UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO – BACHARELADO

PROTÓTIPO DE UM DIÁRIO DE CLASSE EM DISPOSITIVOS MÓVEIS UTILIZANDO J2ME

FABIANO ROSA

BLUMENAU
2005

2005/1-14

FABIANO ROSA

PROTÓTIPO DE UM DIÁRIO DE CLASSE EM DISPOSITIVOS MÓVEIS UTILIZANDO 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 Ciências da Computação — Bacharelado.

Prof. Marcel Hugo - Orientador

BLUMENAU
2005

2005/1-14

PROTÓTIPO DE UM DIÁRIO DE CLASSE EM DISPOSITIVOS MÓVEIS UTILIZANDO J2ME

Por

FABIANO ROSA

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

Presidente:	Prof. Marcel Hugo, Mestre – Orientador, FURB
Membro:	Prof. Francisco Adell Péricas, Mestre – FURB
Membro:	Prof. Maurício Capobianco Lopes, Mestre – FURB

Blumenau, 15 de junho de 2005

Dedico este trabalho aos meus pais, aos meus amigos, aos mestres da universidade, ao meu orientador, e especialmente à minha noiva, Liliane, que me incentivou e acreditou no meu potencial.

AGRADECIMENTOS

A Deus pela oportunidade de estar concluindo mais uma etapa da minha vida.

Aos meus pais e familiares, que sempre me apoiaram e incentivaram durante toda a minha formação acadêmica.

Ao professor Maurício Capobianco Lopes, por ter colaborado com a especificação dos requisitos deste trabalho.

A professora Joyce Martins, pela solução sugerida para um dos requisitos do trabalho, na área de Compiladores.

Agradeço aos demais professores do curso de Bacharelado em Ciências da Computação da Universidade Regional de Blumenau, por todo conhecimento repassado no curso.

A minha noiva, Liliane, por ter sido compreensiva durante este trabalho, por me incentivar e sempre ter estado ao meu lado nos momentos difíceis e felizes das nossas vidas.

Ao meu orientador, Marcel Hugo, pela ajuda e dedicação para que este trabalho fosse concluído com sucesso.

Se as portas da percepção fossem abertas, o homem veria as coisas como elas realmente são, infinitas.

William Blake

RESUMO

Este trabalho apresenta a especificação e desenvolvimento de um protótipo de “Diário Escolar” para execução em dispositivos móveis. A utilização da plataforma Java aliada à orientação a objetos, a especificação J2ME para a implementação do protótipo e execução em um aparelho celular, padrões de projeto para a especificação do protótipo, e a comunicação segura utilizando o protocolo HTTPs e certificado digital com tecnologia Web Services, são os componentes usados para a realização deste protótipo.

Palavras-chave: Diário de classe. Dispositivos móveis. J2ME.

ABSTRACT

This work presents a prototype specification and development of “Classroom Register” to execute on mobile devices. To implement and execute this prototype on a cellular phone, Java platform together with object orientation and J2ME specifications, design patterns to prototype specification, the security comunication that uses the HTTPs protocol and digital certification with Web Services technology are used.

Key-Words: Register of classroom. Mobile devices. J2ME.

LISTA DE ILUSTRAÇÕES

Figura 1 – Java 2 Platform 20
Figura 2 – Dispositivos Móveis e J2ME 21
Figura 3 – Componentes J2EE 26
Figura 4 – A plataforma J2EE com os serviços disponíveis 27
Figura 5 – Conteúdo de uma mensagem SOAP 30
Quadro 1 – Exemplo de envelope SOAP para obter o nome do professor 31
Quadro 2 – Exemplo de envelope SOAP para retornar o nome do professor 31
Figura 6 – 23 padrões de projetos descritos por Gamma 32
Figura 7 – Requisitos funcionais e casos de uso relacionados 38
Figura 8 – Casos de uso do sistema 41
Quadro 3 – Caso de uso “registrar freqüência dos alunos” 42
Quadro 4 – Caso de uso “autenticar o professor” 42
Quadro 5 – Caso de uso “manter e configurar avaliações” 43
Quadro 6 – Caso de uso “registrar as avaliações dos alunos” 44
Quadro 7 – Caso de uso “consultar média final dos alunos” 44
Quadro 8 – Caso de uso “selecionar turma” 45
Quadro 9 – Caso de uso “consultar detalhes do aluno” 45
Quadro 10 – Caso de uso “consultar informações sobre a freqüência do aluno” 45
Quadro 11 – Caso de uso “manter atividades do professor” 46
Quadro 12 – Caso de uso “listar alunos associados à turma” 46
Figura 9 – Diagrama de atividades do caso de uso “registrar freqüência dos alunos” 47
Figura 10 – Diagrama de atividades do caso de uso “autenticar o professor” 48
Figura 11 – Diagrama de atividades do caso de uso “manter e configurar avaliações” 48
Figura 12 – Diagrama de atividades do caso de uso “registrar as avaliações dos alunos” 49
Figura 13 – Diagrama de atividades do caso de uso “consultar média final dos alunos” 50
Figura 14 – Diagrama de atividades do caso de uso “selecionar turma” 51
Figura 15 – Diagrama de atividades do caso de uso “consultar detalhes do aluno” 51
Figura 16 – Diagrama de atividades do caso de uso “consultar informações sobre a freqüência do aluno” 52
Figura 17 – Diagrama de atividades do caso de uso “manter atividades do professor” 52
Figura 18 – Diagrama de atividades do caso de uso “listar alunos associados à turma” 53

Figura 19 – Modelo conceitual de um sistema acadêmico 55
Figura 20 – Diagrama de classes de projeto do módulo J2ME 57
Figura 21 – Diagrama de classes de projeto do módulo do WEB 59
Figura 22 – Diagrama de seqüência do caso de uso “registrar freqüência dos alunos” 60
Figura 23 – Diagrama de seqüência do caso de uso “registrar as avaliações dos alunos” 61
Figura 24 – Diagrama de seqüência do caso de uso “selecionar turma” 61
Figura 25 – Diagrama de seqüência do caso de uso “listar alunos associados à turma” 62
Figura 26 – Artefatos e seus ambientes de execução 63
Quadro 13 – Mensagem SOAP informando o acesso negado 67
Figura 27 – Tela de login ao iniciar a aplicação 70
Figura 28 – Aplicação J2ME requisitando dados da rede 71
Figura 29 – Tela com a lista de turmas 71
Figura 30 – Tela principal com o menu 72
Figura 31 – Tela com data de aula sugerida pelo sistema 73
Figura 32 – Tela com a lista de datas de aulas 74
Figura 33 – Tela para efetuar o registro da freqüência dos alunos 75
Figura 34 – Tela com a lista das avaliações da turma 76
Figura 35 – Tela para inserir uma nova avaliação 76
Figura 36 – Tela para efetuar o registro das notas dos alunos na avaliação 77
Figura 37 – Tela com a lista de atividades do professor na aula 78
Figura 38 – Registro do histórico da atividade realizada 78
Figura 39 – Lista de atividades 79
Figura 40 – Edição da atividade 80
Figura 41 – Tela com a lista de alunos da turma 80
Figura 42 – Tela com os detalhes do aluno 81
Figura 43 – Tela listando a matrícula do aluno 82
Figura 44 – Tela exibindo as informações da freqüência do aluno 83
Figura 46 – Tela exibindo a média final e as notas das avaliações do aluno 84
Quadro 14 – Funcionalidades específicas de cada trabalho. 86

LISTA DE SIGLAS

API – *Application Programming Interface*

CDC – *Connected Device Configuration*

CLCD – *Connected Limited Device Configuration*

EJB – *Enterprise Java Beans*

FP – *Foundation Profile*

GALS – *Gerador de Analisadores Léxicos e Sintáticos*

GRASP – *General Responsibility Assignment Software Patterns*

HTTP – *Hypertext Transfer Protocol*

J2EE – *Java 2 Enterprise Edition*

J2ME – *Java 2 Micro Edition*

J2SE – *Java 2 Standart Edition*

JAD – *Java Application Descriptor*

JAR – *Java Archive*

JVM – *Java Virtual Machine*

KVM – *Kilo Virtual Machine*

MIDP – *Mobile Information Device Profile*

MVC – *Model-View-Controller*

PC – *Personal Computer*

PDA – *Personal Digital Assistant*

PP – *Personal Profile*

RMS – *Record Management System*

SMTP – *Simple Mail Transfer Protocol*

SOAP – *Simple Object Acess Protocol*

UML – *Unified Modeling Language*

XML – *eXtensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO 14
1.1 OBJETIVOS DO TRABALHO 15
1.2 ESTRUTURA DO TRABALHO 16
2 FUNDAMENTAÇÃO TEÓRICA 17
2.1 DIÁRIO DE CLASSE 17
2.2 DISPOSITIVOS MÓVEIS 18
2.3 PLATAFORMA JAVA 2 19
2.4 J2ME 20
2.4.1 Configuração CLDC 22
2.4.2 Perfil MIDP 23
2.5 J2EE 24
2.5.1 EJB 27
2.5.1.1 EJBs de sessão 28
2.6 WEB SERVICES 29
2.6.1 *Simple Object Acess Protocol* (SOAP) 29
2.7 PADRÕES DE PROJETO 31
2.7.1 Padrão de projeto *Model-View-Controller* (MVC) 33
2.7.2 Padrão de projeto *Facade* 33
2.7.3 Padrão de projeto *Singleton* 33
2.8 TRABALHOS CORRELATOS 34
3 DESENVOLVIMENTO DO TRABALHO 36
3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO 37
3.2 REQUISITOS NÃO FUNCIONAIS 38
3.3 ESPECIFICAÇÃO 39
3.3.1 Casos de uso do problema 40
3.3.2 Descrição dos casos de uso 41
3.3.3 Diagrama de atividades do problema 46
3.3.4 Diagrama de classes 53
3.3.4.1 Diagrama de classes do modelo conceitual 54
3.3.4.2 Diagrama de classes de projeto 56
3.3.5 Realizações de casos de uso 59

3.3.6 Diagrama de Implantação 62
3.3.7 Ferramenta utilizada na especificação do trabalho 63
3.4 IMPLEMENTAÇÃO 64
3.4.1 Técnicas e ferramentas utilizadas 65
3.4.1.1 Plataforma Eclipse 65
3.4.1.1.1 Plugin EclipseME 65
3.4.1.1.2 Plugin JBoss-IDE 66
3.4.1.2 J2ME Wireless Toolkit 66
3.4.1.3 Bliblioteca kSOAP2 e kXML 66
3.4.1.4 Segurança na comunicação com o Web Services 67
3.4.1.5 Cálculo da expressão matemática da fórmula da média final 67
3.4.1.6 Padrões de projeto implementados 68
3.4.1.6.1 Emprego do padrão MVC 68
3.4.1.6.2 Emprego do padrão *Facade* e *Singleton* 69
3.4.2 Operacionalidade da implementação 69
3.5 RESULTADOS E DISCUSSÃO 85
4 CONCLUSÕES 87
4.1 EXTENSÕES 88
REFERÊNCIAS BIBLIOGRÁFICAS 90
APÊNDICE A – Utilização do XDoclet para geração de artefatos para o EJB e o Web Service 92
APÊNDICE B – Implementação de um método utilizando a biblioteca kSOAP2 93
APÊNDICE C – Implementação do método que utiliza HTTPs e autenticação Basic 94
APÊNDICE D – Especificação da entrada para o gerador léxico e sintático GALS 96
APÊNDICE E – Implementação do padrão *Singleton* e *Facade* no trabalho 97

1 INTRODUÇÃO

Devido à crescente expansão do mercado de telefonia móvel, atualmente com mais de um bilhão de clientes e crescendo a cada dia, onde o número de telefones celulares já supera o número de telefones fixos, cada vez mais a telefonia celular está tornando-se acessível à população, deixando de ser um acessório para ser uma ferramenta de auxílio no dia-a-dia das pessoas. Atualmente no Brasil, de acordo com Rydlewski (2004, p. 101), 32 milhões de brasileiros têm telefone celular, 10 milhões de consumidores pretendem trocar o aparelho por um mais moderno e 23 milhões de pessoas querem comprar seu primeiro celular.

Na mesma velocidade, vem aumentando o avanço da tecnologia *wireless* nos celulares, onde no início da telefonia celular, o consumidor tinha como recurso principal a capacidade de efetuar ligações e ter uma agenda telefônica disponível. Hoje já é possível manipular arquivos de mídia, tirar fotos e enviar a outros celulares ou para endereços eletrônicos, ouvir rádio AM/FM e sons em formato MP3, ter acesso à Internet, e principalmente, executar jogos e aplicações. Estes vêm abrindo novos nichos de mercado para as operadoras de telefonia e para as empresas de desenvolvimento e se tornando o centro das atenções dos fabricantes de dispositivos móveis e dos desenvolvedores de aplicações e jogos.

Segundo Almeida (2004, p. 21), a tecnologia Java fornece o *Java2 Micro Edition* (J2ME) para o desenvolvimento de aplicações em dispositivos móveis, desde telefones celulares, *pagers*, *Personal Digital Assistant* (PDA), até *smartfones* e *set-top boxes* de TVs. Para se ter uma idéia do avanço desta tecnologia, em 2002 existiam pouco mais de 50 milhões de telefones celulares com Java; em 2003, somente a fábrica da Nokia entregou mais de 100 milhões de celulares com Java, e, de acordo com as estimativas, por volta de 2007, praticamente 100% dos celulares estarão habilitados a executarem jogos e aplicações construídas em Java. Para os desenvolvedores, Java traz diversas vantagens, como por

exemplo, a portabilidade, na qual a aplicação pode ser executada em celulares de diferentes marcas e modelos, além da própria linguagem Java, que a cada dia se torna mais popular no desenvolvimento de softwares.

Visto o acima descrito, este trabalho propõe a construção do protótipo de um aplicativo para celular, utilizando os recursos provenientes da tecnologia J2ME para a aplicação que será executada no celular, para que um professor de uma determinada disciplina possa gerenciar e monitorar a freqüência e as notas de alunos e da tecnologia *Java 2 Enterprise Edition* (J2EE) para prover recursos de encapsulamento de regras de negócio e a troca de dados padrão, entre o servidor de aplicações e o celular, e dos *Design Patterns*, que oferecem soluções e padrões de projeto prontos e testados para serem aplicados neste trabalho.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é construir um protótipo para gerenciar e monitorar a freqüência e as notas de alunos de disciplinas ministradas por um determinado professor, onde o protótipo será executado em um aparelho celular que atenda a especificação Java (J2ME).

Os objetivos específicos do trabalho são:

a) utilizar os recursos da tecnologia J2ME e verificar suas limitações quando aplicada a telefones celulares;
b) executar a aplicação no telefone celular interagindo com um servidor de aplicações;
c) efetuar o sincronismo entre os dados registrados no celular e os dados residentes em um servidor.

1.2 ESTRUTURA DO TRABALHO

Este trabalho está dividido em quatro capítulos.

No primeiro capítulo, encontra-se uma introdução e os objetivos a serem alcançados com o desenvolvimento deste trabalho. No segundo capítulo é apresentada a fundamentação teórica para os itens este trabalho, contextualizando o conceito de Diário de Classe, a tecnologia Java, a utilização de dispositivos móveis e a aplicação do Java neste ambiente, os recursos utilizados do J2EE e a utilização de Web Services, e os padrões de projeto aplicados no desenvolvimento do protótipo. O terceiro capítulo aborda o desenvolvimento do trabalho, apresentando os requisitos, especificação e artefatos gerados durante a especificação do projeto, ferramentas utilizadas na especificação, desenvolvimento e execução do protótipo, resultados e problemas encontrados durante o desenvolvimento. E finalmente no quarto capítulo trata das considerações finais sobre o trabalho e sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

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

a) diário de classe;

b) dispositivos móveis;

c) tecnologia Java;

d) J2ME;

e) J2EE;

f) Web Services;

g) padrões de projeto;

h) trabalhos correlatos.

2.1 DIÁRIO DE CLASSE

Segundo FURB (2001), o Diário de Classe é um documento oficial da instituição, que tem por finalidade registrar o histórico dos acadêmicos em sala de aula. Trata-se de um documento importante nos processos de reconhecimento e autorização dos cursos. Neste documento são registrados, principalmente, a freqüência do acadêmico durante o semestre letivo, as notas das avaliações aplicadas e observações sobre o rendimento escolar do acadêmico. No que diz respeito ao registro da freqüência, o mínimo exigido é de 75% da carga horária total da disciplina, para fins de aprovação, ficando a cargo do professor, o controle da presença do acadêmico. Em relação ao processo de avaliação, a verificação da aprendizagem é de responsabilidade do professor da disciplina, e está baseada em todas a atividades curriculares, como provas orais, escritas e práticas, exercícios de aplicação,

pesquisas, trabalhos práticos, saídas a campo, projetos, estágios e outros procedimentos definidos pelo Colegiado do Curso. A composição da média final deverá ser de, no mínimo, 3 (três) notas parciais, com exceção das disciplinas de estágio supervisionado e outras, como as atividades de conclusão de curso. Para a aprovação do acadêmico na disciplina, a média final deverá ser igual ou superior a 6,0 (seis), onde a mesma deve ter uma casa decimal. Para avaliar o rendimento escolar do acadêmico, a nota será expressa numa escala de 0,0 (zero) a 10,0 (dez), com uma casa decimal, sendo que seu registro será feito no Diário de Classe, a ser entregue ao final de cada semestre letivo.

2.2 DISPOSITIVOS MÓVEIS

Segundo Laudon e Laudon (1999, p. 160), novas formas de comunicações estão sendo proporcionadas por redes móveis de dados, usando dispositivos móveis, como telefones celulares e PDA - um computador manual composto de uma caneta, com recursos internos e organizacionais de comunicações. Muito mais do que assistentes pessoais ou agendas eletrônicas, os dispositivos móveis passaram a ser computadores que podem ser facilmente levados a qualquer lugar, criados para atender profissionais e pessoas em movimento que necessitam de rapidez, facilidade e segurança no acesso a informações corporativas e pessoais. Alguns dispositivos móveis encontrados no mercado hoje são:

a) telefones celulares;
b) *pagers*;
c) PDAs;
d) dispositivos embarcados;
e) computadores de bordo automotivo;
f) *handhelds*.

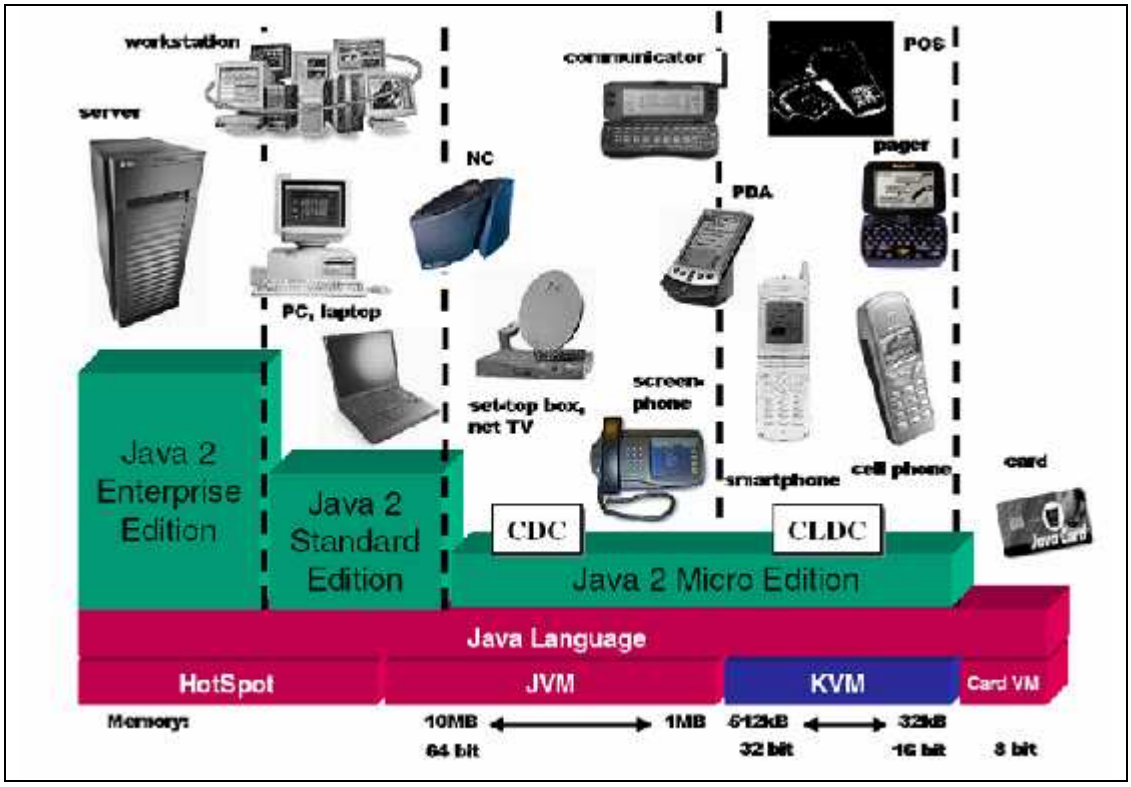
Além disso, as grandes inovações trazidas pela tecnologia *wireless* fizeram com que a indústria deste setor tenha tido um crescimento explosivo nos últimos anos, tornando-se uma das mais eficientes e rápidas áreas tecnológicas do mundo, permitindo que as pessoas comuniquem-se de forma barata e fácil sem ficarem presas aos seus telefones ou computadores de mesa.

2.3 PLATAFORMA JAVA 2

De acordo com Sun Microsystems (2004a), Java 2 é a nova versão da plataforma Java para desenvolvimento. Percebendo que um só produto não conseguiria abranger todas as necessidades do mercado, a Sun projetou a Java 2 em três edições: *Java 2 Platform, Standard Edition* (J2SE); *Java 2 Platform, Enterprise Edition* (J2EE); e *Java 2 Platform, Micro Edition* (J2ME). Cada edição define uma tecnologia e um conjunto de ferramentas com propósitos diferentes:

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.

As três edições da Java 2 Platform e os dispositivos que cada tecnologia atende estão representadas na Figura 1.



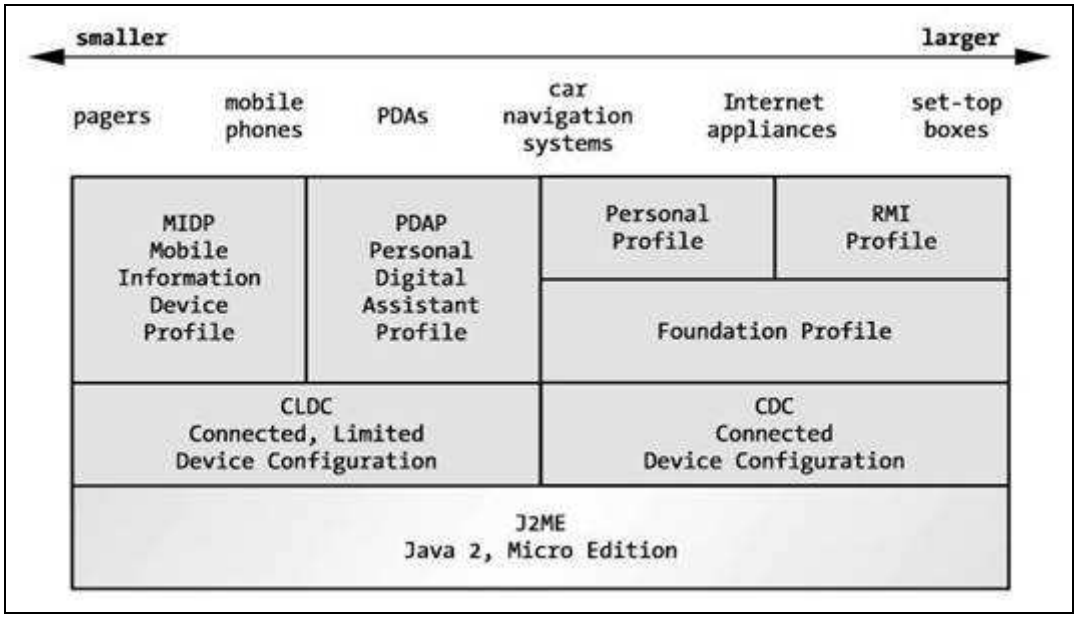
Fonte: Corbera (2003)

Figura 1 – Java 2 Platform

2.4 J2ME

De acordo com Sun Microsystems (2004b), J2ME define um padrão para inserir e executar aplicações em Java dentro de dispositivos que não possuem o poder de processamento de ambientes J2SE completos, tais como PDAs, celulares, TVs, eletrodomésticos entre outros.

Conforme Fonseca (2002, p. 07), a Máquina Virtual Java do J2ME, normalmente chamada de Kilobyte Virtual Machine (KVM), a qual é implementada de acordo com a especificação da comunidade Java, não é a mesma utilizada pelas versões J2EE e J2SE, sendo quase sempre um conjunto menor dessas versões. Por isso, códigos não podem ser portados diretamente de uma versão maior de Java para o J2ME. A relação entre os dispositivos móveis e J2ME está representada na Figura 2.



Fonte: Knudsen (2003, p. 2)

Figura 2 – Dispositivos Móveis e J2ME

Conforme a Sun Microsystems (2004b), atualmente a plataforma J2ME consiste de um conjunto de configurações (*configurations*) e perfis (*profiles*).

Uma configuração serve como a base de execução e desenvolvimento das aplicações J2ME, onde uma mesma configuração atende vários dispositivos diferentes, não possuindo detalhes específicos do dispositivo. A *Application Programming Interface* (API) de uma configuração pode ter subconjuntos da API de J2SE. No momento existem duas Configurações: *Connected Device Configuration* (CDC) e *Connected Limited Device Configuration* (CLDC). A configuração CDC possui um conjunto de APIs que suportam equipamentos fixos de porte médio, tal como televisores. A configuração CLDC é um conjunto de APIs destinadas a aparelhos cujo poder de processamento, *display* e memória são limitados.

Um perfil funciona como o complemento da configuração, em que esta define o básico de funcionalidades e o perfil atende detalhes específicos do dispositivo. Alguns dos perfis existentes são: *Mobile Information Device Profile* (MIDP), *Foundation Profile*, PDA *Profile*, *PersonalJava*, etc. Cada perfil é destinado a uma categoria específica de dispositivos, e

consiste num conjunto mínimo de bibliotecas de classes que determinado aparelho deve suportar.

2.4.1 Configuração CLDC

Conforme Schmitt Junior (2004), 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 Perfil 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 (SCHMITT JUNIOR, 2004).

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 recurso 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.

Conforme a Sun Microsystems (2004c), existem duas versões do perfil MIDP: a versão 1.0, onde estão disponíveis APIs para o ciclo de vida da aplicação, conectividade de rede via HTTP, interface com o usuário e persistência de dados; e a versão 2.0, que traz APIs para a utilização do protocolo HTTPs, utilização de recursos multimídia do dispositivo, desenvolvimento de jogos, novos componentes de formulário e outras melhorias e novidades para o perfil MIDP.

2.5 J2EE

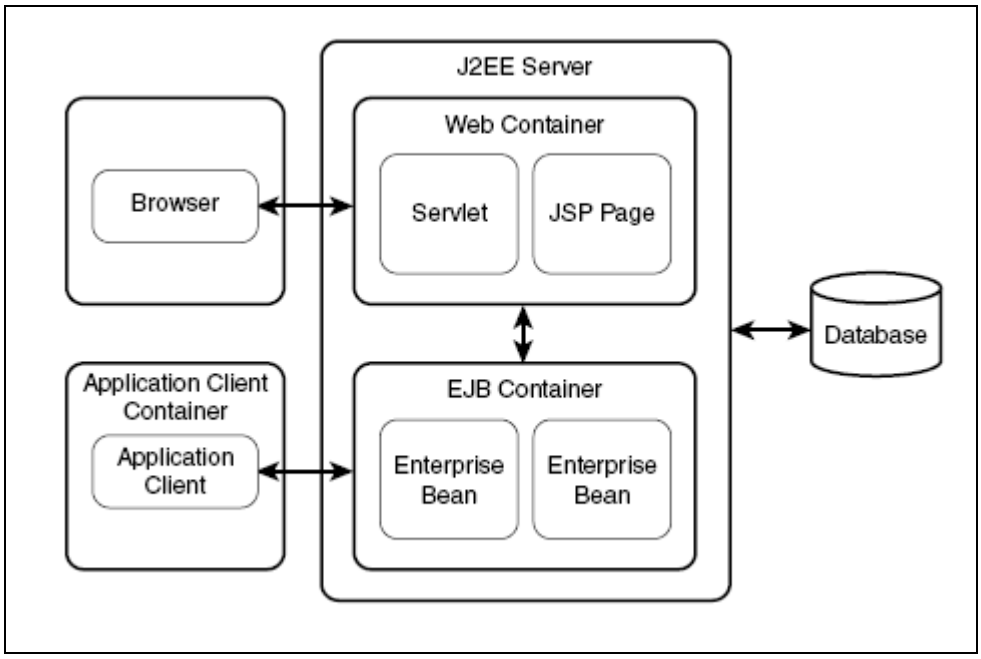
Bond (2003) descreve a especificação J2EE como um padrão dinâmico para a produção de aplicativos corporativos seguros, escaláveis e altamente disponíveis. A especificação define quais serviços devem ser fornecidos pelos servidores que suportam J2EE. Estes serviços forneceram contêineres J2EE nos quais os componentes J2EE serão executados. Os contêineres forneceram um conjunto definido de serviços para os componentes. A especificação J2EE fornece uma definição por meio da qual os fornecedores

corporativos podem produzir servidores de aplicações comerciais, como o IBM *WebSphere*, ORACLE OC4J e BEA *WebLogic Server*, ou soluções *open source*, como o servidor de aplicação JBoss, estes no qual, implementam a especificação J2EE em seus produtos, assim seguindo um padrão estabelecido pela comunidade Java.

A plataforma J2EE fornece um ambiente comum para a construção de aplicativos corporativos seguros, escaláveis e independentes de plataforma. Muitas empresas estão distribuindo bens e serviços para os clientes através da Internet, usando tais servidores baseados na especificação J2EE. Os requisitos deste ambiente exigem padrões abertos para construir aplicativos, os quais estão listados a seguir:

a) J2SE, ambiente independente de plataforma;
b) componentes que apresentam interfaces com o usuário com base na Web;
c) componentes para encapsular processos corporativos;
d) acesso a dados em repositórios de dados corporativos;
e) conectividade com outras fontes de dados e sistemas legados;
f) suporte para *eXtensible Markup Language* (XML), a linguagem padrão usada em operações de comércio eletrônico.

O J2EE especifica que um servidor de aplicações compatível com a especificação deve fornecer um conjunto definido de contêineres para abrigar os componentes J2EE. Os contêineres fornecem um ambiente em tempo de execução para os componentes. Desse modo, a J2SE está disponível em cada contêiner. As APIs no J2EE também estão disponíveis, para fornecer comunicação entre componentes, persistência, descoberta de serviços e outros. Existem dois tipos de componentes distribuídos, gerenciados e executados em um servidor de aplicações J2EE: os componentes Web e os componentes *Enterprise Java Beans* (EJB). Estes componentes e seus relacionamentos entre os diferentes contêineres e componentes, estão representados na Figura 3.



Fonte: Bond (2003, p. 14)

Figura 3 – Componentes J2EE

Os contêineres J2EE devem fornecer a cada tipo de componente um conjunto de serviços definido, onde todo servidor compatível com J2EE deve suportá-los, e a especificação fornecer APIs (Figura 4) que implementam os serviços.

a) conectividade;

b) serviços de diretório;

c) acesso a dados e persistência;

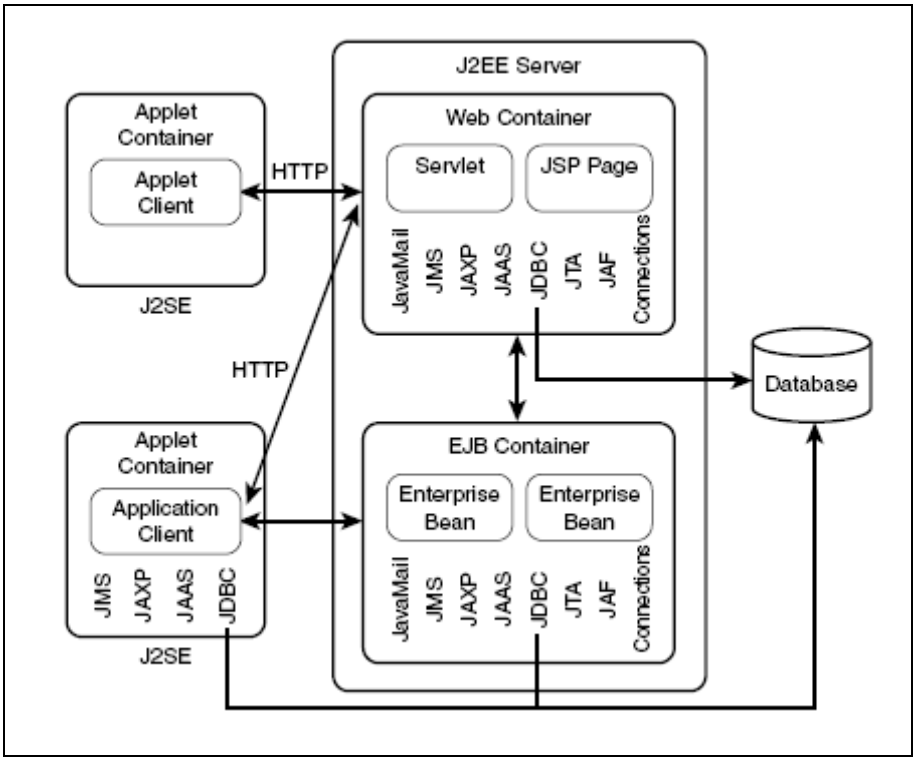
d) conectividade legada;

e) segurança;

f) suporte para XML;

g) transações;

h) troca de mensagens e *e-mail*.



Fonte: Bond (2003, p. 16)

Figura 4 – A plataforma J2EE com os serviços disponíveis

2.5.1 EJB

Conforme Bond (2003), em uma aplicação J2EE típica, os *Enterprise Java Beans* (EJBs) contem a lógica do negócio do aplicativo e dados corporativos ativos. Embora seja possível usar objetos Java padrão para controlar a lógica do negócio e dados corporativos, usar EJBs resolve muitos dos problemas que poderiam ser encontrados ao utilizar objetos Java simples, como escalabilidade, gerenciamento de ciclo de vida e gerenciamento de estados.

Um EJB é basicamente um componente gerenciado que é criado, controlado e destruído pelo contêiner J2EE em que está sendo executado. Esse controle permite que o contêiner gerencie o número de EJBs existentes e os recursos que estão usando, como memória e conexões de banco de dados. Cada contêiner manterá um conjunto (*pool*) de instâncias de EJBs, que já estão prontas e atribuídas a uma aplicação cliente, e quando a

aplicação não precisar mais do EJB, a sua instância voltará para o conjunto de instâncias e todos os recursos alocados pelo EJB serão liberados.

Em um projeto de aplicação, normalmente são identificados dois tipos de funcionalidades: a manipulação de dados e o fluxo do processo do negócio. Para esses diferentes tipos de funcionalidades do sistema, existem diferentes tipos de EJB: os EJBs de sessão e EJBs dirigidos por mensagens, para mapear o fluxo do processo do negócio, e os EJBs de entidade, para mapear e manipular dados de um repositório de dados.

2.5.1.1 EJBs de sessão

Os EJBs de sessão são componentes importantes dentro da plataforma J2EE, pois eles permitem que a funcionalidade do negócio da aplicação seja especificada e desenvolvida independentemente da camada lógica de apresentação. Diferentes aplicações cliente podem utilizar os mesmos EJBs de sessão de um negócio especificado, como uma aplicação Web e uma aplicação para dispositivos móveis. Existem dois tipos de EJBs de sessão: os EJBs de sessão com estados e os EJBs de sessão sem estados.

Nos EJBs de sessão com estados, o estado de um processo de negócio, pode ser mantido através de métodos de negócio requisitados ao EJB em diferentes momentos, como por exemplo, uma compra em uma loja virtual na Web, onde a escolha dos produtos, o cadastro do cliente e a finalização do pedido são feitos com etapas em momentos distintos. Já o EJB de sessão sem estados não possui a características de manter estados, assim se diferenciando na sua utilização, pois tem como objetivo fornecer métodos de negócios independentes para o processo do cliente.

2.6 WEB SERVICES

Gumz (2004) resume o *Web Service* como um padrão de computação distribuída, na qual deve existir a criação, publicação, localização e acesso por sistemas remotos. O *Web Service* pode ser visto como um serviço, publicado na forma de um componente de software independente de implementação e plataforma, onde suas interfaces públicas e regras são definidas e descritas usando o XML. O *Web Service* não precisa necessariamente estar disponível somente no ambiente WEB, pois o mesmo pode ser utilizado em qualquer rede de trabalho, Internet ou Intranet, onde geralmente é utilizado o protocolo *Hypertext Transfer Protocol* (HTTP) para efetuar a comunicação entre o cliente e o *Web Service*. Para se obter a integração e a portabilidade entre as aplicações de uma organização ou entre parceiros de negócio, é utilizado a tecnologia XML, componente chave do *Web Service*, que possibilita que ambas as partes, cliente e provedor do serviço, troquem mensagens com informações, mesmo estando em ambientes e sistemas diferentes.

2.6.1 *Simple Object Acess Protocol* (SOAP)

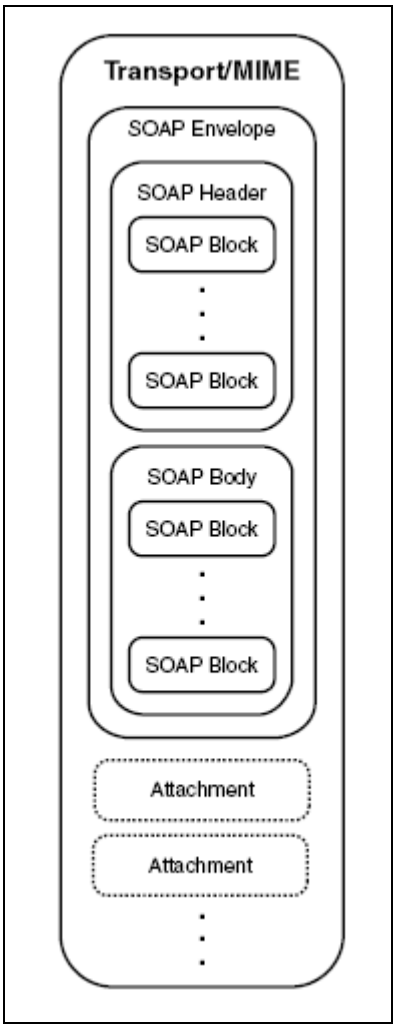
Para efetuar a troca de mensagens entre o cliente e o *Web Service*, é utilizado o protocolo SOAP, onde representado como um envelope onde a mensagem é representada e estruturada utilizando o XML.

Conforme D'ippolito (2004), o SOAP é formado por 3 partes principais (Figura 5):

a) SOAP *Envelope*: forma o elemento raiz do documento e encapsula os dois próximos elementos;
b) SOAP *Header*: é uma parte opcional do SOAP que carrega informações necessárias que não estão presentes na assinatura do método, como por exemplo,

informação sobre chave pública de criptografia, informação de sessão e informação e autenticação;

c) SOAP *Body*: o elemento mais importante, que contém um ou mais métodos e seus parâmetros armazenados no XML.



Fonte: Bond (2003, p. 857).

Figura 5 – Conteúdo de uma mensagem SOAP

Um exemplo de envelope SOAP em uma requisição e uma resposta pode ser visto nos quadros 1 e 2.

```
<soapenv:Envelope xmlns:i="http://www.w3.org/1999/XMLSchema-instance"
   xmlns:d="http://www.w3.org/1999/XMLSchema"
   xmlns:c="http://schemas.xmlsoap.org/soap/encoding/"
   xmlns:v="http://schemas.xmlsoap.org/soap/envelope/">
   <soapenv:Header />
   <soapenv:Body>
      <n0:getNomeProfessor id="o0" c:root="1" xmlns:n0="urn:soap-call">
         <codigoVinculo i:type="d:string">1001</codigoVinculo>
      </n0:getNomeProfessor>
   </soapenv:Body>
</soapenv:Envelope>
```

Quadro 1 – Exemplo de envelope SOAP para obter o nome do professor

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:xsd="http://www.w3.org/1999/XMLSchema"
   xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance">
 <soapenv:Body soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <ns1:getNomeProfessorResponse xmlns:ns1="urn:soap-call">
   <ns1:getNomeProfessorResponse xsi:type="ns2:string"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:ns2="http://www.w3.org/2001/XMLSchema">Fabiano Rosa
      </ns1:getNomeProfessorResponse>
  </ns1:getNomeProfessorResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

Quadro 2 – Exemplo de envelope SOAP para retornar o nome do professor

2.7 PADRÕES DE PROJETO

Segundo Gamma (2000), os padrões de projeto tornam mais fácil reutilizar projetos e arquiteturas bem-sucedidas, expressando técnicas testadas e aprovadas, assim facilitando o desenvolvimento de novos sistemas. Os padrões de projeto ajudam a escolher alternativas de projeto que tornam um sistema reutilizável e podem melhorar a documentação e a manutenção de sistemas ao fornecer uma especificação explícita de interações de classes e objetos e o seu objetivo subjacente.

Larman (2004) aborda a utilização de um conjunto de padrões conhecido como *General Responsibility Assignment Software Patterns* (GRASP), onde este descreve padrões com princípios fundamentais de projeto baseado em objetos e atribuição de responsabilidade. Os principais padrões GRASP são:

a) especialista na informação (Information Expert);

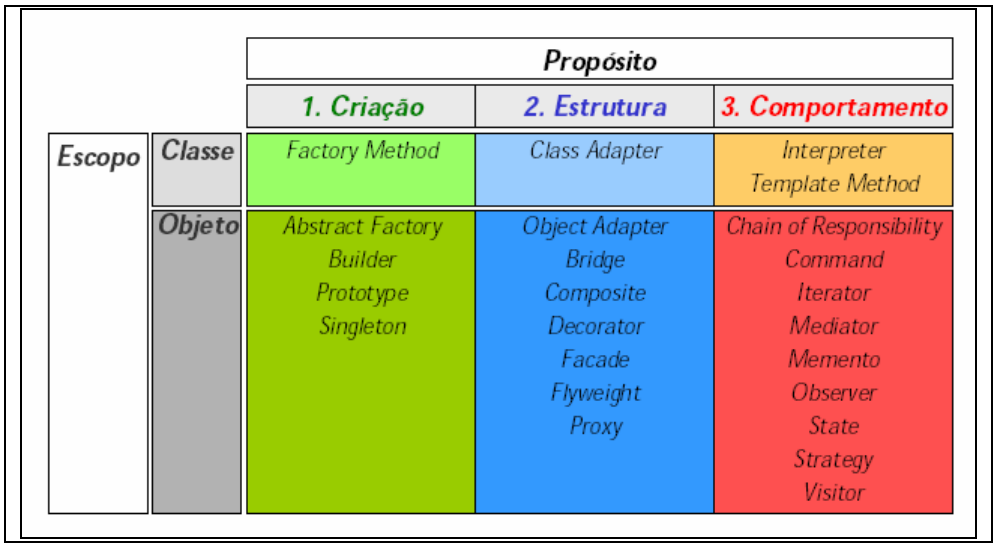
b) criador (Creator);

c) coesão alta (High Cohesion);

d) aclopamento fraco (Low Coupling);

e) controlador (Controller).

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 6.



Fonte: Argonavis (2004).

Figura 6 – 23 padrões de projetos descritos por Gamma

Cada padrão de projeto é caracterizado pelos seguintes elementos:

a) nome do padrão,

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 *Model-View-Controller*, *Facade* e *Singleton*, pois foram estes os padrões aplicados no projeto que resultaram em soluções mais eficientes para concluir alguns objetivos do trabalho.

2.7.1 Padrão de projeto *Model-View-Controller* (MVC)

Gamma (2000) define que o padrão MVC tem como objetivo separar os elementos de um projeto em 3 objetos: *Model* (Modelo), *View* (Visão) e *Controller* (Controlador). O Modelo é o objeto da aplicação, a Visão é a apresentação da interface com o usuário e o Controlador define a maneira como a interface do usuário reage às entradas do mesmo. Com esta separação proposta pelo MVC, o projeto ganha uma maior flexibilidade e reutilização, deixando clara a responsabilidade de cada camada envolvida na aplicação.

2.7.2 Padrão de projeto *Facade*

Segundo Gamma (2000), a finalidade do padrão *Facade* é oferecer uma interface para um conjunto de interfaces de um subsistema. Este padrão define uma interface de nível mais elevado, o qual torna mais fácil a utilização das interfaces do subsistema. O padrão *Facade* permite que os objetos individuais cuidem de uma única tarefa, deixando que a fachada se encarregue de divulgar as suas operações.

2.7.3 Padrão de projeto *Singleton*

Este padrão tem como objetivo garantir que uma classe só tenha uma única instância

no sistema e um único ponto de acesso à sua instância (GAMMA, 2000). Para isto, a classe deve ter um atributo do tipo da sua própria classe privado e estático, um construtor privado, e um método público estático que retorna a instância do objeto.

2.8 TRABALHOS CORRELATOS

Em Schaefer (2004) é relatado o desenvolvimento de um protótipo de uma aplicação em J2ME, utilizando o telefone celular para coletar dados de atividades externas de uma empresa e enviar para outro sistema para que possam ser feitas consultas e análises dos dados para dar apoio à tomada de decisões. Foi utilizado o perfil MIDP 2.0 do J2ME para construir as interfaces gráficas e emuladores de dispositivos móveis para a execução do protótipo. Os dados são coletados no celular pela aplicação J2ME e enviados a um servidor de e-mail via o protocolo *Simple Mail Transfer Protocol* (SMTP), onde são armazenadas para posteriores consultas e análises.

Schmitt Junior (2004) desenvolveu um protótipo de *front end* de Controle de Acesso, utilizando J2ME, onde tem como objetivo principal automatizar o controle de segurança patrimonial, oferecendo flexibilidade na aplicação, em relação ao modo de armazenamento das informações e a interface com o usuário, se beneficiando com os recursos de um dispositivo móvel, baseado na tecnologia J2ME. No desenvolvimento do trabalho, são explorados recursos avançados do J2ME, como por exemplo, o *Record Management System* (RMS), para o armazenamento das informações no dispositivo e a troca de informações com base no protocolo HTTP, utilizando o formato XML para a troca de dados com um servidor.

Outro trabalho de desenvolvimento de software para dispositivos móveis é relatado em Depiné (2002), o qual mostrou a implementação de um aplicativo em Java que possibilita efetuar os cálculos de tempo e deslocamento necessários para realização de um rally de

regularidade. O software foi construído utilizando a linguagem Java (J2ME) para equipamentos portáteis, e instalado em um dispositivo móvel (celular). Para a implementação do protótipo foram utilizados a configuração CLDC 1.0 e o perfil MIDP 1.0 do J2ME, e o emulador para o celular i85s da Motorola, para a execução da aplicação. Foi necessária a criação de uma classe em Java para emular cálculos matemáticos com números de ponto-flutuante, utilizando somente aritmética inteira, devido a limitações do J2ME, pois o mesmo não suporta números de ponto-flutuante.

3 DESENVOLVIMENTO DO TRABALHO

O protótipo de diário de classe resultante deste trabalho é um software para permitir o gerenciamento e o monitoramento da freqüência e das notas dos alunos e a visualização de informações adicionais a respeito das turmas, alunos e atividades em aula de um professor. O software é formado por quatro partes principais, descritas a seguir:

a) um módulo que é executado em um dispositivo celular, utilizando a tecnologia J2ME, onde o mesmo se comunica com um Web Service em um servidor de aplicações, via HTTPs e SOAP;
b) um módulo WEB, desenvolvido em J2EE, onde são mantidas informações sobre avaliações e atividades do professor;
c) um componente de negócio, implementado como um EJB de sessão para ser implementadas as regras de negócio do sistema, onde o mesmo é disponibilizado na forma de Web Service para possibilitar a comunicação com o módulo em J2ME;
d) uma fachada que é utilizada pelo EJB para ser acoplado a um sistema legado, com o objetivo de prover os dados acadêmicos sobre as turmas, alunos, aulas e outras informações provindas da instituição de ensino.

Para o desenvolvimento do trabalho foram necessários um levantamento e análise dos requisitos que definem as características que o sistema deve ter. Os requisitos deste projeto foram levantados e definidos em uma reunião realizada na FURB, com a participação do professor e chefe da Divisão de Registro Acadêmico, Maurício Capobianco Lopes e do professor Marcel Hugo, onde foram identificadas necessidades do professor em relação à aplicação do Diário de Classe. Uma especificação se fez necessária para expressar por intermédio de diagramas como os requisitos foram tratados no trabalho. A análise dos

requisitos e suas especificações são tratadas a seguir.

3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

Os requisitos funcionais especificam as principais características da camada de negócio do sistema. Estes requisitos descrevem os serviços que o sistema deve oferecer, como deve reagir a interação com o usuário e o comportamento do sistema em determinadas situações. Apenas os principais requisitos funcionais definidos na reunião realizada com os professores foram contemplados neste trabalho, onde estes são:

a) RF01 - registrar a freqüência dos alunos em cada aula;
b) RF02 - registrar as notas das avaliações aplicadas aos alunos;
c) RF03 - consultar o percentual e total de ausência dos alunos nas aulas ministradas pelo professor;
d) RF04 - consultar as informações detalhadas do aluno, como código de pessoa, vínculo, curso, currículo, nome e fotografia e as turmas da sua matrícula;
e) RF05 - cadastrar uma configuração para as avaliações, com o objetivo de calcular a média final das avaliações dos alunos;
f) RF06 - autenticar o professor, através de um login e senha previamente cadastrados no sistema, de forma segura e confiável;
g) RF07 - listar as turmas associadas ao professor;
h) RF08 - manter as atividades que serão realizadas em cada aula, podendo ser registrada uma observação para cada atividade.

Na Figura 7, estão representados os requisitos funcionais relacionados com os casos de uso identificados na fase de especificação do protótipo.

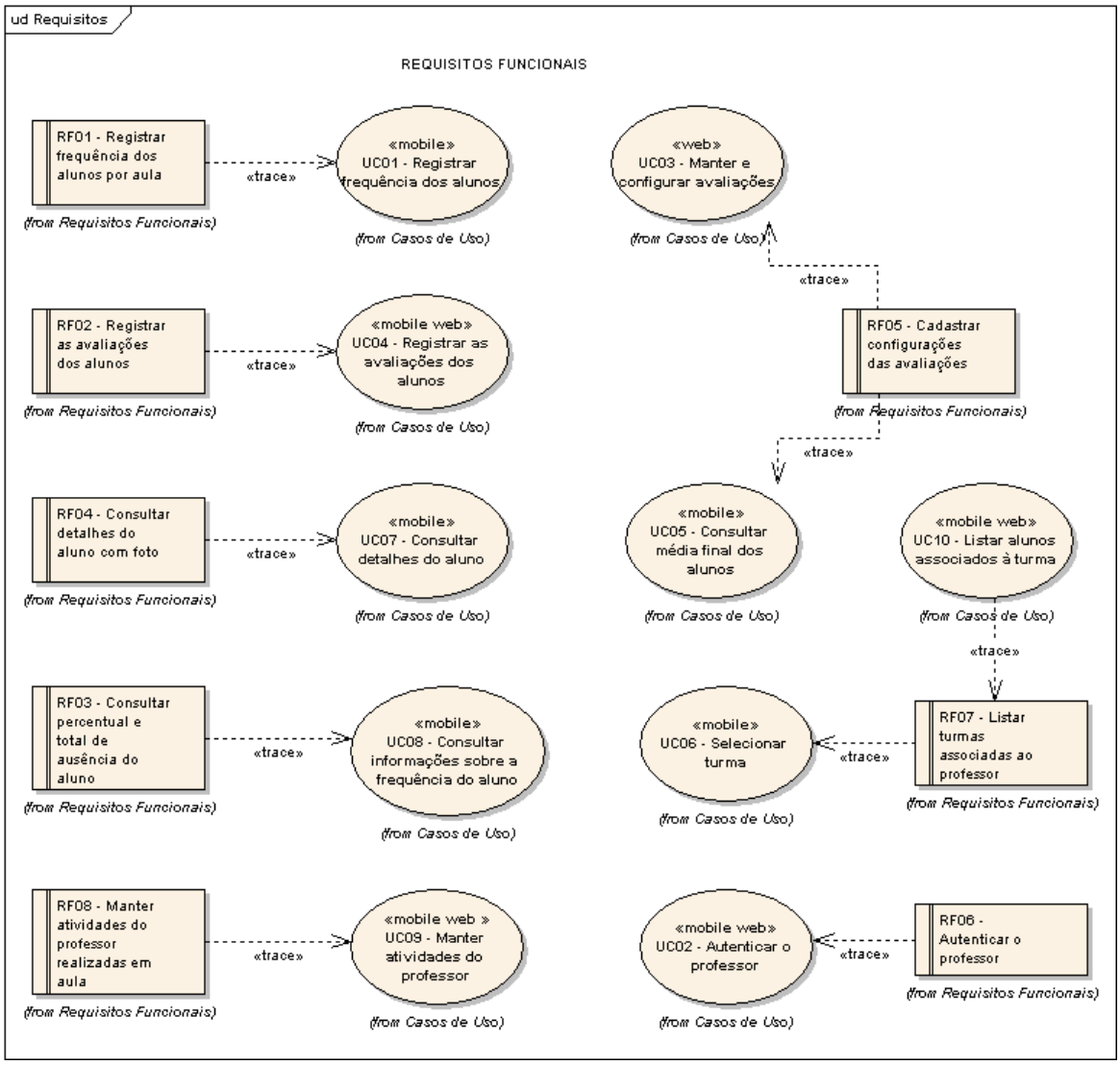


Figura 7 – Requisitos funcionais e casos de uso relacionados

3.2 REQUISITOS NÃO FUNCIONAIS

Requisitos de segurança, protocolo de comunicação e interface com o usuário são tratados como não-funcionais, mas apesar de não terem influência no negócio da aplicação, estão diretamente relacionados com a implementação do protótipo, assim sendo, são listados os requisitos não funcionais do sistema:

a) meio de comunicação seguro entre o cliente e o servidor utilizando o protocolo HTTPs;

b) utilização do formato XML e SOAP para efetuar a troca de dados entre o servidor e o cliente;

c) interface com o usuário através de dispositivo móvel e browser Web.

3.3 ESPECIFICAÇÃO

De acordo com Booch, Rumbaugh e Jacobson (2000), a *Unified Modeling Language* (UML), usada na especificação do projeto, é uma linguagem gráfica para visualização, especificação, construção e documentação de artefatos de sistemas complexos de software. A UML proporciona uma forma padrão para a preparação de planos de arquitetura de projetos de sistemas, incluindo aspectos conceituais tais como processos de negócios e funções do sistema, além de itens concretos como as classes escritas em determinada linguagem de programação, esquemas de banco de dados e componentes de software reutilizáveis. São definidos pela UML doze tipos de diagramas divididos em três categorias:

a) diagramas estruturais;

b) diagramas comportamentais;

c) diagramas de gerenciamento de modelos.

Para a especificação deste trabalho, de acordo com a necessidade do projeto, foram utilizados apenas os principais diagramas da UML, e uma metodologia de modelagem proposta por Larman (2004), seguindo apenas os princípios fundamentais para tornar produtivo o processo de modelagem.

A seguir serão abordados os casos de uso do problema, diagrama de atividades dos casos de uso, diagramas de classe e diagramas de realização de casos de uso. Por fim, é abordada a ferramenta de modelagem utilizada na especificação do projeto.

3.3.1 Casos de uso do problema

Os casos de uso demonstram a interação do usuário com o sistema, destacando as ações que serão realizadas. Para a especificação do protótipo, foram definidos dez casos de uso:

a) registrar freqüência dos alunos;

b) autenticar o professor;

c) manter e configurar avaliações;

d) registrar as avaliações dos alunos;

e) consultar média final dos alunos;

f) selecionar turma;

g) consultar detalhes do aluno;

h) consultar informações sobre a freqüência do aluno;

i) manter atividades do professor;

j) listar alunos associados à turma.

A seguir os casos de uso definidos acima, são representados no diagrama de caso de uso, conforme a figura 8.

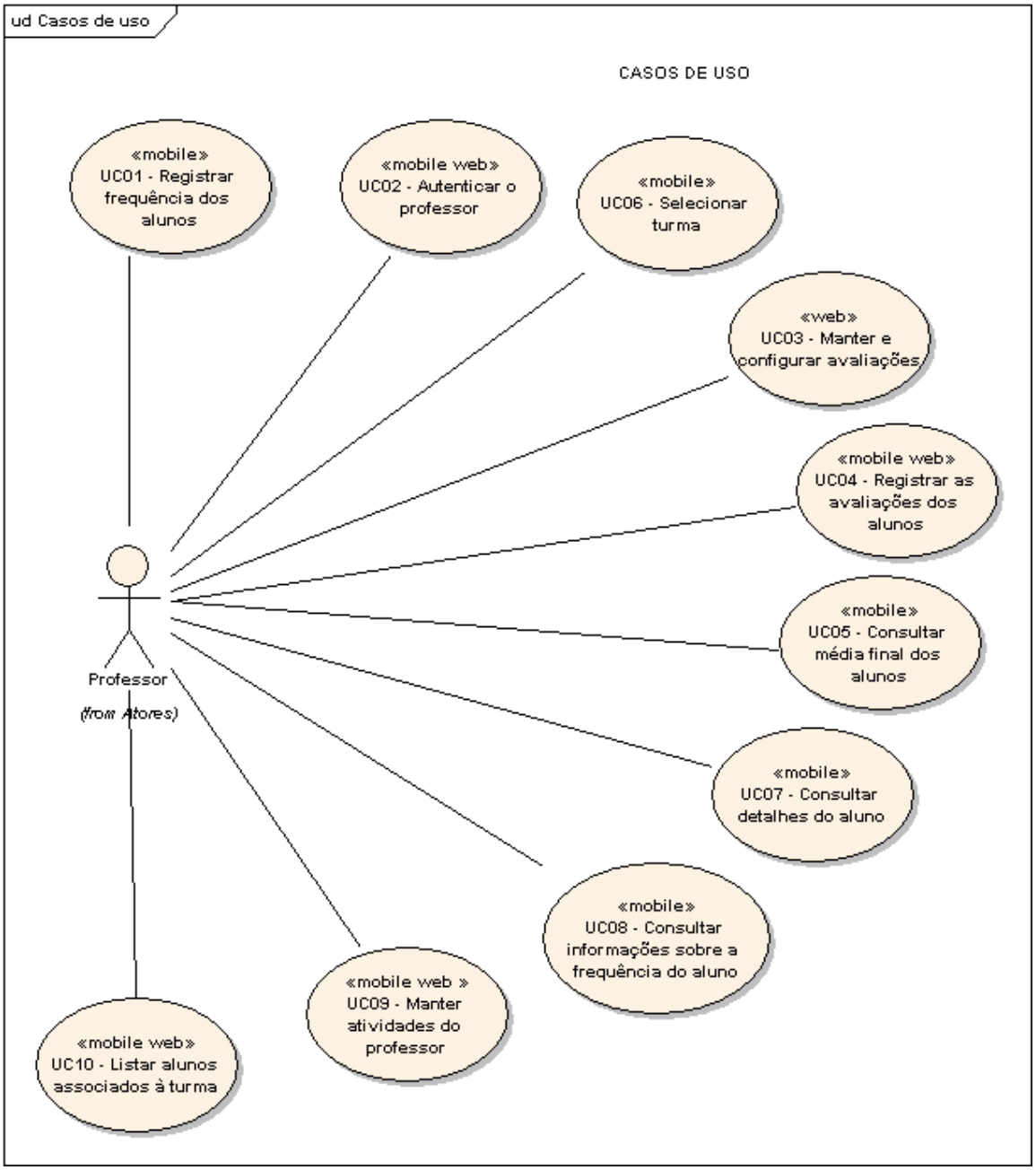


Figura 8 – Casos de uso do sistema

3.3.2 Descrição dos casos de uso

A seguir serão detalhados os casos de uso especificados na figura 8. As descrições para os casos de uso podem ser vistas nos quadros de 3 a 12.

Sumário	O professor seleciona um data de aula da lista de datas, e registra a freqüência (“C” ou “F”) de cada aluno.
Ator	Professor
Pré-condições	- O professor deve ter selecionado uma turma para registrar a freqüência do aluno; - O sistema deve ter no mínimo uma aula associada à turma.
Fluxo principal	- O sistema sugere uma data de aula válida ao professor; - O professor escolhe a data sugerida ou seleciona outra data a partir da lista de datas; - O professor registra a freqüência do aluno na(s) aula(s) desejada(s); - O sistema efetua o registro das freqüências do(s) aluno(s); - O sistema exibe uma mensagem ao professor informando que a freqüência foi registrada com sucesso.
Fluxo de exceção	Não há.
Pós-condições	A freqüência do aluno na respectiva aula foi registrada.

Quadro 3 – Caso de uso “registrar freqüência dos alunos”

Sumário	O professor informa o seu login e password na tela de login do sistema. Se o seu login falhar, será mostrada uma mensagem de erro para o mesmo, e se o login estiver correto, será exibida uma tela com as turmas relacionadas ao professor para a seleção da turma.
Ator	Professor
Pré-condições	O usuário deve estar previamente cadastrado e liberado no sistema.
Fluxo principal	- O professor informa o seu login e password; - O sistema valida os dados informados; - O sistema verifica se o login é válido; - O sistema lista as turmas associadas ao professor.
Fluxo de exceção	Se o login for inválido, é exibida uma mensagem ao professor.
Pós-condições	É associada uma sessão do sistema ao login do professor e exibida uma lista de turmas associadas ao mesmo.

Quadro 4 – Caso de uso “autenticar o professor”

Sumário	O professor cadastra as avaliações que serão aplicadas à turma e configura uma fórmula para o cálculo da média final.
Ator	Professor
Pré-condições	O professor deve ter selecionado uma turma para efetuar a manutenção e configuração das avaliações.
Fluxo principal	- O professor seleciona uma turma da lista de turmas associadas ao professor; - Se já houver alguma avaliação e fórmula cadastrada, é exibida uma lista das avaliações; - O professor realiza a manutenção das avaliações; - É exibida uma lista das avaliações; - O professor cadastra a fórmula para o cálculo da média final; - O sistema faz uma validação da fórmula antes de registrar no sistema; - As avaliações e a fórmula são registradas no sistema; - É exibida uma mensagem de sucesso ao professor.
Fluxo alternativo: Inclusão de uma nova avaliação	- O professor informa um código único para a avaliação, , onde este será usado para configurar a fórmula da média final; - O professor informa a descrição da avaliação; - O professor informa alguma observação para a avaliação; - O professor informa a data prevista da realização da avaliação; - O sistema efetua o registro da nova avaliação; - O sistema exibe uma mensagem de sucesso ao professor; - O sistema exibe a lista de avaliações.
Fluxo alternativo: Alteração de uma avaliação	- O professor seleciona uma avaliação da lista de avaliações; - O professor informa a descrição da avaliação; - O professor informa alguma observação para a avaliação; - O professor informa a data prevista da realização da avaliação; - O sistema efetua o registro da alteração da avaliação; - O sistema exibe uma mensagem de sucesso ao professor.
Fluxo alternativo: Exclusão de uma avaliação	- O professor seleciona uma avaliação da lista de avaliações; - O professor exclui a avaliação; - O sistema efetua o registro das exclusões; - O sistema exibe uma mensagem de sucesso ao professor; - O sistema exibe a lista de avaliações.
Fluxo de exceção	- Se já existir uma avaliação cadastrada com o código informado, é exibida uma mensagem ao professor; - Se a fórmula cadastrada for inválida, é exibida uma mensagem ao professor.
Pós-condições	As avaliações serão criadas, alteradas ou excluídas e será exibidas uma lista com as avaliações e fórmula para o cálculo da média final configurada e validada.

Quadro 5 – Caso de uso “manter e configurar avaliações”

Sumário	O professor seleciona uma avaliação da lista de avaliações, e para cada aluno, vai registrar a nota obtida por cada aluno.
Ator	Professor
Pré-condições	- O professor deve ter selecionado uma turma para registrar as avaliações do aluno; - O sistema deve ter no mínimo uma avaliação associada à turma.
Fluxo principal	- O professor seleciona uma turma da lista de turmas associadas ao professor; - O sistema exibe uma lista de avaliações associadas à turma; - O professor seleciona uma avaliação da lista de avaliações; - O sistema exibe a lista de alunos da turma; - O professor seleciona o aluno desejado e registra sua nota na avaliação anteriormente selecionada; - O sistema registra as notas das avaliações; - O sistema exibe uma mensagem de sucesso ao professor.
Fluxo de exceção	Não há.
Pós-condições	O sistema registra a nota do aluno na avaliação selecionada.

Quadro 6 – Caso de uso “registrar as avaliações dos alunos”

Sumário	O professor seleciona um aluno da lista associado à turma, e é calculada a sua média final de acordo com a configuração cadastrada para as avaliações.
Ator	Professor
Pré-condições	- O professor deve ter selecionado uma turma; - Aluno selecionado pelo professor.
Fluxo principal	- O professor seleciona um aluno; - O sistema verifica se existe pelo menos uma avaliação cadastrada para a turma do aluno; - O sistema verifica se existe alguma nota lançada para o aluno em alguma avaliação; - O sistema verifica se existe uma configuração cadastrada para o cálculo da média; - O sistema verifica se a configuração é válida em relação às avaliações; - O sistema efetua o cálculo da média das avaliações do aluno; - O sistema apresenta ao professor as avaliações do aluno, suas notas em cada avaliação e a média final.
Fluxo de exceção	- Se a turma não possui avaliações cadastradas, é exibido uma mensagem ao professor; - Se não existir avaliações com notas lançadas para o aluno, é exibido uma mensagem ao professor; - Se não existir configuração cadastrada para as avaliações, é exibido uma mensagem ao professor.
Pós-condições	Lista das avaliações com suas notas e a média final do aluno.

Quadro 7 – Caso de uso “consultar média final dos alunos”

Sumário	O sistema apresenta uma lista de turmas associadas ao código do professor, onde cada turma tem um identificador único, para o professor selecionar.
Ator	Professor
Pré-condições	- O professor deve no mínimo ter uma turma associada ao seu login; - O professor deverá ter efetuado um login válido no sistema.
Fluxo principal	- O sistema verifica se existem turmas associadas ao código do professor que efetuou o login no sistema; - O sistema apresenta ao professor uma lista com o código e descrição das turmas associadas ao professor; - O professor seleciona a turma desejada.
Fluxo de exceção	Se nenhuma turma estiver associada ao professor, é exibida uma mensagem ao professor.
Pós-condições	Será exibida uma lista das turmas associadas ao professor.

Quadro 8 – Caso de uso “selecionar turma”

Sumário	O professor seleciona um aluno a partir de uma lista de alunos e são exibidos seus dados, como código de pessoa, vínculo, curso, currículo, nome e fotografia, e as turmas de sua matrícula.
Ator	Professor
Pré-condições	O professor deve ter selecionado uma turma para exibir os detalhes do aluno.

Fluxo principal	- O professor seleciona um aluno da lista de uma determinada turma;
	- O sistema retorna os dados do aluno selecionado;
	- É exibido ao professor:
	  - código de pessoa
	  - vínculo
	  - curso
	  - currículo
	  - nome
	  - fotografia
	  - turmas da matrícula

Fluxo de exceção	Não há.
Pós-condições	É apresentado ao professor os detalhes do aluno selecionado.

Quadro 9 – Caso de uso “consultar detalhes do aluno”

Sumário	O professor seleciona um aluno a partir de uma lista de alunos e é exibida a quantidade de faltas e percentual de ausência do aluno.
Ator	Professor
Pré-condições	O professor deve ter selecionado uma turma.

Fluxo principal	- O professor seleciona um aluno da lista de uma determinada turma;
	- O sistema retorna os dados do aluno selecionado;
	- É exibido ao professor:
	  - quantidade de faltas
	  - percentual de ausência

Fluxo de exceção	Não há.
Pós-condições	É apresentada ao professor a quantidade de faltas e o percentual de ausência do aluno.

Quadro 10 – Caso de uso “consultar informações sobre a freqüência do aluno”

Sumário	O professor mantém suas atividades realizadas em aula, informando o status, descrição e observação.
Ator	Professor
Pré-condições	O professor deve ter selecionado uma turma.
Fluxo principal	- O sistema sugere uma data de aula válida ao professor; - O professor escolhe a data sugerida ou seleciona outra data a partir da lista de datas; - O professor realiza uma manutenção nas atividades; - As atividades são registradas no sistema; - É exibida uma mensagem de sucesso ao professor.
Fluxo alternativo: Inclusão da atividade	- O professor informa uma descrição da atividade; - O professor informa alguma observação para a atividade; - O sistema efetua o registro da nova atividade; - O sistema exibe uma mensagem de sucesso ao professor; - O sistema exibe a lista de atividades.
Fluxo alternativo: Alteração da atividade	- O professor seleciona uma atividade da lista de atividades; - O professor informa uma descrição da atividade; - O professor informa alguma observação para a atividade; - O sistema efetua a alteração da atividade; - O sistema exibe uma mensagem de sucesso ao professor.
Fluxo alternativo: Exclusão da atividade	- O professor seleciona uma atividade da lista de atividades; - O professor exclui a atividade; - O sistema efetua o registro das exclusões; - O sistema exibe uma mensagem de sucesso ao professor; - O sistema exibe a lista de atividades.
Fluxo de exceção	Não há.
Pós-condições	Atividades do professor realizadas em aula cadastradas.

Quadro 11 – Caso de uso “manter atividades do professor”

Sumário	O sistema apresenta uma lista de alunos associados ao código da turma selecionada pelo professor.
Ator	Professor
Pré-condições	O professor deve ter selecionado uma turma para listar os alunos.
Fluxo principal	- O sistema verifica se existem alunos associados à turma selecionada; - O sistema exibe uma lista de alunos, contendo o nome de cada aluno; - O professor seleciona um aluno.
Fluxo de exceção	Se nenhum aluno estiver associado à turma selecionada, é exibida uma mensagem ao professor.
Pós-condições	O professor seleciona um aluno da lista de alunos.

Quadro 12 – Caso de uso “listar alunos associados à turma”

3.3.3 Diagrama de atividades do problema

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, os quais estão representados nas figuras de 9 a 18.

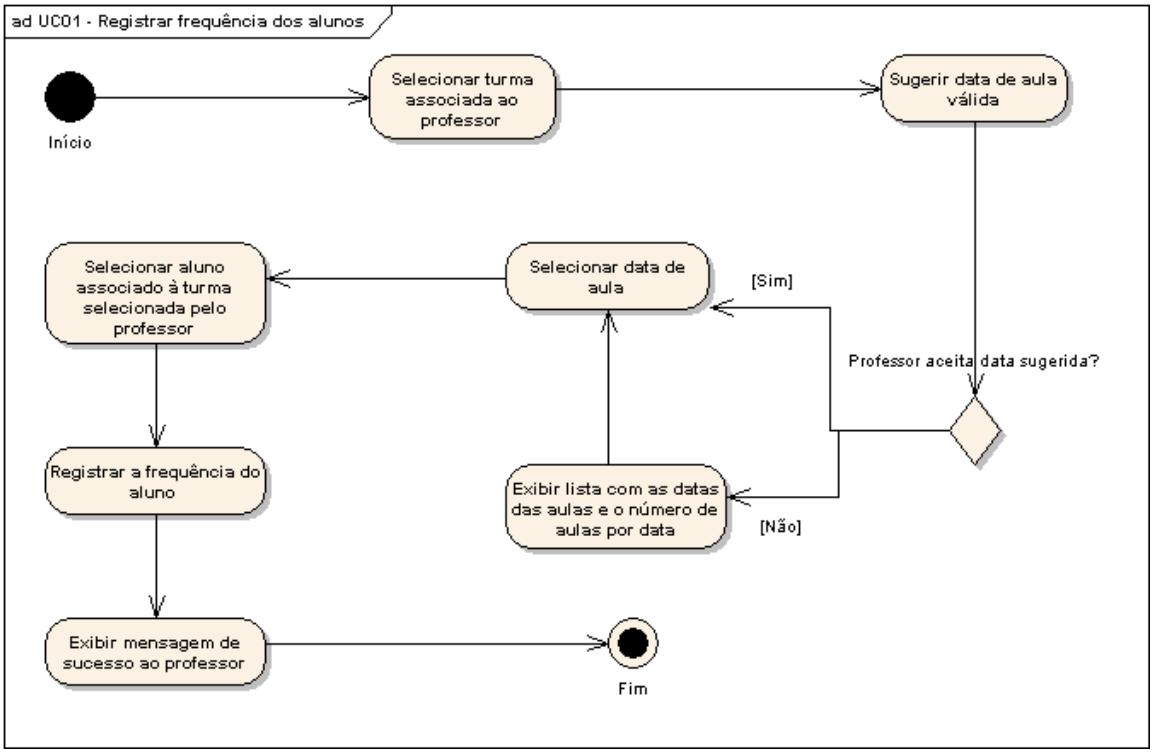


Figura 9 – Diagrama de atividades do caso de uso “registrar freqüência dos alunos”

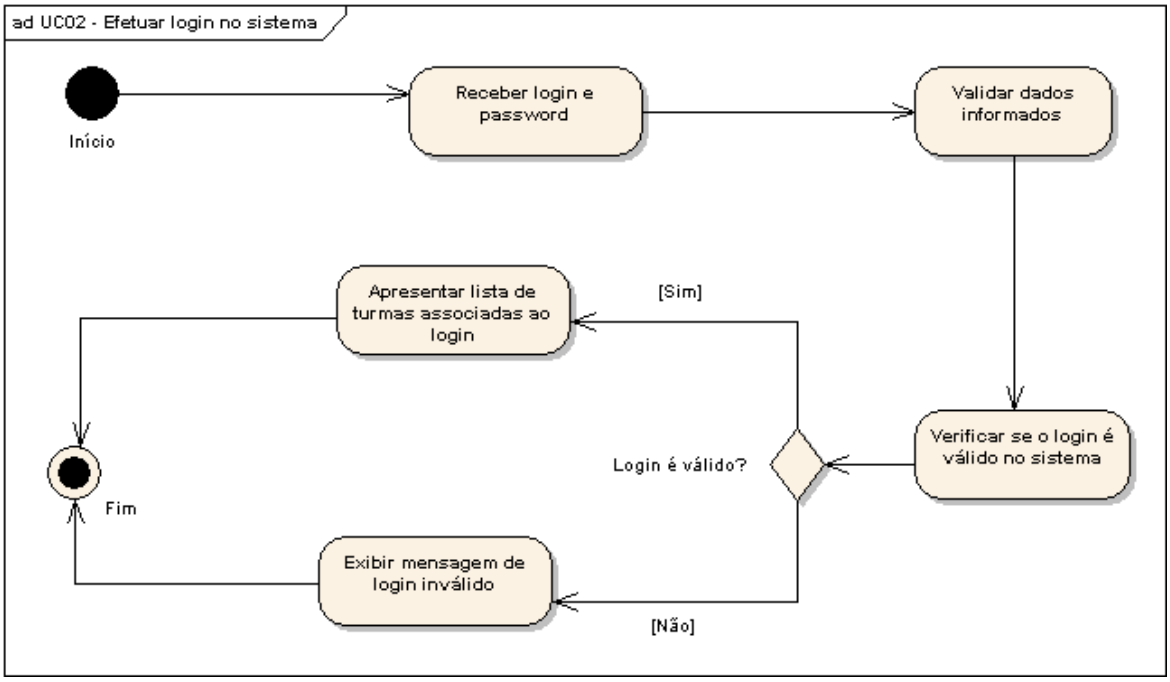


Figura 10 – Diagrama de atividades do caso de uso “autenticar o professor”

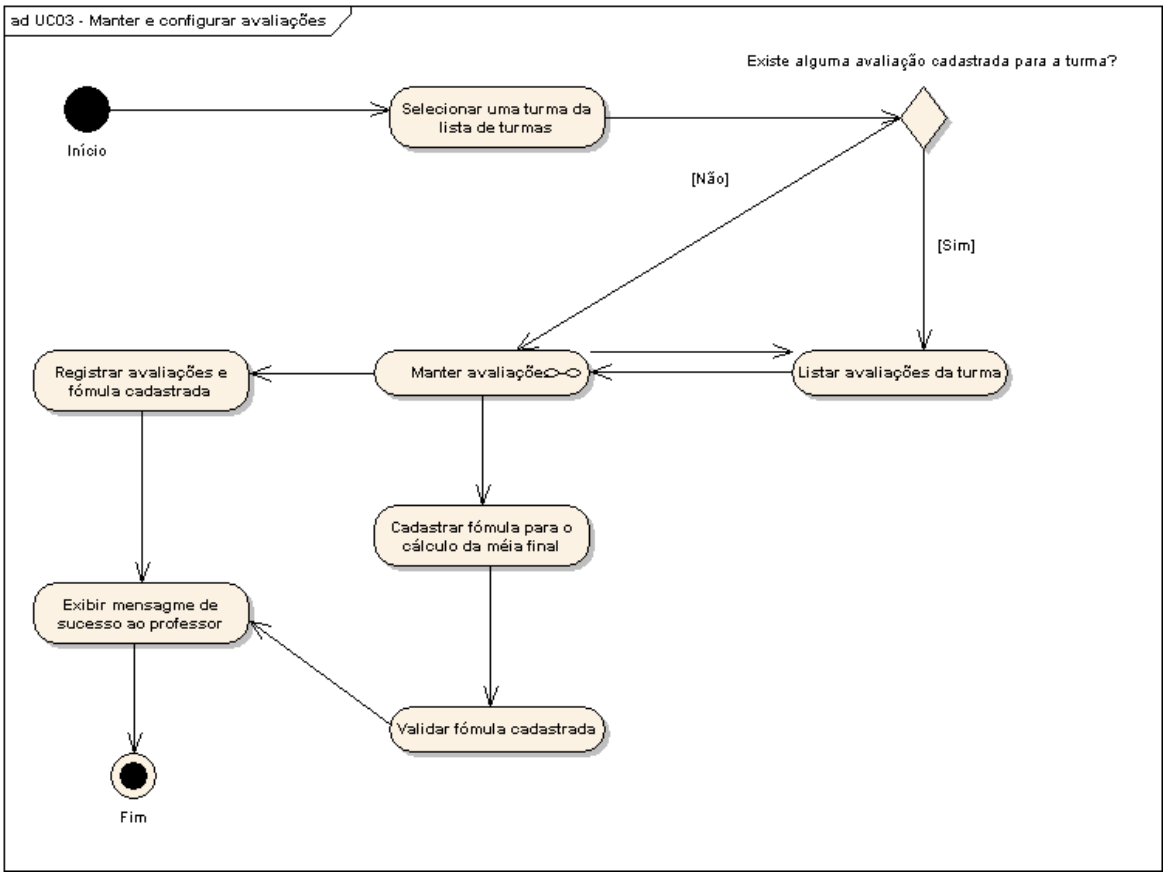


Figura 11 – Diagrama de atividades do caso de uso “manter e configurar avaliações”

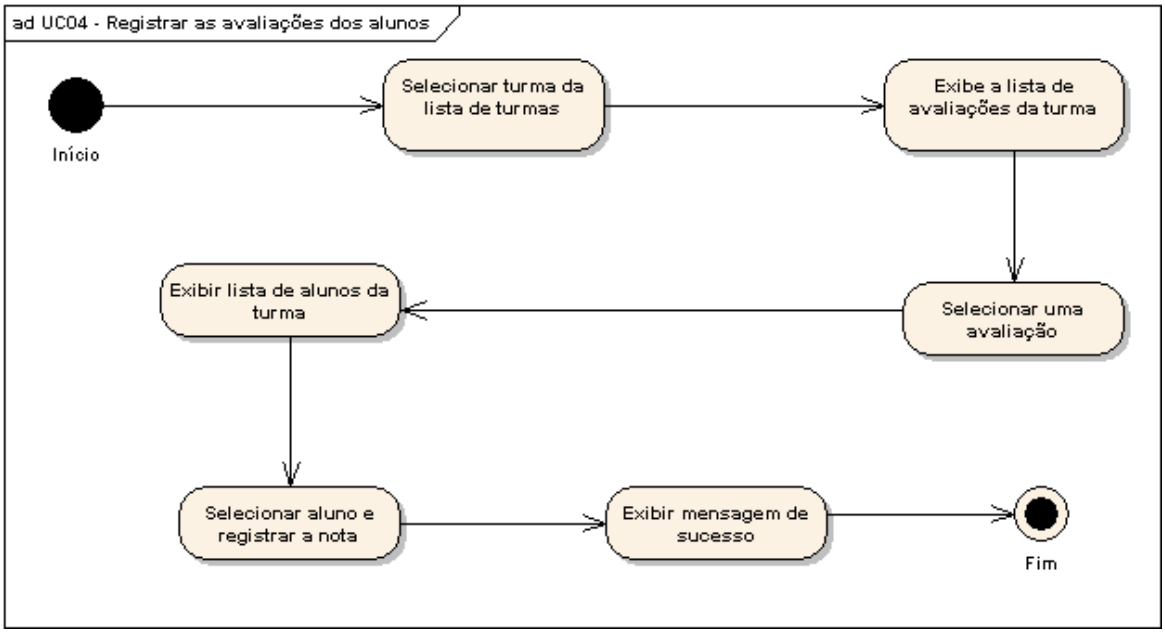


Figura 12 – Diagrama de atividades do caso de uso “registrar as avaliações dos alunos”

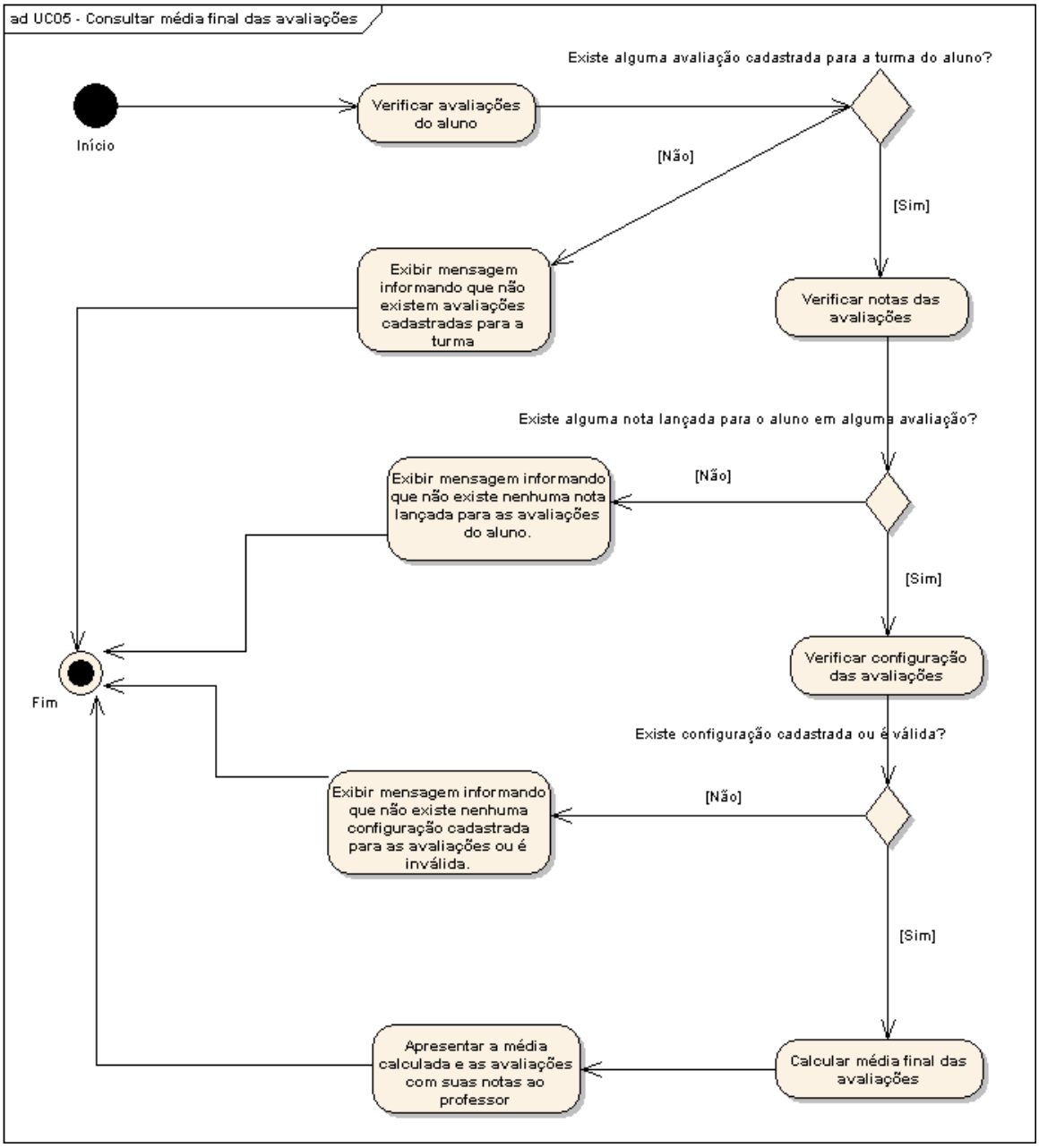


Figura 13 – Diagrama de atividades do caso de uso “consultar média final dos alunos”

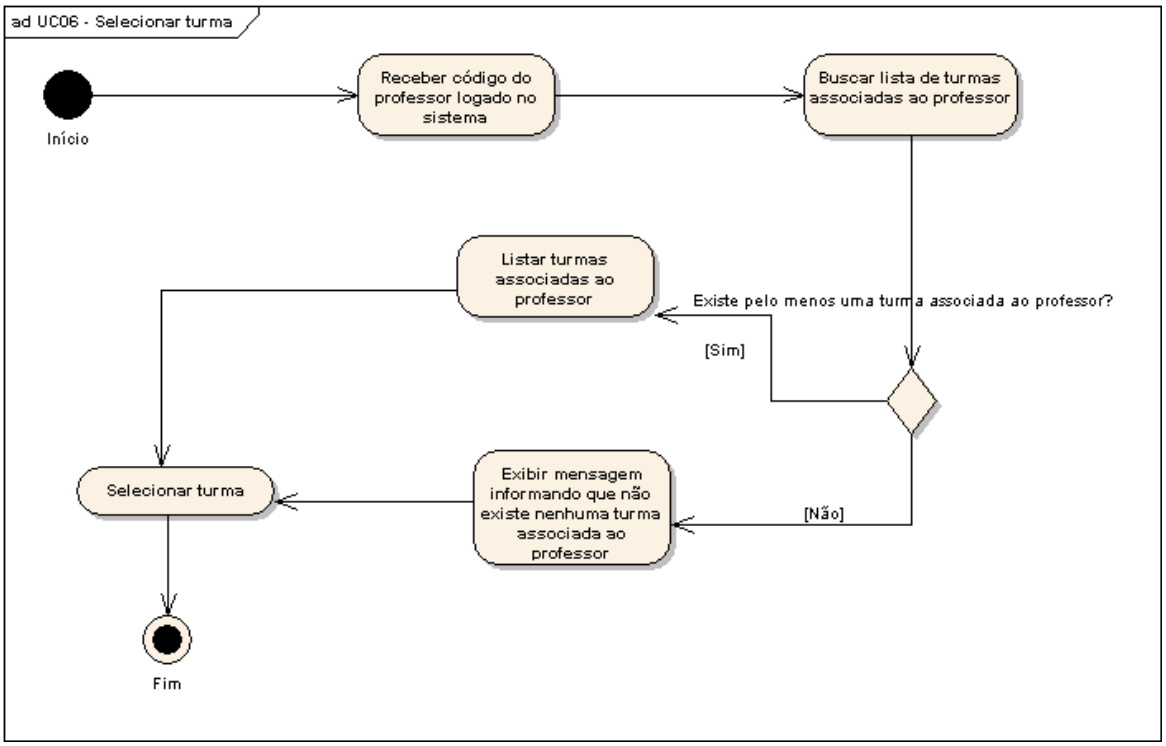


Figura 14 – Diagrama de atividades do caso de uso “selecionar turma”

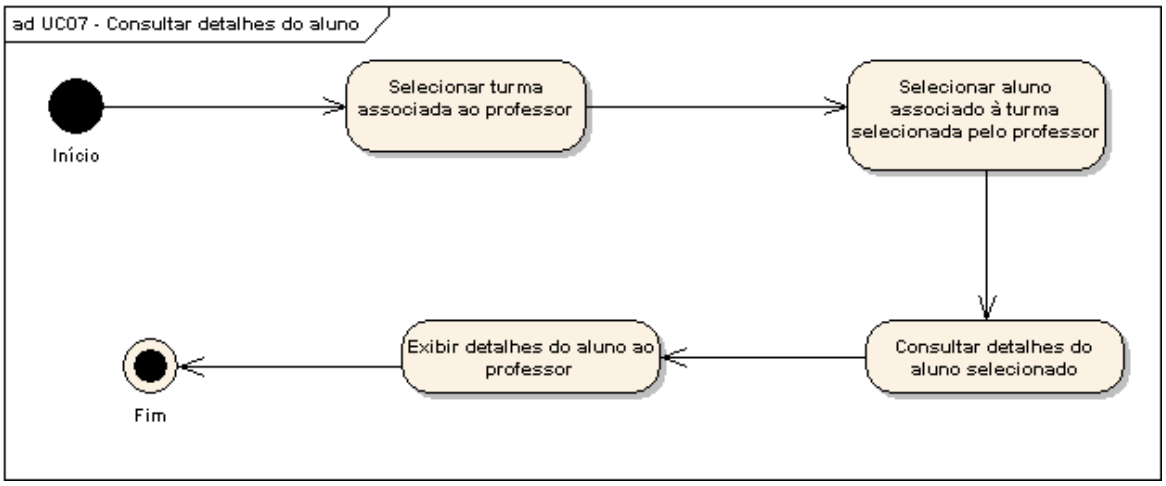


Figura 15 – Diagrama de atividades do caso de uso “consultar detalhes do aluno”

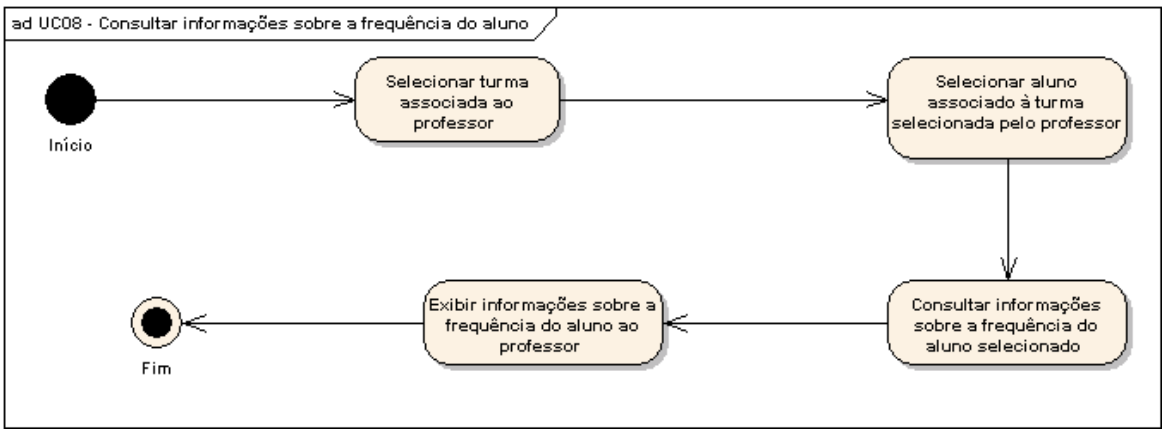


Figura 16 – Diagrama de atividades do caso de uso “consultar informações sobre a freqüência do aluno”

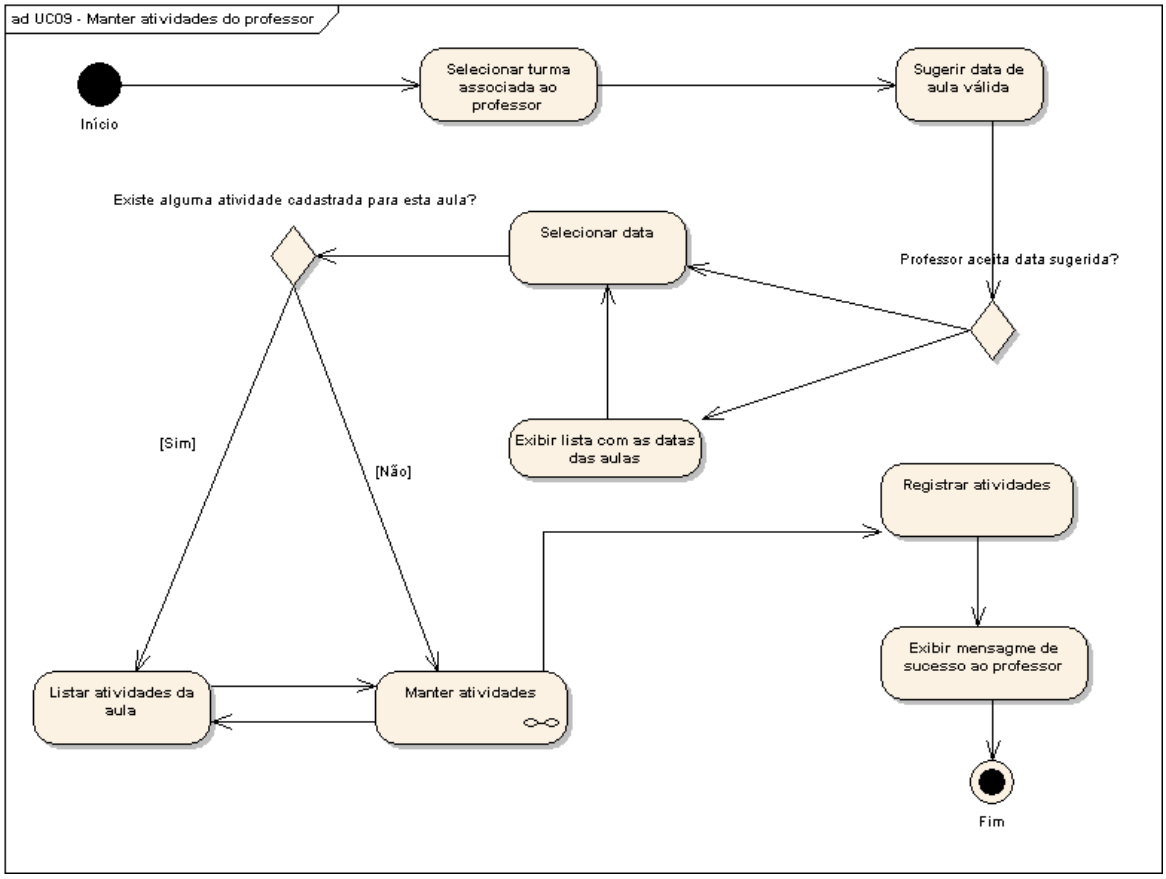


Figura 17 – Diagrama de atividades do caso de uso “manter atividades do professor”

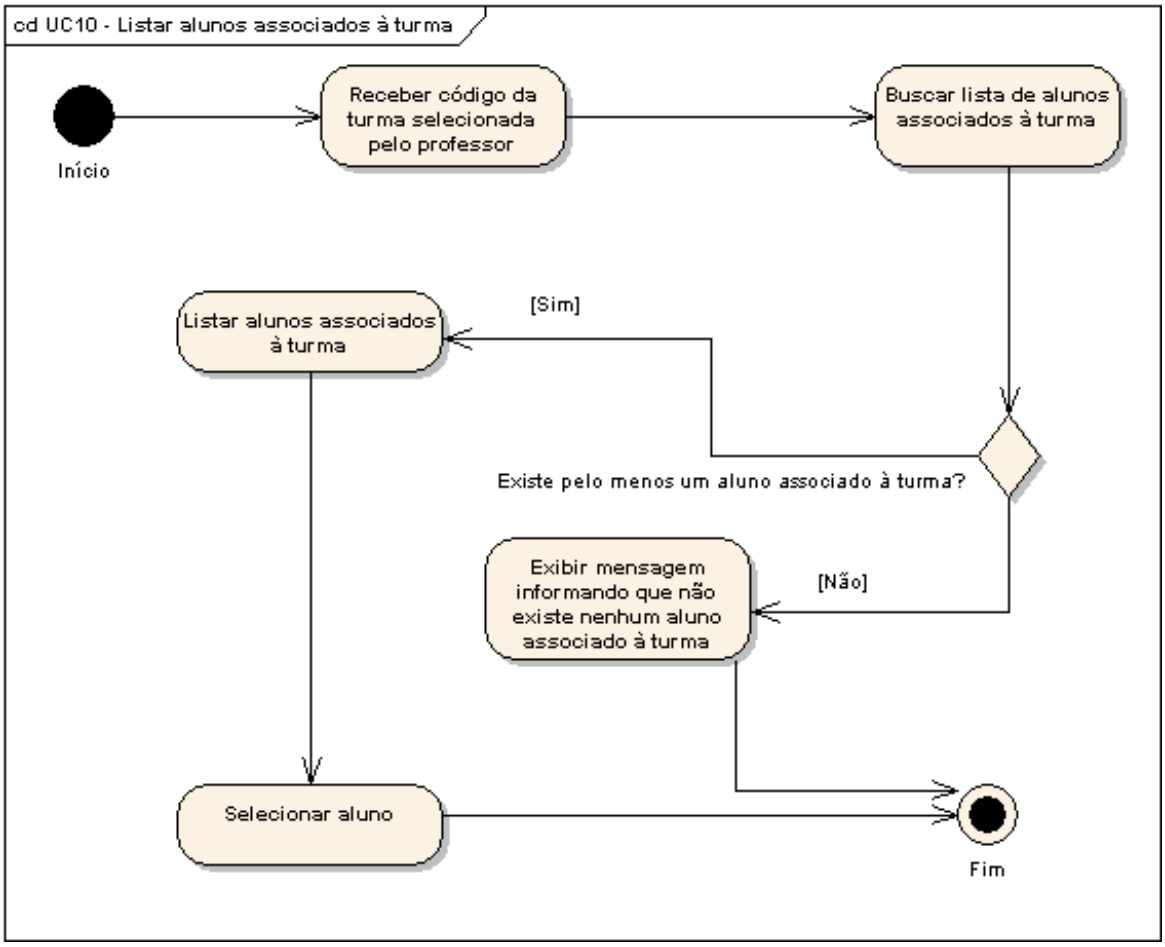


Figura 18 – Diagrama de atividades do caso de uso “listar alunos associados à turma”

3.3.4 Diagrama de classes

Os diagramas de classes são os diagramas encontrados com maior freqüência na modelagem de sistemas orientados a objetos, onde os diagramas demonstram um conjunto de classes, interfaces e colaborações e seus relacionamentos. Os diagramas de classes são importantes não só para a visualização, a especificação e a documentação de modelos estruturais, mas também para a construção de sistemas executáveis por intermédio da engenharia de produção e reversa.

Nesta seção serão especificados os diagramas de classes modelados para a especificação do protótipo. Os diagramas foram divididos em diagramas de análise e

diagramas de projeto.

3.3.4.1 Diagrama de classes do modelo conceitual

O modelo conceitual descreve a estrutura da informação que o sistema vai gerenciar. Trata-se de um artefato do domínio do problema e não do domínio da solução. O modelo conceitual não deve ser confundido com a estrutura da organização nem com o modelo de dados, pois este modelo visa representar a compreensão da informação, e não a representação física ou arquitetural do sistema.

Na Figura 19 estão representados os conceitos identificados a partir dos casos de uso do sistema. Estes conceitos estão representados por classes e associações, os quais auxiliaram no projeto das interfaces com o usuário, regras de negócio, e para definir como as informações devem estar organizadas em um sistema acadêmico, o qual deve ser acoplado ao componente de negócio do sistema. Estas classes não foram implementadas no projeto, pois serviram apenas para descrever a estrutura de informação de um sistema acadêmico.

A classe “Professor” representa um professor de uma universidade, este descrito por um código de pessoa e vínculo com a instituição de ensino, seu nome e uma senha para efetuar o login no sistema.

Um professor está associado a uma ou mais turmas, estas representadas pela classe “Turma”, descrita com um código, nome e uma fórmula para calcular a média final do aluno com base nas notas das avaliações cadastradas.

Uma turma pode ter várias avaliações e várias aulas associadas, representadas respectivamente pela classe “Avaliação” e pela classe “Aula”.

As aulas podem ter várias atividades que o professor realizará em sala de aula, descritas pela classe “Atividade”.

A classe “Aluno” representa os alunos de uma determinada turma, onde uma turma pode ter vários alunos associados e um aluno pode estar associado a várias turmas.

Para um aluno e uma avaliação, pode ser registrada uma nota da avaliação, representada pela classe “Nota”. Para um aluno e uma aula, pode ser registrada uma freqüência, representada pela classe “Frequencia”, onde é informado o tipo de freqüência “C” para seu comparecimento e o tipo de freqüência “F” para sua falta na aula.

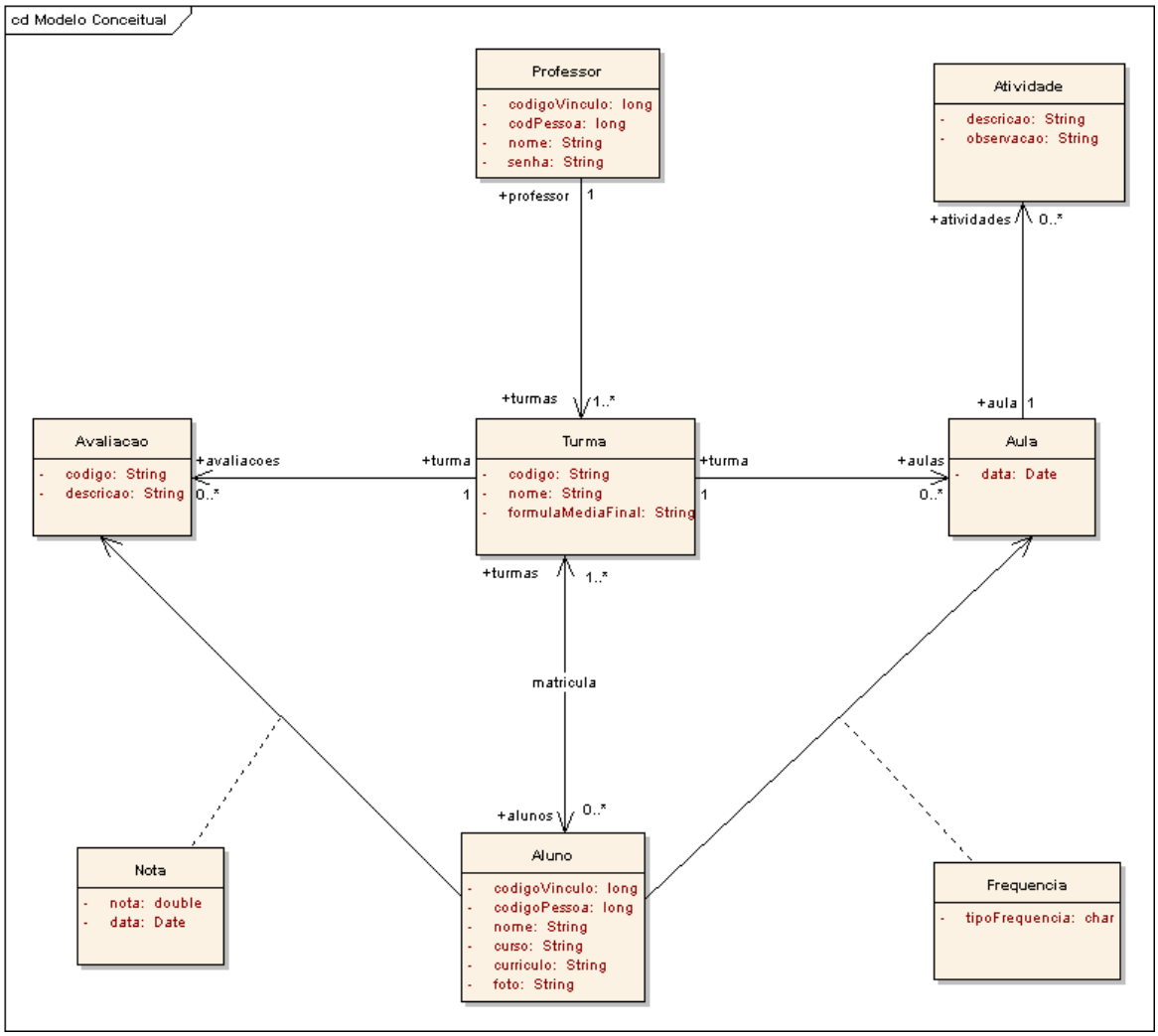


Figura 19 – Modelo conceitual de um sistema acadêmico

3.3.4.2 Diagrama de classes de projeto

Para especificar as classes que foram implementadas no projeto do módulo J2ME, que é executado no dispositivo celular, foi modelado um diagrama de classes seguindo o padrão de projeto MVC, onde as classes estão representadas na Figura 20.

Para a camada de visão, foram especificadas classes de interface com o usuário, agrupadas no pacote *view*, onde as classes utilizam elementos de interface do J2ME, como listas, formulários, imagens e caixas de texto. As classes deste pacote são responsáveis por apresentar as telas do sistema ao professor no dispositivo celular.

A classe “DiarioEletronicoMidlet”, representada na camada de controle, realiza as principais funções de controle do sistema, onde são especificados os métodos que as classes da camada de visão necessitam para apresentar ou receber as informações do sistema ao professor. Esta classe representa o único *midlet* da aplicação J2ME, o qual é executado ao ser iniciada a aplicação no dispositivo celular.

A classe “DiarioEletronicoFacade” é uma classe *facade* para comunicação com o *Web Service* que está disponível em um servidor de aplicações. Só é permitido uma instância desta classe em todo o sistema, pois ela representa a fachada para o *Web Service*. Esta classe é responsável por prover e enviar informações da camada de controle ao *Web Service*, onde sua implementação pode ser substituída sem nenhum impacto no sistema, por outra classe que utilize um protocolo diferente do SOAP para se comunicar com um servidor.

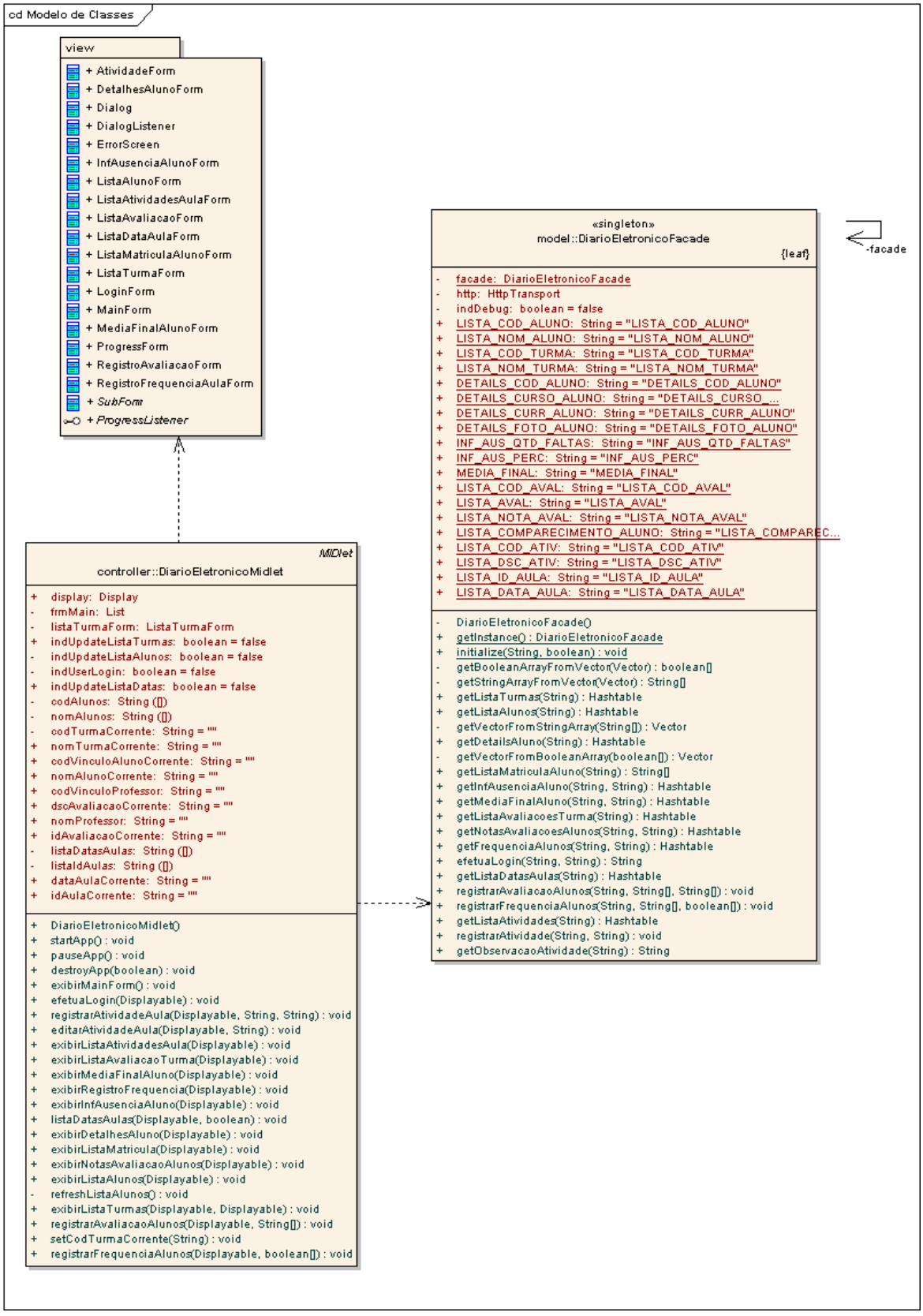


Figura 20 – Diagrama de classes de projeto do módulo J2ME

Na Figura 21, são representada as principais classes que compõem o módulo que será

executado no servidor de aplicações. As classes deste diagrama também foram modeladas seguindo o padrão de projeto MVC,

A classe “DiarioEletronicoAction”, da camada de controle, implementa todas as funções de controle para as páginas Web disponíveis ao professor neste módulo.

A classe “DiarioEletronicoBean”, da camada de controle, implementa a especificação de um EJB de sessão, onde a principal função desta classe é agrupar os métodos de negócio do sistema usados pela aplicação J2ME como as páginas Web, e disponibilizar em forma de Web Service seus métodos de negócio, para que a aplicação em J2ME possa utilizar através de mensagens SOAP.

A classe “DiarioEletronicoFacade”, da camada de modelo, é responsável em implementar a fachada entre o EJB e o sistema acadêmico, onde esta tem os métodos de negócio necessários pela aplicação para consultar ou manipular as informações acadêmicas. Só é permitido uma instância desta classe em todo o sistema, pois ela representa a fachada para o sistema acadêmico.

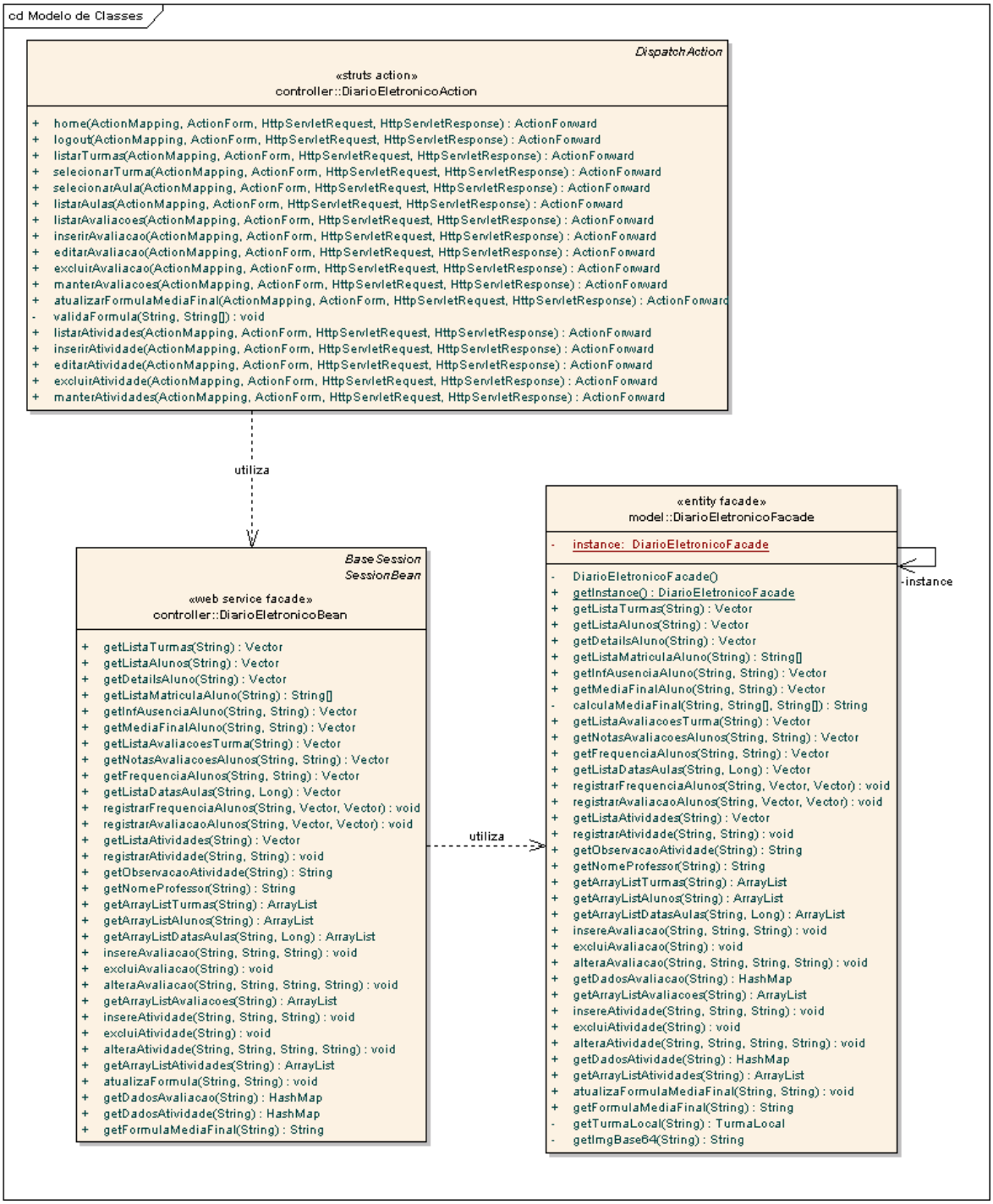


Figura 21 – Diagrama de classes de projeto do módulo do WEB

3.3.5 Realizações de casos de uso

O diagrama de seqüência, também conhecido por realização do caso de uso, demonstra como as instâncias de classe interagem entre si relatando a troca de mensagem entre elas.

Com o objetivo de promover a descoberta de métodos nas principais classes, aplicou-se o diagrama de seqüência apenas nos principais casos de uso:

a) registrar freqüência dos alunos (figura 22);

b) registrar as avaliações dos alunos (figura 23);

c) selecionar turma (figura 24);

d) listar alunos associados à turma (figura 25).

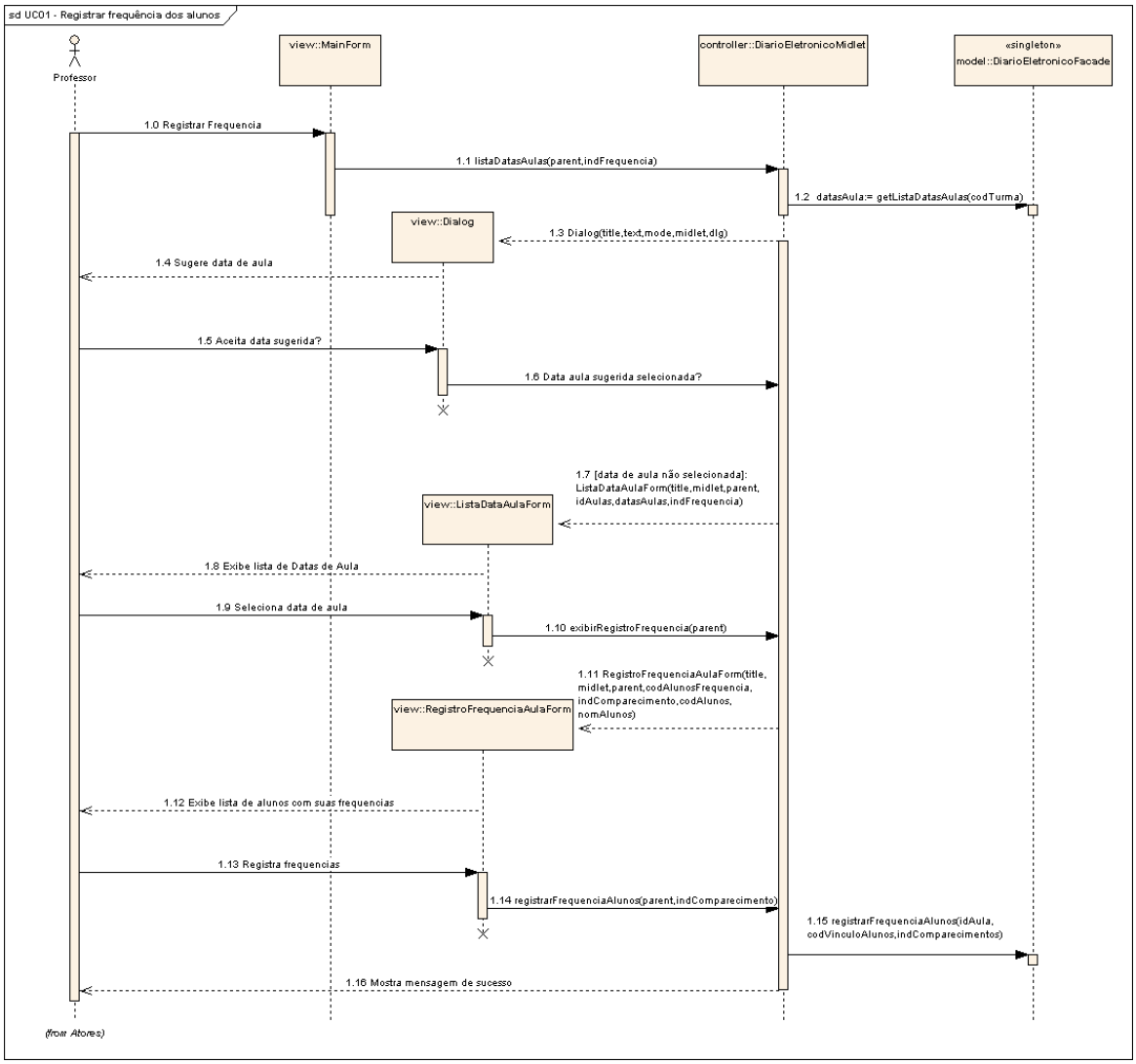


Figura 22 – Diagrama de seqüência do caso de uso “registrar freqüência dos alunos”

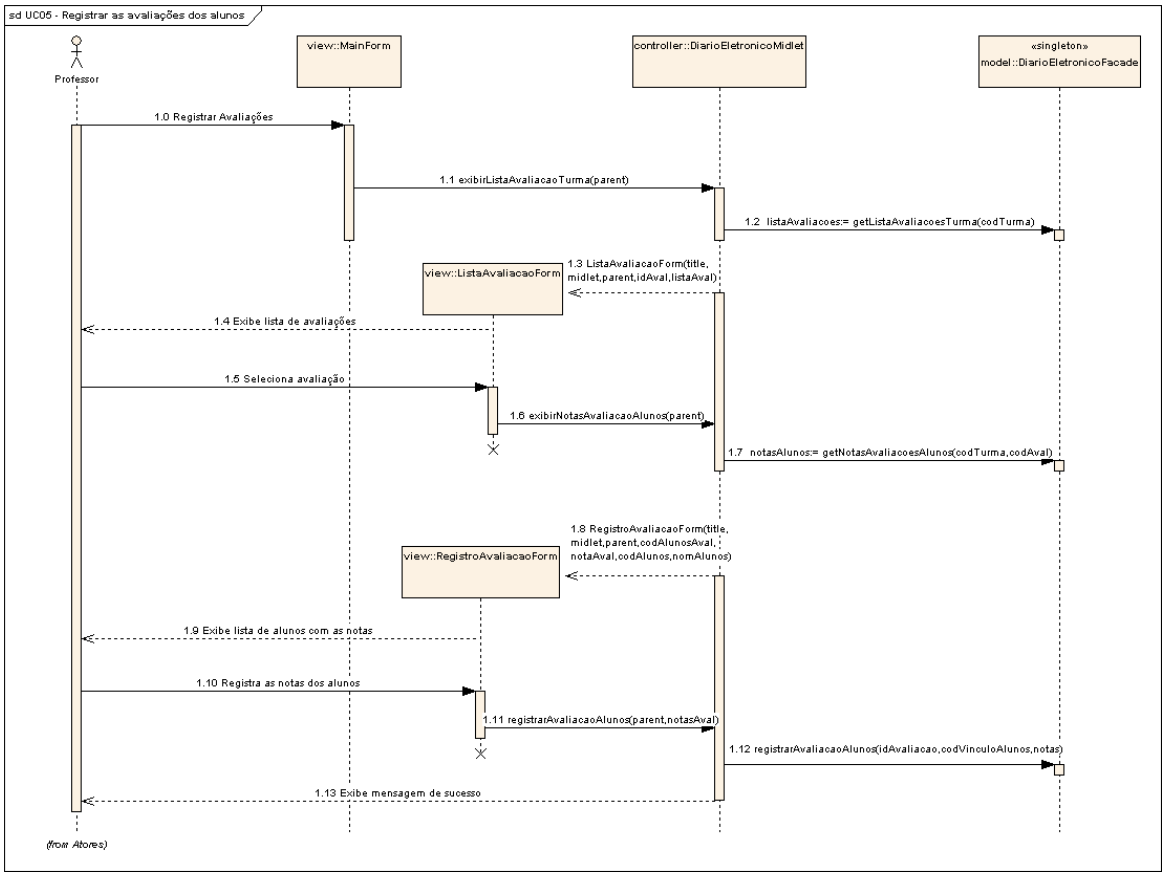


Figura 23 – Diagrama de seqüência do caso de uso “registrar as avaliações dos alunos”

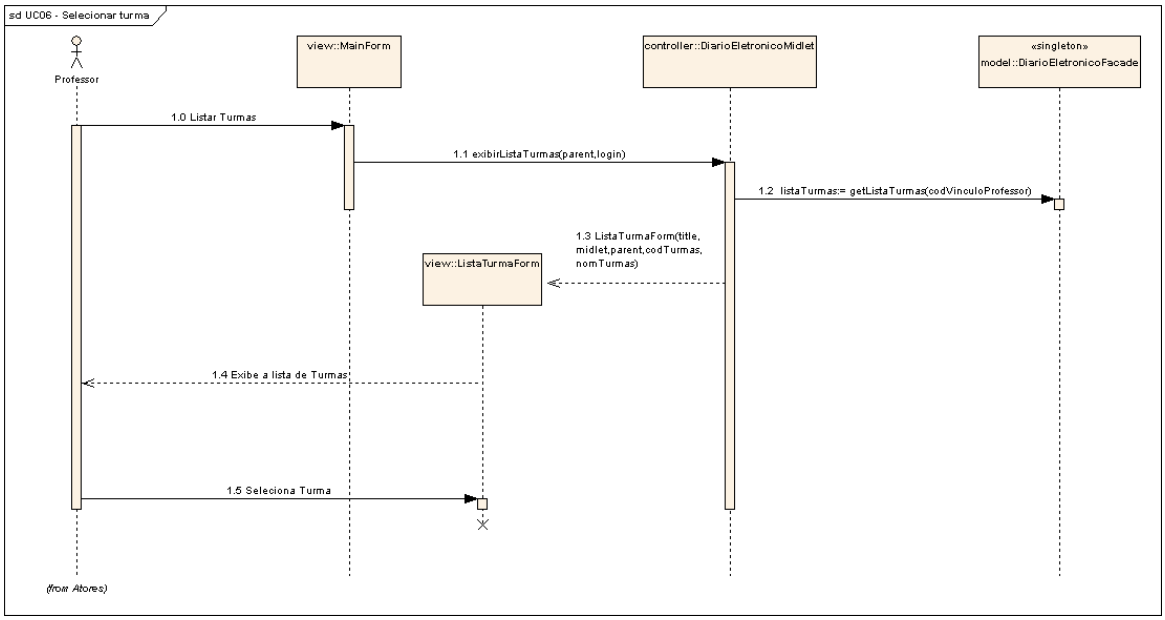


Figura 24 – Diagrama de seqüência do caso de uso “selecionar turma”

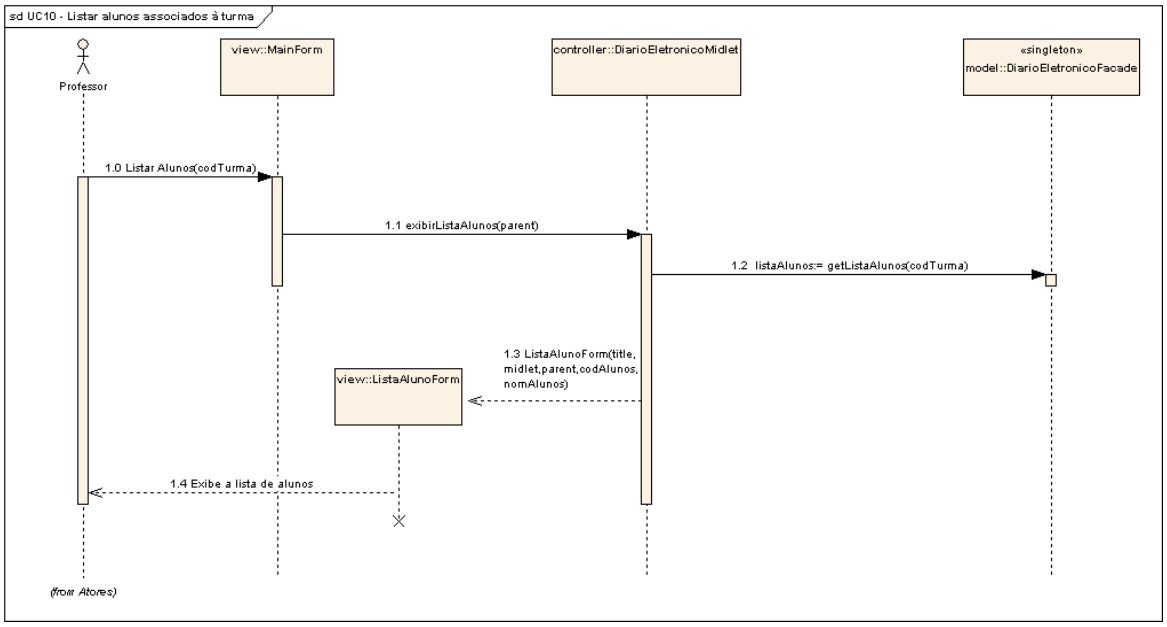


Figura 25 – Diagrama de seqüência do caso de uso “listar alunos associados à turma”

3.3.6 Diagrama de Implantação

O diagrama de implantação tem como objetivo modelar os aspectos físicos de um sistema orientado a objetos. O diagrama de implantação mostra a configuração dos nós de processamento em tempo de execução e os componentes que nele existem.

Na Figura 26 são representados os dois nós de processamento do sistema, o dispositivo móvel e o servidor de aplicações e os artefatos gerados pelo projeto e os componentes dos artefatos. No dispositivo móvel, são executados dois artefatos, o arquivo “DiarioEletronico.jar”, contendo as classes e recursos necessários para a aplicação J2ME, e o arquivo “DiarioEletronico.jad”, o descritor de uma aplicação J2ME. No servidor de aplicações, é executado o arquivo “diarioweb.ear”, que contém quatro artefatos:

a) O arquivo “diarioweb.war” e seus componentes, que representa a aplicação Web;

b) O arquivo “diarioweb-ejb.jar” e seus componentes, contendo o EJB do sistema;

c) O arquivo “diarioweb-ws.wsr”, contendo o descritor do Web Service;

d) O descritor de uma aplicação J2EE “application.xml” do arquivo “diarioweb.ear”.

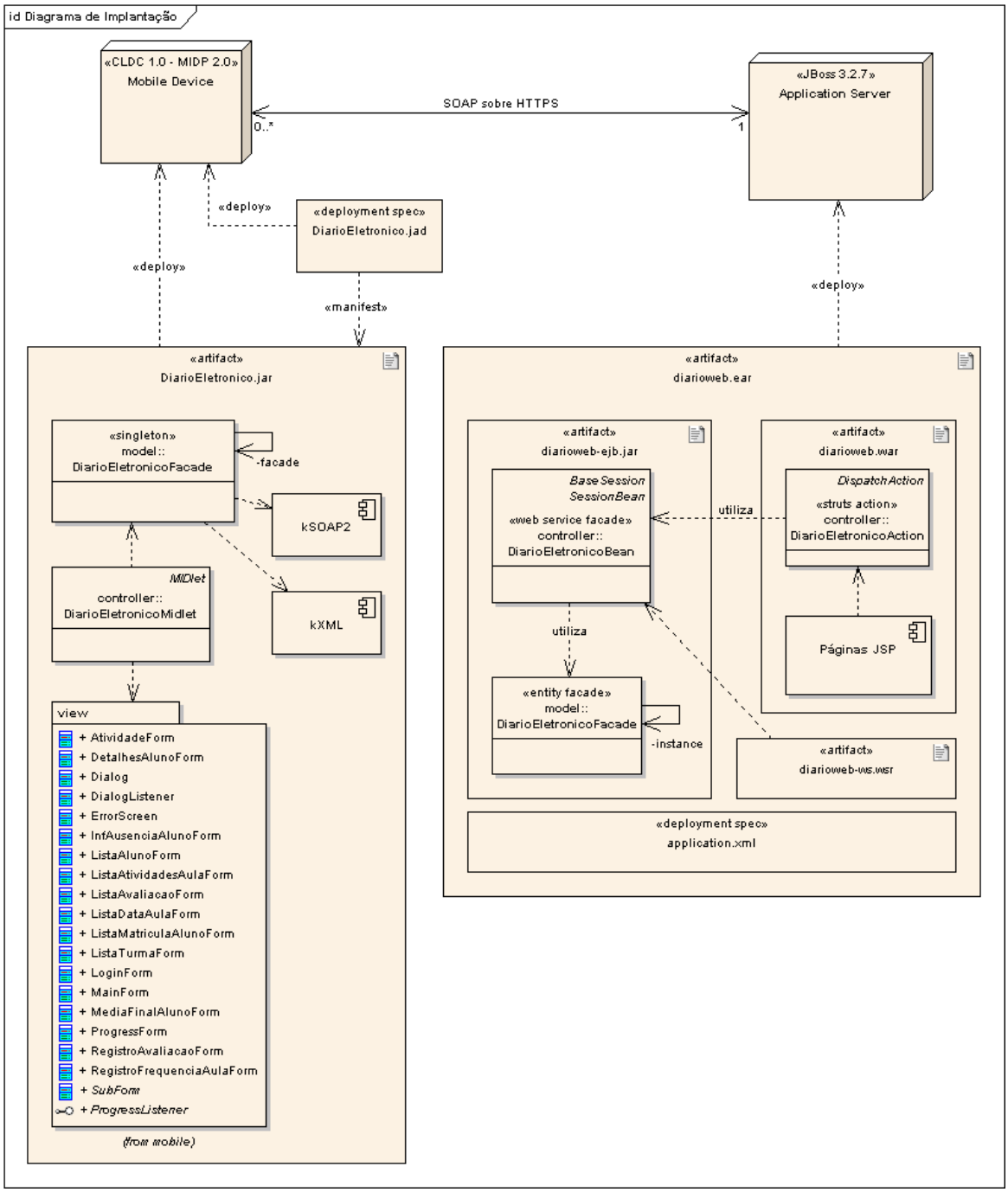


Figura 26 – Artefatos e seus ambientes de execução

3.3.7 Ferramenta utilizada na especificação do trabalho

Para a modelagem e especificação do projeto, utilizou-se a ferramenta CASE Enterprise Architect, baseada na linguagem UML. Segundo Sparx Systems (2005), o

Enterprise Architect é uma ferramenta CASE orientada a objetos voltada ao completo ciclo de vida do desenvolvimento de um projeto, fornecendo um ambiente para o desenvolvimento de software, gerenciamento de projetos, gerenciamento de requisitos e analise de negócio.

Com o uso do Enterprise Architect, foi possível realizar a modelagem do projeto em diferentes visões de sistema:

a) visão de caso de uso;

b) visão lógica;

c) visão dinâmica;

d) visão de implantação.

A visão de caso de uso foi utilizada para organizar os requisitos do sistema, agrupando-os em requisitos funcionais e requisitos não funcionais. A partir dos requisitos funcionais foram identificados os casos de uso e atores do sistema, utilizando o diagrama de casos de uso para a modelagem.

Na visão dinâmica, foram modeladas as classes identificadas a partir dos casos de uso nos diagramas de classes. A ferramenta permitiu a geração de código fonte em Java e a engenharia reversa para o sincronismo com os diagramas.

Para a modelagem dos diagramas de seqüência, foi utilizada a visão dinâmica, onde foi descrita a troca de mensagens entre os objetos e métodos de negócio do sistema.

Na visão de implantação, foi representado os aspectos físicos do sistema, como ambientes de execução e componentes, utilizando o diagrama de implantação para a modelagem.

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 e implementação do sistema utilizaram-se ferramentas e técnicas que auxiliaram no decorrer do trabalho, mantendo as informações organizadas e consistentes, e agilizando o processo de desenvolvimento.

A seguir são descritas as funcionalidades utilizadas de cada ferramenta e técnicas de desenvolvimento.

3.4.1.1 Plataforma Eclipse

O Eclipse é um dos ambientes de desenvolvimento mais popular atualmente para a plataforma Java e considerado uma das ferramentas chaves em se tratando de iniciativas Open-Source. O Eclipse possui facilidades como controle de projetos, compilação, depuração, execução, visualização de todos os arquivos contidos no projeto de forma clara e ferramentas de gerenciamento de trabalho coletivo.

Graças à tecnologia de *plugins*, o Eclipse permite personalizar o ambiente de trabalho do desenvolvedor de acordo com o projeto que está sendo desenvolvido. Esta característica extensível da ferramenta possibilitou a instalação de *plugins* que auxiliaram em atividades específicas no projeto. A seguir são descritos dois *plugins* utilizados no trabalho.

3.4.1.1.1 Plugin EclipseME

O *plugin* EclipseME utilizado juntamente com a ferramenta Eclipse forneceu recursos

para o desenvolvimento de aplicações J2ME, permitindo a criação de projetos com MIDlets e a configuração dos principais aspectos de uma aplicação J2ME, como por exemplo, o perfil e configuração do MIDlet.

3.4.1.1.2 Plugin JBoss-IDE

O *plugin* JBoss-IDE permitiu a integração com o JBoss, onde este é um servidor de aplicações open-source que implementa a especificação J2EE, oferecendo recursos para iniciar e parar o serviço do JBoss, efetuar a depuração de código no servidor e gerar arquivos de pacote. Um dos principais recursos do *plugin*, foi auxiliar na geração de código e descritores de aplicação, utilizando-se do mecanismo de automação de tarefas de desenvolvimento repetitivo e gerador de código XDoclet. O Apêndice A demonstra a utilização do XDoclet para geração de artefatos para o EJB e o *Web Service*.

3.4.1.2 J2ME Wireless Toolkit

J2ME Wireless Toolkit é o ambiente de desenvolvimento de aplicações J2ME fornecido pela Sun Microsystems, onde este foi usado em conjunto com o *plugin* EclipseME para executar os MIDlets em um emulador utilizando a máquina virtual KVM, fornecer as APIs e documentações necessárias para a construção da aplicação J2ME.

3.4.1.3 Bliblioteca kSOAP2 e kXML

kSOAP2 e *kXML* são bibliotecas para a utilização de SOAP e XML otimizadas para serem executas em uma aplicação J2ME. Estas bibliotecas foram utilizadas em conjunto no

projeto para prover a comunicação do dispositivo móvel com o Web Service, decodificando envelopes SOAP no formato XML. O Apêndice B demonstra o emprego da biblioteca *kSOAP2*.

3.4.1.4 Segurança na comunicação com o Web Services

Para que a aplicação que é executada no dispositivo celular utilize a comunicação de forma segura e confidencial com o *Web Service*, foi utilizado um certificado digital no servidor e o protocolo HTTPs para estabelecer a comunicação com o servidor de aplicações.

Também foi configurado no *Web Service* uma autenticação do tipo *Basic*, para que os métodos do *Web Service* não fiquem expostos na Internet, assim, só permitindo que uma aplicação autenticada utilize o *Web Service*. O Apêndice C demonstra o método que estabelece a comunicação HTTPs com a autenticação *Basic* utilizado pela aplicação J2ME.

Ao ser solicitado um método do *Web Service* sem ser feita a autenticação, é retornado um envelope SOAP informando que o acesso foi negado, conforme o Quadro 13.

```
<soapenv:Envelope>
   <soapenv:Body>
      <soapenv:Fault>
         <faultcode>soapenv:Server.generalException</faultcode>
         <faultstring>Access denied.</faultstring>
      </soapenv:Fault>
   </soapenv:Body>
</soapenv:Envelope>
```

Quadro 13 – Mensagem SOAP informando o acesso negado

3.4.1.5 Cálculo da expressão matemática da fórmula da média final

Para efetuar o cálculo da média final do aluno, em que a fórmula cadastrada pelo

professor é uma expressão matemática, onde esta pode conter variáveis que são os códigos das avaliações, foi necessário utilizar um analisador léxico e sintático. Estes analisadores foram gerados a partir de uma ferramenta de geração de código em Java, o Gerador de Analisadores Léxicos e Sintáticos (GALS), também construída com a linguagem Java. Foi necessário especificar uma entrada para o analisador léxico e sintático, para que o GALS pudesse gerar as classes Java necessárias. O Apêndice D demonstra a especificação da entrada.

3.4.1.6 Padrões de projeto implementados

Alguns padrões de projeto foram utilizados com o objetivo de oferecer flexibilidade e qualidade no desenvolvimento do trabalho. Pode-se relacionar três padrões de projeto que tiveram grande influencia no trabalho, o padrão MVC, *Facade* e *Singleton*. A seguir são detalhadas as utilizações dos mesmos no projeto.

3.4.1.6.1 Emprego do padrão MVC

O padrão MVC foi empregado no desenvolvimento da aplicação J2ME e da aplicação J2EE. Sua utilização foi muito importante para separar as classes em camadas de visão, controle e modelo, deixando clara a responsabilidade e funcionalidade de cada classe em sua camada respectiva.

3.4.1.6.2 Emprego do padrão *Facade* e *Singleton*

Para que as camadas de visão e controle pudessem abstrair a implementação da comunicação com o Web Service da aplicação J2ME e a troca de informações com o sistema acadêmico pela aplicação J2EE, foi empregado em conjunto os padrões *Facade* e *Singleton*, onde o padrão *Singleton* permitiu que existisse apenas uma instância do objeto no sistema, e o padrão *Facade* permitiu representar uma "fachada" entre a aplicação e o sistema acadêmico e *Web Service*. O Apêndice D demonstra a implementação desses padrões na classe "DiarioEletronicoFacade" responsável pela fachada com o sistema acadêmico.

3.4.2 Operacionalidade da implementação

Para demonstrar o funcionamento do protótipo proposto, são apresentadas a seguir algumas funções passo a passo da utilização do sistema simulando um caso real de utilização.

Ao ser iniciada a aplicação no celular, uma tela de login é exibida ao professor (figura 27), onde o mesmo informa seu código de usuário e sua senha. Há duas opções de escolha, opção "Ok" e opção "Sair". A opção "Ok" inicia o processo de autenticação do professor e, se os dados informados estiverem corretos, é exibida a lista de turmas. A opção "Sair" finaliza o sistema.

Figura 27 – Tela de login ao iniciar a aplicação

Ao requisitar o acesso à aplicação, é exibida uma tela de confirmação ao usuário, solicitando a autorização para acessar os recursos de rede do dispositivo celular (figura 28), conforme a política de segurança do J2ME.

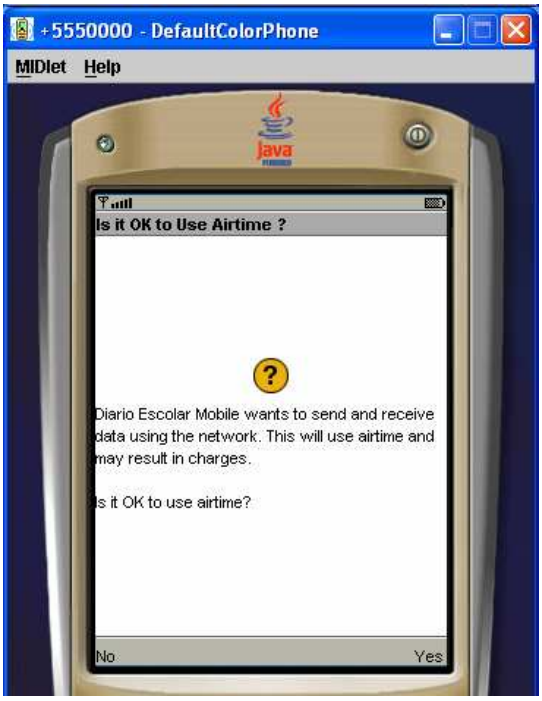


Figura 28 – Aplicação J2ME requisitando dados da rede

Inicialmente, o professor deve obrigatoriamente selecionar uma turma escolar a partir da lista de turmas, onde é exibido o código e o nome das turmas associadas ao professor, conforme a Figura 29.

Figura 29 – Tela com a lista de turmas

Ao ser selecionada uma turma, é exibida a tela principal com um menu em forma de

lista (figura 30), onde são oferecidas as opções a seguir:

a) registrar freqüência;

b) registrar avaliação;

c) registrar atividades;

d) listar alunos;

e) selecionar turma;

f) sobre.

Figura 30 – Tela principal com o menu

Ao selecionar a opção “Registrar Freqüência”, é sugerida pelo sistema, uma data de aula da turma, conforme a Figura 31, para o registro da freqüência.

Figura 31 – Tela com data de aula sugerida pelo sistema

Há duas opções de escolha, “Sim” e opção “Não”. A opção “Sim” seleciona a data de aula sugerida pelo sistema e exibe a tela para ser efetuado o registro da freqüência. Se for selecionada a opção “Não”, o professor não irá efetuar o registro da freqüência na data sugerida pelo sistema, mas deverá selecionar uma outra data de aula a partir de uma lista de datas de aula da turma no semestre letivo, conforme a Figura 32.

Figura 32 – Tela com a lista de datas de aulas

Após ser selecionada uma data de aula, é exibida a tela com uma lista de nome de alunos da turma corrente, onde é permitido indicar se o aluno está presente na aula ou não. Para cada aluno da lista, é exibida uma imagem, que de acordo com a indicação da opção, é representada visualmente ao professor a falta do aluno (imagem “F”), ou o seu comparecimento na aula selecionada (imagem “C”). Inicialmente, todos os alunos estão indicados como “F”, ou falta na aula. Na Figura 33, é exibida a tela para o registro da freqüência, com quatro alunos presentes (“C”), e dois alunos com falta (“F”).

Figura 33 – Tela para efetuar o registro da freqüência dos alunos

Ao modificar a freqüência dos alunos é necessário que as informações sejam salvas através da escolha da opção “Salvar”. Uma mensagem confirmando o sucesso da operação é mostrada ao professor. A opção “Voltar” retorna a tela principal.

Após o professor ser notificado sobre o sucesso da operação, é retornada a tela principal. Na tela principal, o professor seleciona a opção “Registrar avaliação”, para registrar as notas dos alunos em uma avaliação. Antes de ser exibida a tela para o registro das notas, o professor deve selecionar, a partir de uma lista de avaliações da turma, conforme a Figura 34, a avaliação desejada para o registro das notas.

Figura 34 – Tela com a lista das avaliações da turma

As avaliações exibidas na Figura 34 são mantidas (inseridas, alteradas, excluídas e consultadas) no módulo web, conforme Figura 35.

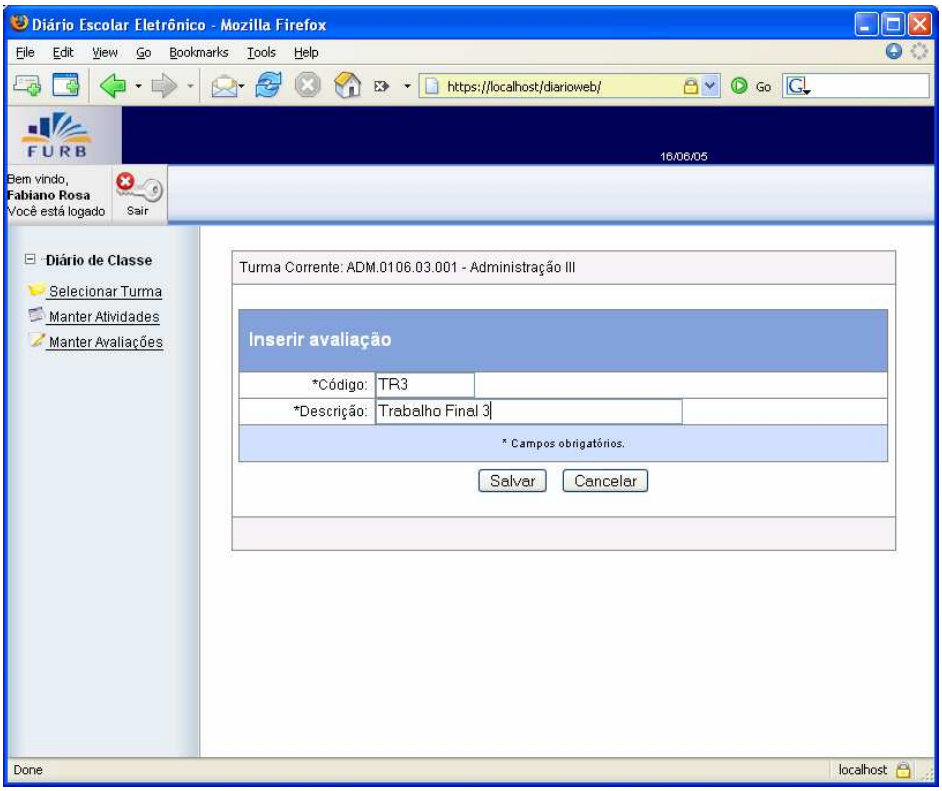


Figura 35 – Tela para inserir uma nova avaliação

Ao selecionar a avaliação desejada, o professor seleciona a opção do menu “Nota dos Alunos” para ser exibida a tela para o registro das notas. A opção “Voltar” retorna à tela principal.

Conforme a Figura 36, para ser registrada a nota dos alunos, é exibida uma lista com o nome dos alunos e uma caixa de texto para ser digitada a nota na respectiva avaliação. A nota da avaliação para cada aluno deve seguir as regras definidas para o Diário de Classe pela instituição de ensino.

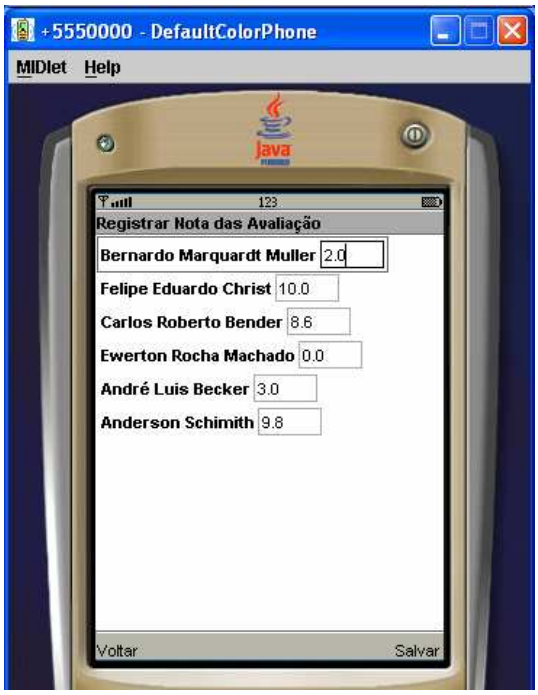


Figura 36 – Tela para efetuar o registro das notas dos alunos na avaliação

Ao modificar a nota da avaliação de cada alunos é necessário que as informações sejam salvas através da escolha da opção “Salvar”. Uma mensagem confirmando o sucesso da operação é mostrada ao professor. A opção “Voltar” retorna para a tela com a lista de avaliações.

Concluída a operação acima, é retornada a tela principal, e o professor seleciona a opção “Registrar Atividades”, onde o professor pode registrar o andamento das atividades realizadas em sala de aula. O sistema sugere uma data de aula, como foi visto na Figura 31, e o professor pode selecioná-la ou escolher outra data, a partir de uma lista de datas de aulas, já

exemplificada na Figura 32. Após a data de aula ser selecionada, é exibida a tela com uma lista de descrições das atividades (figura 37), para a seleção do professor.

Figura 37 – Tela com a lista de atividades do professor na aula

Selecionada a atividade, o professor registra observações a respeito da atividade da aula, em uma área de edição na tela de edição da atividade, conforme a figura 38.

Figura 38 – Registro do histórico da atividade realizada

Se for desejado armazenar as informações sobre a atividade, o professor seleciona a opção “Salvar” e é mostrada uma mensagem de sucesso, concluindo a operação. A opção “Voltar” retorna a tela com a lista de atividades. A manutenção das atividades, inclusão, alteração, exclusão e consulta das mesmas, pode ser feita no módulo web, conforme as Figuras 39 e 40.

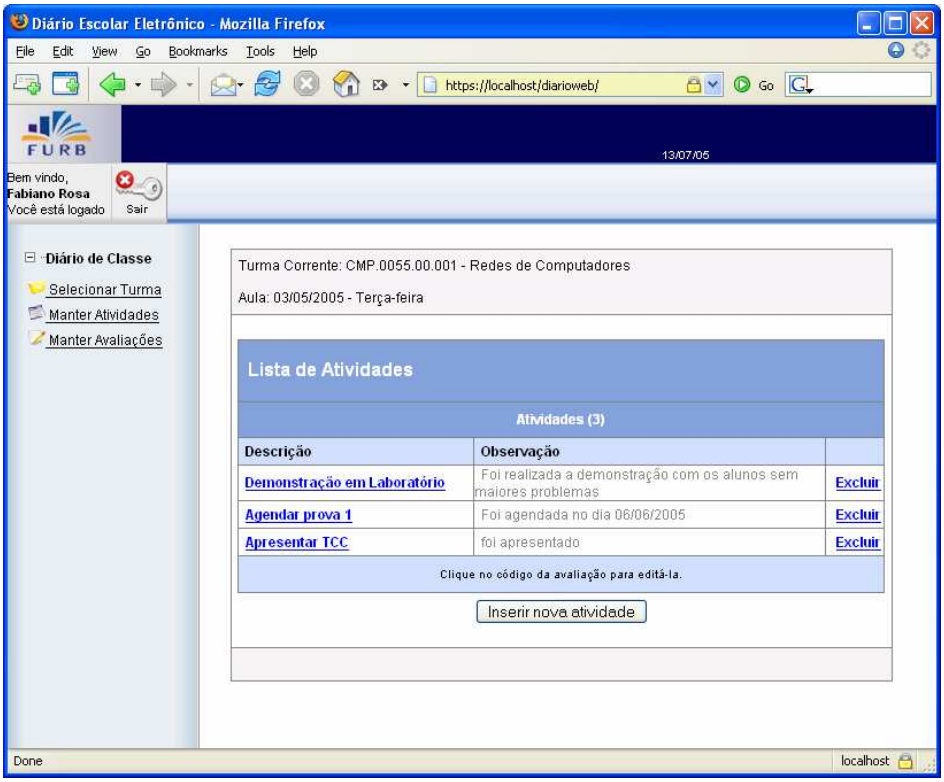


Figura 39 – Lista de atividades

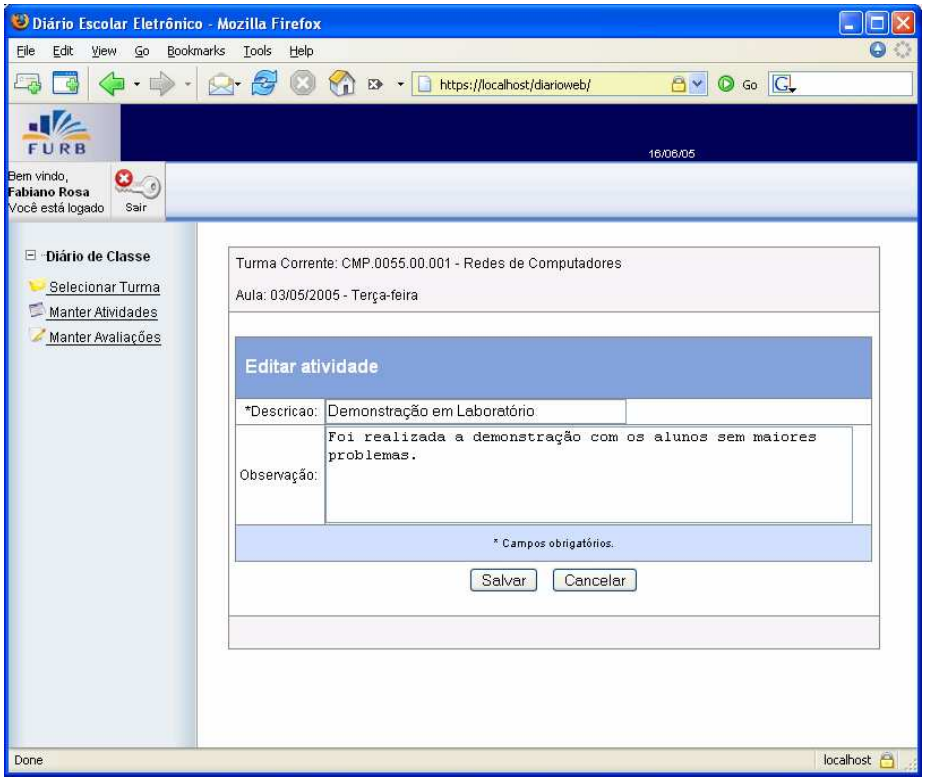


Figura 40 – Edição da atividade

Ao concluir o registro da atividade, é retornada a tela principal, onde o professor seleciona a opção “Lista de Alunos”, para visualizar os alunos da turma corrente. A Figura 41 demonstra a tela com a lista de alunos, exibindo o nome de cada aluno matriculado na turma.

Figura 41 – Tela com a lista de alunos da turma

Para cada aluno, está disponível as seguintes opções de menu:

a) detalhes: informações pessoais e acadêmicas do aluno;

b) ausência: informações sobre ausência das aulas do aluno;

c) média final: avaliações com as notas e a média final do aluno.

Selecionada a opção “Detalhes”, são exibidas a informações pessoais e acadêmicas sobre o aluno, listadas na tela de detalhes do aluno, conforme a Figura 42. As informações exibidas são descritas a seguir:

a) pessoa: código de pessoa do aluno na instituição;

b) vínculo: código de vínculo do aluno na instituição;

c) nome: nome completo do aluno;

d) curso: curso em que o aluno está matriculado (código - nome);

e) currículo: currículo atual do aluno (ano / semestre);

f) fotografia: foto do aluno.

Figura 42 – Tela com os detalhes do aluno

Na tela de detalhes do aluno, está disponível a opção “Matrícula”, que ao ser selecionada, é exibida uma tela com a lista dos nomes das turmas onde o aluno está matriculado, conforme a Figura 43.

Figura 43 – Tela listando a matrícula do aluno

Após visualizar os detalhes do aluno, o professor seleciona a opção “Voltar” para retornar a tela com a lista de alunos, e no menu, seleciona a opção “Ausência” (figura 41), onde são exibidas informações do aluno: o código do vínculo, nome completo, e informações sobre a freqüência nas aulas, como a quantidade de faltas e o percentual de ausência nas aulas ministradas pelo professor, conforme a Figura 44.

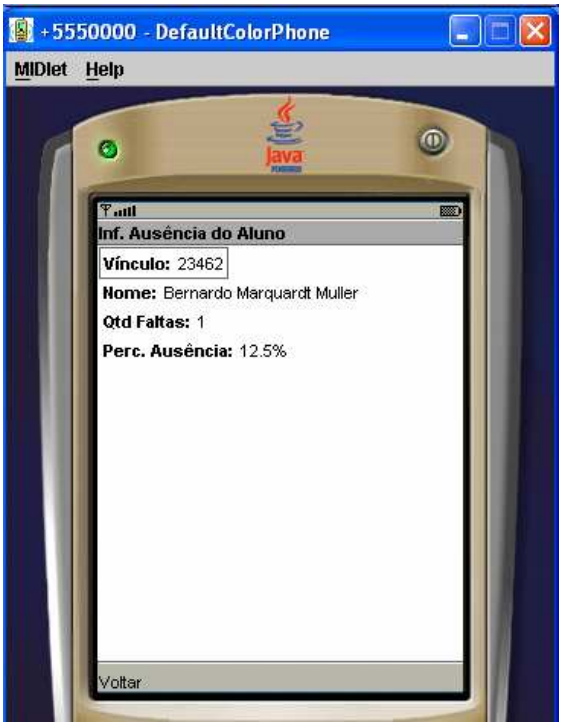


Figura 44 – Tela exibindo as informações da freqüência do aluno

Por fim, a terceira opção na lista de aluno, “Média Final”, apresenta ao professor uma lista com a descrição das avaliações, a nota de cada avaliação, e a média final do aluno, calculada com base em uma fórmula cadastrada na tela com a lista de atividades do módulo web, conforme a Figura 45.

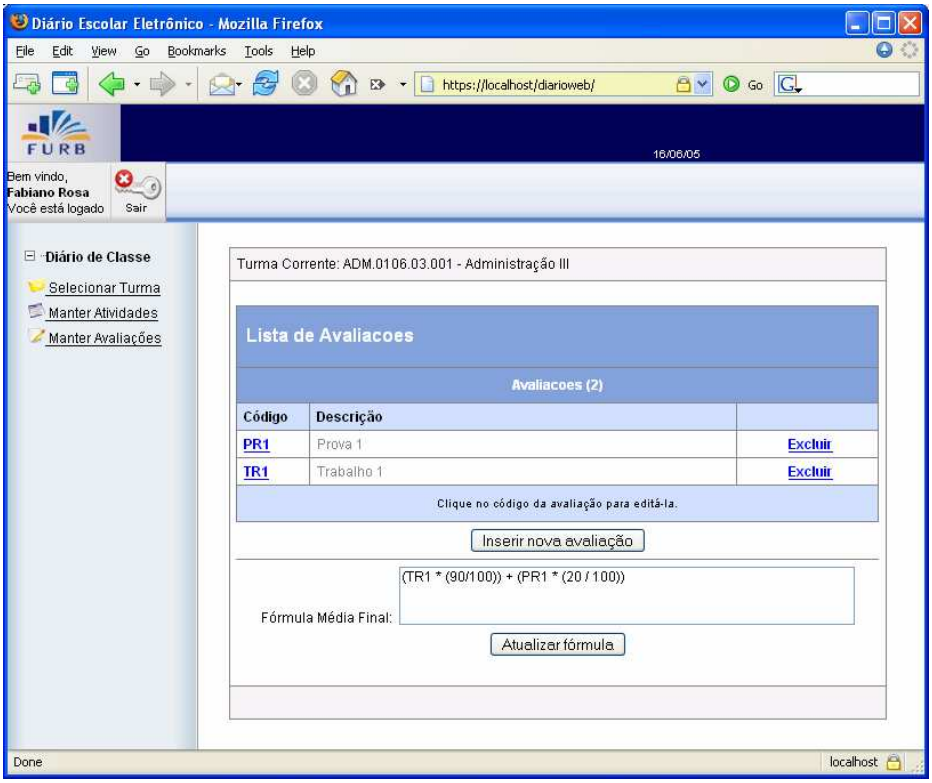


Figura 45 – Lista de avaliações com a fórmula para o cálculo da média final

A seguir, a Figura 46 demonstra a tela com as avaliações e suas notas e a média final do aluno.

Figura 46 – Tela exibindo a média final e as notas das avaliações do aluno

3.5 RESULTADOS E DISCUSSÃO

A etapa de testes e validação dos requisitos deste trabalho avaliou aspectos de desempenho, portabilidade, viabilidade e comunicação na Internet do software. Para a aplicação em J2ME, foram feitas simulações nos emuladores que vem com o *J2ME Wireless Toolkit* da Sun Microsystems e em emuladores dos celulares da Motorola, fornecidos pela mesma, onde foram identificadas apenas diferenças na exibição das telas no display dos emuladores, devido às dimensões da área de visualização da aplicação nos emuladores não serem as mesmas. A comunicação com o *Web Service* no emulador foi realizada com sucesso, mas se percebeu uma pequena baixa no desempenho na troca de mensagens entre a aplicação J2ME e o Web Service. Assim, foram feitos ajustes e otimizações na aplicação para reduzir a quantidade de informações que são enviadas e recebidas, e reduzir o número de mensagens trocadas entre a aplicação J2ME e o *Web Service*, implementando controles que verificam a necessidade de atualização das informações na aplicação do dispositivo móvel.

Para validar a portabilidade de uma aplicação J2ME, que é garantida pela especificação do perfil MIDP e a configuração CLDC, foi executada a aplicação do diário de classe em um dispositivo celular da Motorola, modelo C650, com o perfil MIDP 2.0 e configuração CLDC 1.0. A execução da aplicação J2ME no celular da Motorola só foi obtida sem a comunicação com o *Web Service*, onde foi alteranda a aplicação para operar com dados estáticos, devido a alguns contratempos com a configuração do acesso à rede do celular. Foram percebidas apenas algumas diferenças na exibição dos componentes visuais fornecidos pela J2ME, como caixas de texto, mas estas diferenças não prejudicaram a funcionalidade da aplicação no celular.

Como um dos objetivos era que a aplicação J2EE obtivesse as informações de um sistema acadêmico, foi criada uma base de dados em um banco ORACLE, de acordo com o

modelo conceitual, para simular a integração da classe fachada com os dados acadêmicos. Assim, foi possível validar a chamada dos métodos de negócio do EJB e do *Web Service* a partir de um *browser* e da aplicação J2ME executada no emulador.

Com todos estes testes, foi possível validar os requisitos funcionais e não-funcionais propostos no trabalho com sucesso, permitindo também que fossem realizados ajustes e otimizações na aplicação.

O Quadro 14 relaciona os trabalhos correlatos mencionados na fundamentação teórica com a aplicação desenvolvida, evidenciando aspectos e características em cada uma delas.

Funcionalidades	**Este Projeto**	**Schaefer (2004)**	**Schmitt Junior (2004)**	**Depiné (2002)**
Implementação com J2ME	x	x	x	x
Utilização de XML com o protocolo HTTP	x		x	
Meio de comunicação confidencial e seguro	x			
Troca de informações com um servidor	x	x	x	
Projeto orientado a objetos	x		x	
Comunicação com Web Services	x			
Aplicação de Design Patterns no projeto	x		x	
Armazenamento das informações do celular			x	x
Servlets com componentes no servidor			x	
EJBs como componentes no servidor	x			

Quadro 14 – Funcionalidades específicas de cada trabalho.

4 CONCLUSÕES

Este trabalho demonstra como é possível e viável a utilização de um dispositivo celular por professores de uma universidade como ferramenta para facilitar e informatizar o gerenciamento e monitoramento das atividades escolares dos alunos, possibilitando o acompanhamento da freqüência e das notas dos alunos, e também o registro e acompanhamento de suas próprias atividades realizadas em sala de aula.

Os objetivos do trabalho proposto foram atendidos com sucesso, assim como os requisitos funcionais e não funcionais. A execução da aplicação em um dispositivo celular que implementa a especificação J2ME foi atendida, não só nos emuladores como também em um dispositivo celular real, desde que este fosse compatível com a versão do perfil e de configuração do J2ME.

A utilização de uma EJB como componente de negócio disponibilizado em forma de *Web Service*, permitiu que a troca de informações entre a aplicação J2ME executada no dispositivo móvel e a aplicação J2EE executada no servidor, fosse realizada sem nenhuma dependência de protocolos de comunicação proprietários, mas sim, utilizando os padrões abertos de mercado, o XML e o SOAP.

Um dos principais requisitos não funcionais, a segurança no meio de comunicação, foi implementado com sucesso através da utilização de um certificado digital no servidor e pela autenticação adicionada ao *Web Service*, que permitiu que suas operações não ficassem expostas na Internet.

A utilização do protocolo SOAP com XML possibilitou que o sincronismo das informações entre as aplicações fosse feito de forma simples, devido à utilização de padrões de projeto, abstraindo da aplicação particularidades da utilização de um *Web Service*.

Foi executada a aplicação no dispositivo celular da Motorola, modelo C650, mas sem a

comunicação com o *Web Service*, devido à deficiência do suporte ao desenvolvimento de aplicativos Java em celulares, por parte do fabricante de celulares e das operadoras de telefonia móvel.

A seção de extensões sugere trabalhos que aprimorariam e enriqueceriam a solução de um diário de classe para dispositivos móveis.

4.1 EXTENSÕES

Nesta seção são apresentadas sugestões de extensões e modificações para este trabalho, que estão descritas a seguir:

a) adaptar a classe fachada da aplicação J2ME para que, ao tentar enviar dados ao *Web Service*, em caso de falha, as informações sejam persistidas no dispositivo móvel via API do J2ME, e posteriormente, estas informações sejam sincronizadas com o servidor;

b) permitir que quando o professor estiver consultando os detalhes do aluno na aplicação J2ME, esteja disponível no menu de opções outras funcionalidades relacionadas ao aluno, como o registro da sua freqüência e suas avaliações;

c) permitir que as funções para registrar as freqüências e as avaliações também possam ser feitas em uma página na Web;

d) adicionar uma validação na manutenção das avaliações para que, ao ser modificado um código de uma avaliação ou quando for excluída uma avaliação, seja invalidada a fórmula do cálculo da média final cadastrada para a turma;

e) pesquisar ou elaborar alguma técnica de compressão para as mensagens SOAP, com o objetivo de diminuir o tamanho do pacote de dados trafegado;

f) efetuar a validação do formato da nota de cada avaliação no dispositivo móvel,

levando em consideração as casas decimais;

g) modificar a forma da configuração do cálculo da média final, permitindo ao professor compor a fórmula de forma gráfica ou com assistentes, tornando mais amigável ao mesmo esta funcionalidade;

h) possibilitar ao professor alterar a média final do aluno no dispositivo celular.

REFERÊNCIAS BIBLIOGRÁFICAS

ALMEIDA, Leandro Batista de. et al. Introdução à J2ME e programação MIDP**. Mundo Java,** Rio de Janeiro, n. 5, p. 20-27, 2004.

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

BOND, Martin. **Aprenda J2EE:** com EJB, JSP, Servlets, JNDI, JDBC e XML**.** São Paulo: Pearson Education, 2003. xxxv, 962p.

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. **UML:** guia do usuário : o mais avançado tutorial sobre Unified Modeling Language (UML)**.** Rio de Janeiro: Campus, 2000. xx, 472p.

CORBERA, Rodrigo Garcia. **Tutorial de programação J2ME.** [S.l.], [2003]. Disponível em: <http://geocities.yahoo.com.br/brasilwireless/tutorial_j2me/j2me_01/j2me_01.html>. Acesso em: 25 mar. 2005.

D´IPPOLITO, Leonardo Chagas. **Mecanismo para licenciamento de aplicativos Microsoft.Net baseado em assinatura digital XML**. 2004. 114 p. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

DEPINÉ, Fabio Marcelo. **Protótipo de software para dispositivos móveis utilizando J2ME para cálculo de regularidade em rally**. 2002. 65 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

FONSECA, Jorge Cavalcanti. **Portando a KVM**. 2002. 64 f. Trabalho de Conclusão de Curso (Graduação em Ciências da Computação) – Centro de Informática, Universidade Federal de Pernambuco, Recife.

FURB**. Regimento Geral da Universidade Regional de Blumenau**, 2001. Disponível em: <http://www.furb.br/2005/arquivos/867631-594922/regimento%20geral%20da%20universidade%20-%20resolucao129-2001.htm/>. Acesso em: 28 maio. 2005.

GAMMA, Erich. **Padrões de projeto:** soluções reutilizáveis de software orientado a objetos**.** Porto Alegre: Bookman, 2000. xii, 364p.

GUMZ, Rafael Araújo. **Protótipo de um sistema gerador de interfaces gráficas para testes de Java Web Services**. 2005. 98 f. Monografia de Pós-Graduação (Pós-Graduação em Desenvolvimento de Aplicações para WEB) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

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

LARMAN, Craig. **Utilizando UML e padrões:** uma introdução à análise e ao projeto orientados a objetos**.** 2. ed. Porto Alegre: Bookman, 2004. 608 p, il. Tradução de: Applying UML and Patterns : an Introduction to Object-Oriented Analysis and Design.

LAUDON, Kenneth C.; LAUDON, Jane Price. **Sistemas de informação**. Tradução Dalton Conde de Alencar. Rio de Janeiro: LTC, 1999.

RYDLEWSKI, Carlos et al. A vida sem fio**. Veja,** São Paulo, v. 1874, n. 40, p. 101-110, out. 2004.

SCHAEFER, Carine. **Protótipo de aplicativo para transmissão de dados a partir de dispositivos móveis aplicado a uma empresa de transportes**. 2004. 53 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

SCHMITT JUNIOR, Arno José. **Protótipo de front end de controle de acesso usando J2ME**. 2004. 69 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

SPARX SYSTEMS. **Enterprise Architect**. [S.l.], 2004. Disponível em <http://www.sparxsystems.com.au/>. Acesso em: 07 jun. 2005.

SUN MICROSYSTEMS. **Java technology**. [S.l.], 2004a. Disponível em: <http://java.sun.com/>. Acesso em: 02 maio. 2005.

_____. **Java 2 platform, micro edition**. [S.l.], out. 2004b. Disponível em: <http://java.sun.com/j2me/>. Acesso em: 02 maio. 2005.

_____. **What's New in MIDP 2.0**. [S.l.], out. 2004c. Disponível em: <http://developers.sun.com/techtopics/mobility/midp/articles/midp20/>. Acesso em: 08 julho. 2005.

APÊNDICE A – Utilização do XDoclet para geração de artefatos para o EJB e o Web Service

Utilização do XDoclet na classe “DiarioEletronicoBean”, para a geração das classes *home* e *remote* do EJB, do descritor e de configurações de segurança do Web Services.

```
/**
 * The DiarioEletronico Session Bean this time as a web-service.
 *
 * @author Fabiano Rosa
 * @version $Revision: 1.1 $
 *
 * @ejb:bean
 *      name="DiarioEletronico"
 *      display-name="Diario Eletronico Bean"
 *      type="Stateless"
 *      view-type="remote"
 *      jndi-name="ejb/DiarioEletronico"
 *
 * @ejb:interface
 *      remote-
class="br.furb.diarioeletronico.web.controller.ejb.DiarioEletronicoRemote"
 *      extends="javax.ejb.EJBObject"
 *
 * @ejb:home
 *      remote-
class="br.furb.diarioeletronico.web.controller.ejb.DiarioEletronicoHome"
 *      extends="javax.ejb.EJBHome"
 *
 * @jboss-net:web-service
 *      urn="DiarioEletronico"
 *
 * @jboss-net.authentication
 *      securityDomain="java:/jaas/mobile"
 * @jboss-net.authorization
 *     securityDomain="java:/jaas/mobile"
 *      roles-allowed="*"
 */
public class DiarioEletronicoBean extends BaseSession
       implements SessionBean {
     // IMPLEMENTAÇÃO
}
```

APÊNDICE B – Implementação de um método utilizando a biblioteca kSOAP2

Implementação de um método da classe “DiarioEletronicoFacade” da aplicação J2ME utilizando as classes da biblioteca *kSOAP2*.

```
public Hashtable getListaTurmas(String codVinculoProfessor)
      throws Exception {

      SoapObject method = new SoapObject("urn:soap-call",
            "getListaTurmas");
      method.addProperty("codVinculoProfessor", codVinculoProfessor);
      SoapSerializationEnvelope envelope = new
            SoapSerializationEnvelope(SoapEnvelope.VER10);
      envelope.bodyOut = method;

      http.call(null, envelope);

      SoapObject result = (SoapObject) envelope.getResult();

      Vector vtcodTurmas = (Vector)result.getProperty(0);
      String[] codTurmas = this.getStringArrayFromVector(vtcodTurmas);

      Vector vtNomTurmas = (Vector)result.getProperty(1);
      String[] nomTurmas = this.getStringArrayFromVector(vtNomTurmas);

      Hashtable ht = new Hashtable(2);
      ht.put(DiarioEletronicoFacade.LISTA_COD_TURMA, codTurmas);
      ht.put(DiarioEletronicoFacade.LISTA_NOM_TURMA, nomTurmas);

      return ht;
}
```

APÊNDICE C – Implementação do método que utiliza HTTPs e autenticação Basic

Implementação do método da classe “HttpTransport” da biblioteca *kSOAP2* para efetuar a comunicação via HTTPs e adicionar a autenticação *Basic*.

```
public void call(String soapAction, SoapEnvelope envelope)
      throws Exception, IOException, XmlPullParserException {
    if (soapAction == null)
        soapAction = "\"\"";

    ByteArrayOutputStream bos = new ByteArrayOutputStream();
    XmlSerializer xw = new KXmlSerializer();
    xw.setOutput(bos, null);
    envelope.write(xw);
    xw.flush();
    bos.write('\r');
    bos.write('\n');
    byte[] requestData = bos.toByteArray();
    bos = null;
    xw = null;

    requestDump = debug ? new String(requestData) : null;
    responseDump = null;

    try {
        connected = true;

        // classe HttpsConnection para
        // comunicação que utiliza certificado digital
        connection =
            (HttpsConnection) Connector.open(
                url,
                Connector.READ_WRITE,
                true);

        connection.setRequestProperty("SOAPAction", soapAction);
        connection.setRequestProperty("Content-Type", "text/xml");
        connection.setRequestProperty("Content-Length", "" +
            requestData.length);

        connection.setRequestProperty("User-Agent", "kSOAP/2.0");

        // Adiciona a autenticação BASIC
        if (this.indAuthorization) {
            String userPassword = this.user + ":" + this.password;
            String encoding = Base64.encode(userPassword.getBytes());
            connection.setRequestProperty("Authorization","Basic " +
                  encoding);
        }

        connection.setRequestMethod(HttpConnection.POST);

        os = connection.openOutputStream();
        os.write(requestData, 0, requestData.length);
        //os.flush ();  // removed in order to avoid chunked encoding
        os.close();
```

```
            requestData = null;

            is = connection.openInputStream();

            XmlPullParser xp = new KXmlParser();
            xp.setFeature(XmlPullParser.FEATURE_PROCESS_NAMESPACES, true);
            xp.setInput (is, null);

            envelope.parse(xp);
        } finally {
            if (!connected) {
                throw new InterruptedIOException();
            }
            reset();
        }

        if(envelope.bodyIn instanceof SoapFault) {
            throw ((SoapFault) envelope.bodyIn);
        }
    }
```

APÊNDICE D – Especificação da entrada para o gerador léxico e sintático GALS

Definição da entrada para o gerador GALS para a geração das classes Java responsáveis pela analise léxica e sintática da fórmula da média final.

```
#RegularDefinitions
digito : [0-9]
letra  : [a-z A-Z]

#Tokens
// Simbolos especiais
"+"
"-"
"*"
"/"
"("
")"

// tokens
identificador: {letra} ({letra}|{digito})*
numero: {digito}+ ("," {digito}+)?

//ignorar
: [\s\t\n\r]*

#NonTerminals
<expressao>
<expressaoLinha>
<termo>
<termoLinha>
<fator>

#Grammar
<expressao>::= <termo> <expressaoLinha> ;
<expressaoLinha>::= î | "+" <termo> #1 <expressaoLinha> | "-" <termo> #2
<expressaoLinha> ;

<termo>::= <fator> <termoLinha> ;
<termoLinha>::= î | "*" <fator> #3 <termoLinha> | "/" <fator> #4
<termoLinha>;

<fator>::= "(" <expressao> ")" | identificador #5 | numero #6 ;
```

APÊNDICE E – Implementação do padrão *Singleton* e *Facade* no trabalho

Padrão *Singleton* e *Facade* aplicados em conjunto na classe “DiarioEletronicoFacade” para representar a fachada com o sistema acadêmico.

```
// Classe facade para acessar os dados do sistema acadêmico
public class DiarioEletronicoFacade {

    private static DiarioEletronicoFacade instance;

    private DiarioEletronicoFacade() {
    }

    public static DiarioEletronicoFacade getInstance() {
        if (instance == null) {
            instance = new DiarioEletronicoFacade();
        }
        return instance;
    }

    // Métodos para a aplicação J2ME
    public Vector getListaTurmas(String codVinculoProfessor)
      throws Exception {
        // IMPLEMENTAÇÃO
    }

    public Vector getListaAlunos(String codTurma) throws Exception {
        // IMPLEMENTAÇÃO
    }

    public Vector getDetailsAluno(String codVinculoAluno)
      throws Exception {
        // IMPLEMENTAÇÃO
    }

    public String[] getListaMatriculaAluno(String codVinculoAluno)
      throws Exception {
        // IMPLEMENTAÇÃO
    }

    public Vector getInfAusenciaAluno(String codTurma,
      String codVinculoAluno) throws Exception {
        // IMPLEMENTAÇÃO
    }

    public Vector getMediaFinalAluno(String codTurma, String
      codVinculoAluno) throws Exception {
        // IMPLEMENTAÇÃO
    }

    private String calculaMediaFinal (String formulaMediaFinal,
      String[] codAvaliacoes, String[] notaAvaliacoes) throws Exception {
        // IMPLEMENTAÇÃO
    }

     public Vector getListaAvaliacoesTurma(String codTurma)
```

```
  throws Exception {
    // IMPLEMENTAÇÃO
}

public Vector getNotasAvaliacoesAlunos(String codTurma, String codAval)
  throws Exception {
    // IMPLEMENTAÇÃO
}

public Vector getFrequenciaAlunos(String codTurma, String idAula)
  throws Exception {
    // IMPLEMENTAÇÃO
}

public Vector getListaDatasAulas(String codTurma, Long dataAtual)
  throws Exception {
    // IMPLEMENTAÇÃO
}

public void registrarFrequenciaAlunos (String idAula,
  Vector codVinculoAlunos, Vector indComparecimentos)
  throws Exception {
    // IMPLEMENTAÇÃO
}

public void registrarAvaliacaoAlunos (String idAvaliacao,
  Vector codVinculoAlunos, Vector notas) throws Exception {
    // IMPLEMENTAÇÃO
}

public Vector getListaAtividades(String idAula) throws Exception {
    // IMPLEMENTAÇÃO
}

public void registrarAtividade(String codAtividade, String observacao)
  throws Exception {
    // IMPLEMENTAÇÃO
}

public String getObservacaoAtividade(String codAtividade)
  throws Exception {
    // IMPLEMENTAÇÃO
}

public String getNomeProfessor(String codigoVinculo) throws Exception {
    // IMPLEMENTAÇÃO
}

// Métodos para a aplicação WEB
public ArrayList getArrayListTurmas(String codVinculoProfessor)
  throws Exception {
    // IMPLEMENTAÇÃO
}

public ArrayList getArrayListAlunos(String codTurma) throws Exception {
    // IMPLEMENTAÇÃO
}

public ArrayList getArrayListDatasAulas(String codTurma,
  Long dataAtual) throws Exception {
    // IMPLEMENTAÇÃO
```

```
    }

    public void insereAvaliacao (String codigo, String descricao,
      String codTurma) throws Exception {
         // IMPLEMENTAÇÃO
    }

    public void excluiAvaliacao (String idAvaliacao) throws Exception {
        // IMPLEMENTAÇÃO
    }

    public void alteraAvaliacao (String idAvaliacao, String codigo,
      String descricao, String codTurma) throws Exception {
        // IMPLEMENTAÇÃO
    }

    public HashMap getDadosAvaliacao (String idAvaliacao)
      throws Exception {
        // IMPLEMENTAÇÃO
    }

    public ArrayList getArrayListAvaliacoes(String codTurma)
      throws Exception {
        // IMPLEMENTAÇÃO
    }

    public void insereAtividade (String dscAtividade, String obsAtividade,
      String idAula) throws Exception {
        // IMPLEMENTAÇÃO
    }

    public void excluiAtividade (String idAtividade) throws Exception {
        // IMPLEMENTAÇÃO
    }

    public void alteraAtividade (String idAtividade, String dscAtividade,
      String obsAtividade, String idAula) throws Exception {
        // IMPLEMENTAÇÃO
    }

    public HashMap getDadosAtividade (String idAtividade)
      throws Exception {
        // IMPLEMENTAÇÃO
    }

    public ArrayList getArrayListAtividades(String idAula)
      throws Exception {
        // IMPLEMENTAÇÃO
    }

    public void atualizaFormulaMediaFinal (String codTurma,
      String formulaMediaFinal) throws Exception {
        // IMPLEMENTAÇÃO
    }

    public String getFormulaMediaFinal (String codTurma) throws Exception {
        // IMPLEMENTAÇÃO
    }
}
```