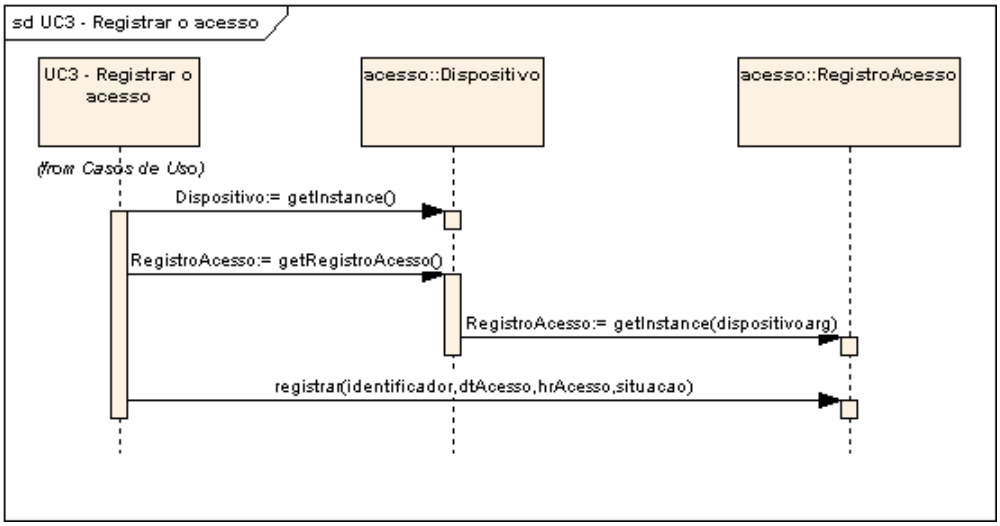


Figura 18 - Diagrama de seqüência do caso de uso “registrar acesso”

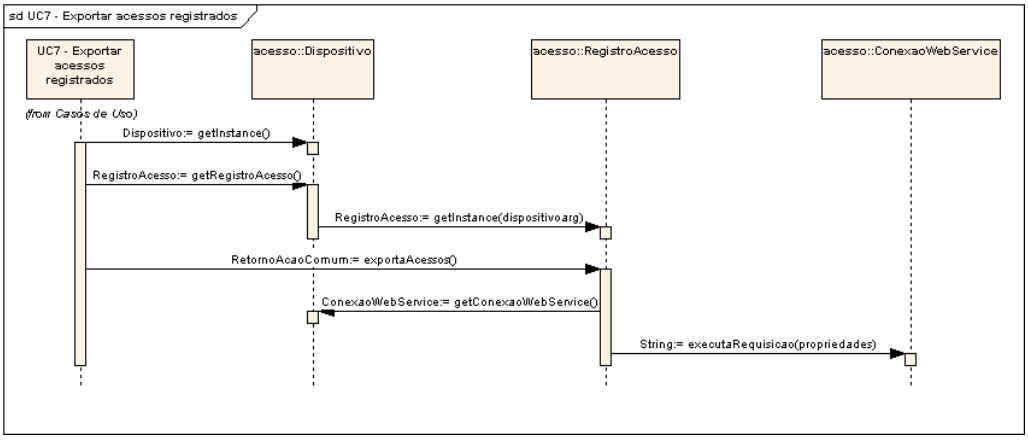


Figura 19 - Diagrama de seqüência do caso de uso ”exportar acessos registrados”

3.3.5 FERRAMENTA UTILIZADA NA ESPECIFICAÇÃO DO TRABALHO

Para a especificação do trabalho, baseada na linguagem UML, utilizou-se a ferramenta CASE Enterprise Architect que é descrita a seguir.

3.3.5.1 Enterprise Architect

Segundo Sparx Systems (2004), o Enterprise Architect é uma ferramenta *CASE* para modelagem, construção e manutenção de sistemas baseada na linguagem de modelagem unificada (UML).

Com o uso do Enterprise Architect foi possível dividir todo o projeto em três principais visões:

a) visão de caso de uso;

b) visão lógica do projeto;

c) visão dinâmica do projeto.

A visão de caso de uso foi utilizada para controlar os requisitos do sistema, separando-os em requisitos funcionais e não-funcionais. A partir dos requisitos a ferramenta possibilitou a criação dos casos de uso, destacando e classificando os atores relacionados. Para cada caso de uso, foi criada nesta visão a realização dos casos de uso. Um recurso disponível na ferramenta e bastante utilizado foi a rastreabilidade dos elementos, que permite a partir de uma classe modelada encontrar o casos de uso e também os requisitos relacionados a mesma. A Figura 20 demonstra um diagrama gerado pela ferramenta, que controla o requisito e o caso de uso que trata o requisito.

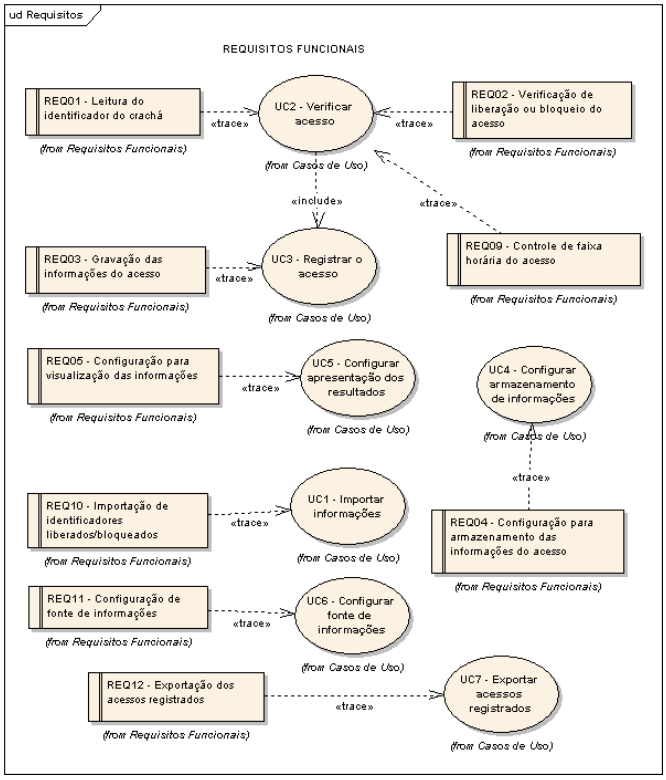


Figura 20- Rastreabilidade dos requisitos e casos de uso

A visão lógica do projeto foi utilizada para manter as classes modeladas e os diagramas de classes. A ferramenta permitiu a geração de código fonte em Java e engenharia reversa para os mesmos fontes. Com isso, foi possível aplicar um ciclo iterativo no desenvolvimento do trabalho.

A visão dinâmica teve como objetivo a utilização de diagramas de seqüência para descrever a troca de mensagens para os principais casos de uso.

Sobre o Enterprise Architect, em suma, pode-se afirmar que a ferramenta é um repositório de informações do projeto de sistema, que possibilitou ainda, a geração de relatórios, em formato HTML e *rich text*, detalhado de todas as visões e diagramas utilizados no projeto.

3.4 IMPLEMENTAÇÃO

Os assuntos seguintes descrevem as ferramentas e técnicas utilizadas para o desenvolvimento do trabalho e a operacionalidade do sistema.

3.4.1 TÉCNICAS E FERRAMENTAS UTILIZADAS

Para a fase de desenvolvimento do sistema utilizaram-se ferramentas que muito auxiliaram na decorrer do trabalho, mantendo todas as informações organizadas e consistentes.

A seguir são descritas as funcionalidades utilizadas de cada ferramenta que auxiliou no desenvolvimento do trabalho e algumas técnicas de desenvolvimento.

3.4.1.1 Plataforma Eclipse

Eclipse é um ambiente de desenvolvimento integrado e extensível que possibilitou a codificação Java do sistema. Controle de projetos, compilação, depuração e execução de progranas Java são alguns de seus recursos. Oferece também recursos de ajuda para montar projetos com todas as restrições necessárias a um projeto Java.

A característica extensível da ferramenta possibilitou a instalação de *plugins* que auxiliaram em atividades específicas dentro do desenvolvimento do trabalho. A seguir são descritos dois *plugins* utilizados no trabalho.

3.4.1.1.1 Plugin EclipseME

O plugin EclipseME que foi acoplado à ferramenta Eclipse forneceu recursos para o desenvolvimento de aplicações Java para a Micro Edition. Permitiu a criação de projetos MIDlets, com toda a estrutura necessária para a implantação da aplicação. Possibilitou configurações especiais da ferramenta Eclipse que automatizaram a compatibilidade das funções APIs e utilização da máquina virtual do J2ME.

3.4.1.1.2 Plugin Tomcat

O plugin Tomcat automatizou a criação e execução de projetos Servlets. Este trabalho usufruiu a técnica de aplicações *web* para a comunicação do dispositivo com o banco de dados. Foi a partir do plugin Tomcat que o serviço *midleware* entre dispositivo e banco de dados foi desenvolvido.

3.4.1.2 J2ME Wireless Toolkit

J2ME Wireless Tollkit é o ambiente de desenvolvimento de aplicações J2ME distribuído pela Sun Microsystems. Apesar de permitir a configuração de projetos MIDlets, este recurso foi atendido pelo Eclipse em conjunto com o plugin EclipseME. Juntamente ao *toolkit* acompanha a máquina virtual KVM e emulador de dispositivos móveis. Estes recursos foram utilizados para execução e testes dos aplicativos J2ME. As APIs do J2ME referenciadas no desenvolvimento também foram disponibilizadas pelo toolkit.

3.4.1.3 Padrões de projeto implementados

Alguns padrões de projeto foram utilizados para oferecer flexibilidade ou atender a necessidades no desenvolvimento do trabalho. Pode-se relacionar três padrões de projeto que tiverem grande influência no trabalho, padrão *Strategy*, padrão *Factory Method* e padrão *Singleton*. A seguir são detalhas as utilizações dos mesmo no projeto.

3.4.1.3.1 Emprego do padrão *Strategy*

O padrão *Strategy* foi empregado no desenvolvimento da aplicação *web* que atende as requisições do MIDlet. Havia a necessidade de responder a quatro principais requisições do MIDlet, importação de informações, exportação de registros do acesso, validação do acesso e registro do acesso. Executar e responder a requisição é a funcionalidade em comum entre essas ações. Para isso criou-se uma estratégia “EstrategiaAcessoServer”, que é uma interface e define o método “execute”. Este método “execute” foi implementado nas quatro classes que implementam a interface, “ImportacaoDados”, “ExportacaoAcessos”, “ValidacaoIdentificador” e “RegistroAcesso”. A Figura 21 demonstra o uso do padrão *Strategy* na aplicação *web*. O Apêndice A demonstra a utilização do padrão *Strategy* no trabalho.

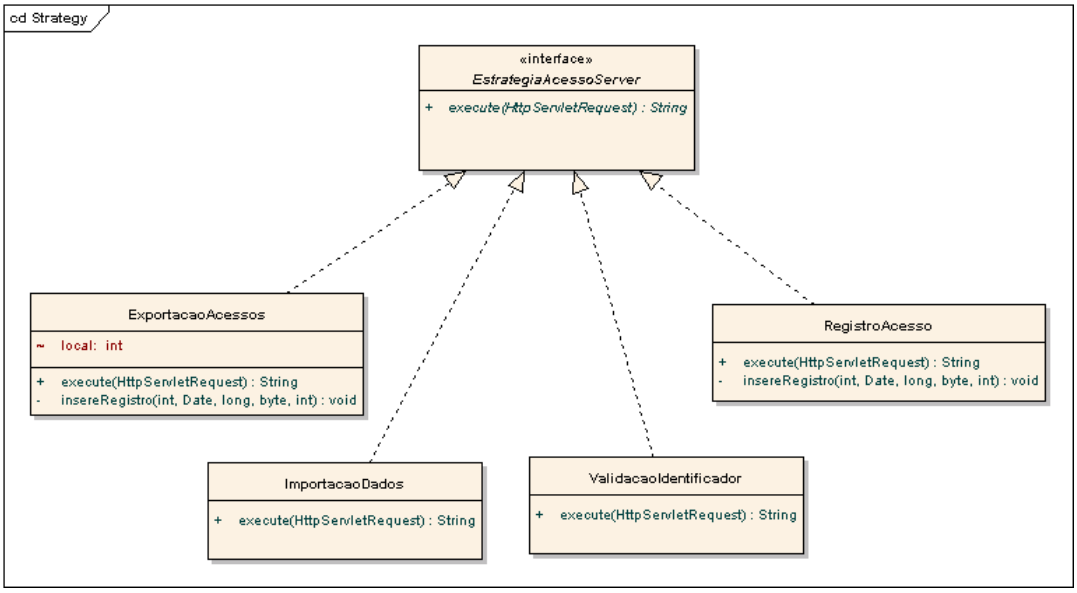


Figura 21- Uso do padrão *Strategy* na aplicação *web*

3.4.1.3.2 Emprego do padrão *Factory Method*

Para tornar flexível a validação e registro do acesso, podendo ser realizada em um banco de dados ou no dispositivo, foi utilizado o padrão de projeto *Factory Method*. Duas grandes classes foram definidas como criadoras, “InformacoesAcesso” e “RegistroAcesso”. Ambas definem métodos abstratos que são tratados por suas classes concretas. O criador “InformacoesAcesso” define os métodos “validaIdentificador” e “importaDados” como abstratos e as classes concretas “CargaInformacoes” e “FonteInformacoes” implementam da maneira que lhe cabem, trabalhando com o dispositivo local e com o banco de dados por intermédio da aplicação *web* respectivamente. O criador “RegistroAcesso” define o método registrar como abstrato e as classes “RegistroLocal” e “RegistroBanco” implementam o método. O Apêndice B demonstra a utilização do padrão *Factory Method* no trabalho.

3.4.1.3.3 Emprego do padrão *Singleton*

Há objetos definidos no sistema que devem ser únicos em toda a aplicação, não permitindo ser criado outro objeto do mesmo tipo ou classe. O padrão *Singleton* foi aplicado nas classes “Dispositivo”, “ApresentacaoInformacoes”, “ConexaoWebService”, “InformacoesAcesso” e “RegistroAcesso” para que haja apenas um único objeto instanciado no sistema. Isso é possível pela declaração privada do método construtor da classe. Há então um método público responsável por retornar a instância já criada, caso esta ainda não esteja

criada ele chama o construtor privado da classe. O Apêndice C detalha o emprego deste padrão de projeto.

3.4.1.4 Recursos especiais da J2ME utilizados

A J2ME oferece recursos especiais de conexão, armazenamento persistente de informações e algoritmos variados em seus perfis e configurações que oferecem um chamativo interessante para a utilização dos mesmos. Vale ressaltar no trabalho a utilização de alguns desses recursos.

3.4.1.4.1 Armazenamento persistente de informações

Utilizou-se no trabalho o recurso de *Record Management System* (RMS) para o armazenamento persistente das informações utilizadas pelo sistema. A idéia inicial do trabalho era usar arquivos locais no formato XML que pudessem guardar essas informações. Alguns protocolos estão disponíveis no J2ME para conexão e utilização de rede, como protocolo HTTP, *Socket*, *Datagram*, *File* e *Port*. Porém o perfil MIDP não implementa o protocolo *File*. Isso significa dizer que através do J2ME, utilizando o MIDP, não é possível atualmente usar um sistema de arquivos. Para suprir a necessidade inicial da solução foi utilizado o recurso RMS. Este recurso possibilita a criação de registros de informações por MIDlet ou até mesmo compartilhá-los entre vários MIDlets. Semelhante a uma tabela de um sistema gerenciador de banco de dados, estes registros são tratados como *arrays* de *bytes* que ficam armazenados de forma persistente no dispositivo, podendo assim o dispositivo ser desligado e as informações não serem perdidas. Todas as configurações do sistema proposta utilizam RMS. E o registro do acesso local e a carga de informações local também usam RMS.

3.4.1.4.2 Interpretação de arquivos XML

O Java oferece classes que automatizam a leitura de arquivos XML. No J2ME estes recursos são disponíveis, porém limitados. Como a troca de mensagens com o *servlet* resulta em informações do tipo XML, houve a necessidade de se utilizar estas classes para fazer o *parser* destas informações.

3.4.1.4.3 Conexão com serviço *web* através do protocolo HTTP

Um recurso interessante da J2ME para os dispositivos móveis é a comunicação através do protocolo HTTP. Este recurso foi utilizado na comunicação da aplicação do dispositivo com a aplicação *servlet* que trabalha com o banco de dados.

3.4.2 OPERACIONALIDADE DA IMPLEMENTAÇÃO

Para demonstração da operacionalidade do sistema, são apresentadas a seguir algumas funções passo a passo da utilização do sistema seguindo uma lógica tentando simular um caso real de utilização do sistema.

O sistema é carregado e uma tela inicial é mostrada ao usuário conforme a Figura 22. Há duas opções de escolha, opção “Sair” e opção “Menu”. A opção “Sair” finaliza o sistema. A opção “Menu” abre o menu geral do sistema.

Figura 22 - Tela inicial

Conforme Figura 23, o menu geral oferece as seguintes opções:

a) configurações que abre um menu com mais opções;
b) importação das informações do acesso;
c) exportação dos registros do acesso;
d) controle do acesso.

Figura 23 - Menu geral

A opção de configuração oferece um menu com mais três opções, conforme pode ser visto na Figura 24.

Figura 24 - Menu configuração

Ao selecionar a opção de configuração geral, uma tela é disposta ao cliente para confirmação do código local que representa o código de permissão do acesso correspondente ao dispositivo. É confirmada nesta tela a utilização ou não do banco de dados para consulta e registro *on-line* das informações e registros do acesso. Há também a opção de habilitar ou não a verificação da faixa horária. A Figura 25 demonstra a tela de configurações geral.

Figura 25 - Tela de configuração geral

Ao modificar as informações é necessário que elas sejam salvas através da escolha da opção “Salvar”. Uma mensagem confirmando o armazenamento é mostrada ao usuário. O armazenamento é persistente, podendo o dispositivo ser desligado e as informações não serão perdidas. Após isso pode-se escolher a opção “Sair” para sair da tela.

Ao escolher a opção “aplicação web” no menu de configurações uma tela para informar o endereço de conexão com a aplicação *web* é mostrada ao usuário conforme Figura 26.

Figura 26 - Tela configuração aplicação web

Para trabalhar de forma *on-line* com o banco de dados é necessário que se configure o meio de conexão com o banco que será realizado através de uma aplicação *web*. Após preenchido o endereço de conexão com a aplicação *web*, pode-se selecionar a opção “salvar” para registrar a configuração escolhida.

Na tela de apresentação dos resultados, selecionada através do menu configurações, são oferecidas as informações que podem ser dispostas ao usuário logo após da validação do acesso. As opções são nome, identificador do acesso, data e hora do acesso e situação do acesso. A mensagem da validação é fixa e sempre será apresentada ao usuário, por este motivo não há disponível a opção de mensagem da validação ao usuário. A Figura 27 demonstra a tela de configuração da apresentação dos resultados, que pela opção “salvar” registra no dispositivo as configurações.

Figura 27 - Tela configuração apresentação das informações

A tela de importação das informações disponível através do menu geral do sistema tem a função de carregar a lista de identificadores que têm permissão de acesso para o determinado dispositivo de acordo com seu código local. Caso esteja configurado para consulta no banco de dados, o sistema informa que a importação será realizada através da aplicação *web* e aguarda que o usuário continue a operação pela escolha da opção “importar”. Ao dar continuidade o sistema apresenta uma confirmação para uso dos recursos de conectividade do dispositivo. Ao confirmar, o dispositivo usa os recursos de rede para conectar-se à aplicação *web* e enviar uma requisição de importação das informações do acesso. A aplicação *web* responde com a lista de identificadores em formato xml, que o sistema trata, armazena localmente de forma persistente e também carrega na memória do dispositivo. A Figura 28 mostra a tela de informação da importação e a Figura 29 exemplifica a tela de confirmação para os recursos de rede do dispositivo.

Figura 28 - Tela de informação da importação

Figura 29 - Tela de confirmação do uso da rede do dispositivo

O próximo passo é a validação do acesso que pode ser realizada através da opção “controlar acesso” no menu principal do sistema. Esta tela solicita o número do identificador

do acesso da pessoa que deseja ter permissão ao local controlado. A Figura 30 mostra a tela de controle do acesso.

Figura 30 - Tela de controle do acesso

Após informar o identificador e solicitar a validação através da opção “validar”, o sistema, caso esteja configurado para consulta *on-line* no banco de dados, envia uma requisição para a aplicação *web* solicitando a validação do identificador. Nesta requisição seguem parâmetros do código do local a ser controlado e verificação ou não da faixa horária. A aplicação *web* faz as consultas necessárias no banco de dados e responde a requisição no formato xml. A partir da análise do xml o sistema efetua duas ações, registro do acesso e apresentação das informações do acesso. Para a ação de registro, não há interface com usuário, ele é realizado ou no dispositivo ou no banco de dados através de uma requisição com a aplicação *web*, isso depende da configuração geral do dispositivo. A ação de apresentação das informações do aceso corresponde a uma tela disposta ao usuário com as informações selecionadas na tela de configuração da apresentação das informações.

Algumas funcionalidades podem ter um tratamento diferenciado de acordo com algumas configurações. A configuração que mais influencia na flexibilidade do sistema é a utilização ou não do banco de dados para consulta e registro do acesso. Já foi citado

anteriormente o caso da importação e validação do acesso caso esteja sendo utilizado o banco de dados. Quando o sistema está configurado para não utilizar o banco de dados, a importação do acesso é um pouco diferenciada. Nessa situação a importação das informações é realizada apenas carregando os dados persistentes do dispositivo para a memória do dispositivo. A Figura 31 demonstra a tela de informação da importação quando não estiver configurado o banco de dados para consulta e registro do acesso.

Figura 31- Tela de informação importação local

O controle do acesso, quando da não utilização do banco de dados, necessita que as informações já estejam carregadas em memória. Por essa necessidade, é importante que antes de controlar o acesso seja realizada a importação das informações.

A tela de controle de acesso e apresentação das informações é igual para ambas configurações de uso do banco de dados. O que modifica é apenas a forma da validação, que no caso da não utilização do banco de dados, faz toda a consulta em memória do dispositivo, registra o acesso em memória persistente e depois apresenta a tela de apresentação das informações do acesso.

Os registros do acesso armazenados no dispositivo precisam em algum momento ser enviados ao banco de dados. Para isso, a opção no menu principal de exportação das

informações é a ação responsável pelo envio de registros do acesso. A tela de exportação apresenta uma mensagem informando que os dados serão enviados a aplicação *web*. Após a confirmação, os registros são enviados ao banco de dados através de uma requisição de exportação à aplicação *web*. A aplicação *web* responde a requisição com um status informando se a operação foi realizada com sucesso. A Figura 32 apresenta a tela de mensagem da exportação.

Figura 32 - Tela de exportação dos acessos registrados localmente

A validação do identificador do acesso pode resultar em diversas situações. As possíveis situações da validação do acesso são:

a) 0, representando acesso válido;
b) 1, representando identificador de acesso sem histórico válido, possível somente na verificação da aplicação *web*;
c) 2, representando identificador sem permissão de acesso ao local, possível somente na verificação da aplicação *web*;
d) 3, representando identificador inválido pela faixa horária;
e) 4, representando identificador inválido ou sem permissão ao local, possível somente na verificação local sem conexão com a aplicação *web*;
f) 98, representando erro ao validar identificador com aplicação *web*;

g) 99, representando erro na validação da aplicação *web*.

Serão apresentadas a seguir as telas das diferentes situações do acesso e também as diferentes apresentações das informações para a situação de acesso válido. A Figura 33 demonstra a situação de identificador de acesso sem histórico válido. A Figura 34 demonstra a situação de identificador sem permissão de acesso ao local. A situação de identificador inválido pela faixa horária é representada na Figura 35. A Figura 36 exemplifica a situação de identificador inválido ou sem permissão ao local.

Figura 33 - Tela validação do acesso sem histórico válido

Figura 34 - Tela validação do acesso sem permissão de acesso ao local

Figura 35 - Tela validação inválido pela faixa horária

Figura 36 - Tela validação identificador inválido ou sem permissão ao local

A Figura 37 exemplifica a situação de acesso válido com todas as informações possíveis sendo mostradas.

Figura 37 - Validação com todas as informações apresentadas

3.5 RESULTADOS E DISCUSSÃO

A etapa de testes do trabalhou tratou de avaliar as classes do sistema em separado com a utilização de testes de unidade. Após os testes de unidade, também foram realizados testes de aceitação, que tomaram como ponto principal os casos de uso do sistema. Os casos de uso foram testados e assim o sistema foi validado como atendente dos requisitos especificados.

No protótipo é possível realizar a leitura de um identificador de acesso, fazer a validação e registro do acesso ao local controlado pelo dispositivo que executa o sistema. Durante a validação é possível o controle de faixas horárias do acesso. Há a opção de carga das informações de acesso no dispositivo, ou seja, os identificadores que têm permissão de acesso válido para o dispositivo podem ser listados quando se deseja utilizar um controle *off-line* sem comunicação com a aplicação *web*. Quando os acessos são registrados localmente no dispositivo, no caso de um controle *off-line*, é possível utilizar o recurso de exportação dos acessos que se comunica com a aplicação *web* e envia os registros ao banco de dados. A validação do acesso e armazenamento dos acessos registrados são configuráveis pelo usuário gerenciador do sistema, podendo realizar um controle com ou sem comunicação com a aplicação *web* que por sua vez conversa com o banco de dados. A apresentação das

informações após validado o acesso também é configurável, possibilitando suprimir algumas informações caso não sejam importantes aos usuários do sistema.

Os padrões de projeto aplicados no trabalho foram úteis, pois auxiliaram no cumprimento de alguns objetivos, como a flexibilidade da aplicação. Ajudaram também a superar de forma elegante alguns desafios do desenvolvimento.

Alguns contratempos foram encontrados durante o desenvolvimento, detalhes que não se conhecia na proposta da solução e que serão discutidos a seguir.

3.5.1 PROBLEMAS ENCONTRADOS DURANTE O DESENVOLVIMENTO

Esperava-se que o J2ME possuísse suporte a utilização de JDBC para comunicação direta com o banco de dados. Este é possível sim no J2ME, porém apenas na configuração CDC e como o problema foca em dispositivos limitados, a configuração que teve que ser escolhida foi a CLDC que não dá suporte ao JDBC. Para suprir esta necessidade, buscou-se a utilização de API para comunicação com a aplicação *web*, onde o MIDlet transporta informações à aplicação *web* e o mesmo se encarrega de comunicar-se com o banco de dados, daí sim utilizando JDBC. Esta solução desvinculou do dispositivo um processo que poderia afetar a capacidade de processamento do mesmo.

Esperava-se que fosse possível trabalhar com um sistema de arquivos no dispositivo móvel através do J2ME. Porém, a máquina virtual do J2ME para a configuração CLDC disponível pela Sun não implementa o protocolo *File*, que é responsável pelo sistema de arquivos. A necessidade de utilizar um sistema de arquivos para armazenamento das configurações do dispositivo, informações do acesso e registros do acesso foi suprida com a utilização do recurso RMS do J2ME. Esta utilização veio a oferecer uma segurança maior das informações, tendo em vista que o sistema de arquivos seria disponível para qualquer outra aplicação, que também fosse executada no dispositivo móvel.

4 CONCLUSÕES

O protótipo especificado cumpriu os requisitos levantados, garantindo assim uma solução que permite o controle de acesso a locais físicos por meio de um dispositivo móvel. Com este trabalho, possibilitou-se a substituição de equipamentos de difícil transporte por dispositivos móveis na função de controle de acesso, oferecendo a flexibilidade da aplicação em relação ao armazenamento de informações e interface ao usuário.

Os objetivos do trabalho foram atendidos. O protótipo oferece flexibilidade de aplicação, em relação ao armazenamento de informações e interface ao usuário na apresentação das informações, adequado a diferentes situações de uso conforme configurações. Garante a portabilidade de diferentes sistemas operacionais de acordo com a tecnologia J2ME. O protótipo incrementa a automatização do controle de segurança patrimonial.

Tecnicamente, o protótipo é portável em um ambiente de computação móvel explorando os recursos de orientação a objetos utilizando padrões de projeto.

O J2ME, pela utilização do perfil MIDP e configuração CLDC, atendeu muito bem a necessidade de se desenvolver um sistema de controle de acesso voltado a dispositivos móveis auxiliando de forma decisiva na solução proposta.

A utilização do JDBC pela aplicação *web* mostrou-se uma solução confiável e flexível para comunicação com o banco de dados.

Não foi possível testar o protótipo em diferentes sistemas operacionais, como objetivo do trabalho propõe a portabilidade, por não se ter acesso a diferentes equipamentos que suportam a tecnologia. Utilizou-se apenas a máquina virtual oferecida pelo *J2ME Wireless Toolkit*.

O uso de padrões de projeto no trabalho foi válido, pois ajudou na organização e manutenibilidade do sistema além de fornecer soluções testadas e aprovadas.

A seção extensões sugere trabalhos que enriqueceriam a solução de controle de acesso para dispositivos móveis.

4.1 EXTENSÕES

Foi tratada neste trabalho apenas a identificação lógica por intermédio da digitação de um identificador de acesso. São sugestões de extensão deste trabalho a comunicação com as portas do dispositivo móvel para a realização do bloqueio físico e/ou captura do identificador do acesso através de uma leitora, seja ela de cartões ou biométrica. Alguns dispositivos oferecem comunicação através de portas seriais e nestes casos poder-se-ia estudar os recursos que J2ME oferece para comunicação com essas portas. No caso da ação de bloqueio físico, o dispositivo poderia controlar equipamentos de hardware enviando um pulso magnético para o mesmo.

Como sugestão cabe também a migração da aplicação *servlet* para um *Java Web Service* garantindo assim uma solução *web* mais robusta e segura. A utilização de *web service* oferece recursos interessantes de segurança na comunicação entre o sistema móvel e a aplicação *web*.

Analisando um sistema de controle de acesso avançado pode-se sugerir a implementação de alguns controles específicos como:

a) histórico de mensagens ao usuário;
b) programação de permissão diferenciada, onde esta permissão tem horário que não consta na faixa horária padrão da pessoa e que pode ser utilizada para um evento extraordinário em uma data específica;
c) controle de créditos de acesso, onde a pessoa tem uma faixa limite de acesso ao determinado local no determinado período.

Pode-se sugerir também a extensão do trabalho para a área de controle de presenças em aulas em instituições de ensino. O controle do acesso aqui proposto avaliaria a possibilidade do aluno/pessoa freqüentar ou não a aula, e a extensão sugerida controlaria a presença do aluno, que depois exportaria para um sistema de gestão escolar assumindo o papel da conhecida lista de chamada.

REFERÊNCIAS BIBLIOGRÁFICAS

AMBLER, Scott W.. **Modelagem ágil**: práticas eficazes para a programação extrema e o processo unificado. Porto Alegre: Bookman, 2003. 352p.

ARGONAVIS. **Argonavis Informática e Consultoria**. Poá, 2004. Disponível em: <http://www.argonavis.com.br/>. Acesso em: 01 nov. 2004.

CAMARA-E, Câmara Brasileira de Comércio Eletrônico. **Câmara Brasileira de Comércio Eletrônico**. São Paulo, 2005. Disponível em: < http://www.camara-e.net>. Acesso em: 11 jan. 2005.

GAMMA, Erich, et al. **Padrões de projeto**: soluções reutilizáveis de software orientado a objetos. Trad. Luiz A. Meirelles Salgado. Porto Alegre: Bookman, 2000. 364 p.

KNUDSEN, Jonathan. **Wireless Java**: developing with J2ME. 2.ed. New York: Apress, 2003, xviii, 364 p.

OMG, Object Management Group. **Unified Modeling Language.** [S.l.], 2004. Disponível em: <http://www.uml.org/>. Acesso em: 01 nov. 2004.

SENIOR SISTEMAS. **Senior Sistemas**. Blumenau, 2004. Disponível em: <http://www.senior.com.br/>. Acesso em: 01 nov. 2004.

SPARX SYSTEMS. **Enterprise Architect**. [S.l.], 2004. Disponível em <http://www.sparxsystems.com.au/>. Acesso em: 01 nov. 2004.

SUN JAVA. **Java technology**. [S.l.], 2004. Disponível em: <http://java.sun.com/>. Acesso em: 01 nov. 2004.

TELECOMWEB. **Os astros da mobilidade**. [S.l.], 2005. Disponível em: <http://www.telecomweb.com.br/>. Acesso em: 11 jan. 2005.

TELEMÁTICA SISTEMAS. **Telemática Sistemas Inteligentes**. São Paulo, 2004. Disponível em <http://www.tsi.com.br>. Acesso em: 01 nov. 2004.

TOPLEY, Kim. **J2ME in a nutshell** : a desktop quick reference. Cambridge: O`Reilly, c2002. xv, 450 p.

WHITE, James, HEMPHILL, David. **Java 2 Micro Edition**: Java in small things. Greenwich: Manning, 2002, 479 p.

APÊNDICE A – Implementação do padrão *Strategy* no trabalho

Padrão *Strategy* aplicado nas classes do *servlet* do trabalho.

```
//Interface que define o método principal da estratégia
public interface EstrategiaAcessoServer {
        public String execute(HttpServletRequest request);
}

public class ExportacaoAcessos implements EstrategiaAcessoServer{
        public String execute(HttpServletRequest request) {
        //IMPLEMENTAÇÃO
        }
}

public class ImportacaoDados implements EstrategiaAcessoServer {
        public String execute(HttpServletRequest request) {
        //IMPLEMENTAÇÃO
        }
}

public class RegistroAcesso implements EstrategiaAcessoServer {
        public String execute(HttpServletRequest request) {
        //IMPLEMENTAÇÃO
        }
}

public class ValidacaoIdentificador implements EstrategiaAcessoServer{
        public String execute(HttpServletRequest request) {
        //IMPLEMENTAÇÃO
        }
}
```

APÊNDICE B – Implementação do padrão *Factory Method* no trabalho

Padrão *Factory Method* aplicado na classe “InformacoesAcesso”.

```
public abstract class InformacoesAcesso {
	//método abstrato
	public abstract RetornoValidacao validaIdentificador(int identificadorAcesso);
	//método abstrato
	public abstract RetornoAcaoComum importaDados(Dispositivo dispositivo);
}

public class FonteInformacoes extends InformacoesAcesso {
	public RetornoValidacao validaIdentificador(int identificadorAcesso){
		//IMPLEMENTACAO
	}
	public RetornoAcaoComum importaDados(Dispositivo dispositivo){
		//IMPLEMENTACAO
	}
}

public class CargaInformacoes extends InformacoesAcesso {
	public RetornoValidacao validaIdentificador(int identificadorAcesso){
		//IMPLEMENTACAO
	}
	public RetornoAcaoComum importaDados(Dispositivo dispositivo){
		//IMPLEMENTACAO
	}
}
```

Factory Method aplicado na classe “RegistroAcesso”.

```
public abstract class RegistroAcesso {
	//método abstrato
	public abstract void registrar(int identificador, Date dtAcesso, long hrAcesso, byte
situacao);
}

public class RegistroBanco extends RegistroAcesso {

	public void registrar(int identificador, Date dtAcesso, long hrAcesso, byte situacao){
		//IMPLEMENTACAO
	}
}

public class RegistroLocal extends RegistroAcesso {
	public void registrar(int identificador, Date dtAcesso, long hrAcesso, byte situacao){
		//IMPLEMENTACAO
	}
}
```

APÊNDICE C – Implementação do padrão *Singleton* no trabalho

Padrão *Singleton* aplicado nas classes “Dispositivo”, “ApresentacaoInformacoes”, “ConexaoWebService”, “InformacoesAcesso” e “RegistroAcesso”.

```
public class Dispositivo {
	private static Dispositivo instancia;
	public static Dispositivo getInstance() {
		if (instancia == null) {
			instancia = new Dispositivo();
		}
		return instancia;
	}
	private Dispositivo(){
		//CRIAÇÃO
	}
}

public class ApresentacaoInformacoes {
	private static ApresentacaoInformacoes instancia;
	public static ApresentacaoInformacoes getInstance(){
		if (instancia == null) {
			instancia = new ApresentacaoInformacoes();
		}
		return instancia;
	}
	private ApresentacaoInformacoes(){
		//CRIAÇÃO
	}
}

public class ConexaoWebService {
	private static ConexaoWebService instancia;
	public static ConexaoWebService getInstance(){
		if (instancia == null) {
			instancia = new ConexaoWebService();
		}
		return instancia;
	}
	private ConexaoWebService() {
		//CRIAÇÃO
	}
}

public abstract class InformacoesAcesso {
	private static InformacoesAcesso instancia;
	public static InformacoesAcesso getInstance(Dispositivo disparg) {
		if (instancia == null){
			if (disparg.isArmazenamentoBanco()) {
```

```
                        instancia = new FonteInformacoes();
                    } else {
                        instancia = new CargaInformacoes();
                    }
                }
                if (dispositivo == null) {
                    dispositivo = disparg;
                }
                return instancia;
        }
        protected InformacoesAcesso(){
                //CRIAÇÃO
        }
}

public abstract class RegistroAcesso {
        private static RegistroAcesso instancia;
        public static RegistroAcesso getInstance(Dispositivo dispositivoarg) {
                if (instancia == null) {
                    if (dispositivoarg.isArmazenamentoBanco()) {
                        instancia = new RegistroBanco();
                    } else {
                        instancia = new RegistroLocal();
                    }
                }
                if (dispositivo == null) {
                    dispositivo = dispositivoarg;
                }
                return instancia;
        }
}
```