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

DIOGO GUERRO
EDENILSON TONDO DA SILVA

INTEGRAÇÃO ENTRE SOFTWARE JAVA SE E JAVA ME UTILIZANDO BLUETOOTH COM FRAMEWORK MARGE

TRABALHO DE CONCLUSÃO DE CURSO

PATO BRANCO
2013

DIOGO GUERRO
EDENILSON TONDO DA SILVA

INTEGRAÇÃO ENTRE SOFTWARE JAVA SE E JAVA ME UTILIZANDO BLUETOOTH COM FRAMEWORK MARGE

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

Orientador: Prof. Géri Natalino Dutra.
Co-Orientador: Prof. Robison Cris Brito

PATO BRANCO
2013

ATA Nº: **211**

DEFESA PÚBLICA DO TRABALHO DE DIPLOMAÇÃO DOS ALUNOS **EDENILSON TONDO DA SILVA** e **DIOGO GUERRO**.

Às 16:00 hrs do dia 19 de abril de 2013, Bloco V da UTFPR, Câmpus Pato Branco, reuniu-se a banca avaliadora composta pelos professores Géri Natalino Dutra (Orientador), Robison Cris Brito (Convidado) e Luís Carlos Ferreira Bueno (Convidado), para avaliar o Trabalho de Diplomação do aluno Edenilson Tondo da Silva, matrícula 657069 e do aluno Diogo Guerro, matrícula 980374, sob o título **Integração de Software de Academia para Dispositivos Móveis com Servidor Desktop Utilizando Bluetooth**; como requisito final para a conclusão da disciplina Trabalho de Diplomação do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, COADS. Após a apresentação os candidatos foram entrevistados pela banca examinadora, e a palavra foi aberta ao público. Em seguida, a banca reuniu-se para deliberar considerando o trabalho **APROVADO**. Às 16:55 hrs foi encerrada a sessão.

Prof. Géri Natalino Dutra, M.Sc.
Orientador

Prof. Robison Cris Brito, M.Sc.
Convidado

Prof. Luís Carlos Ferreira Bueno, M.Sc.
Convidado

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

Prof. Edilson Pontarolo, Dr.
Coordenador do Curso

Dedicamos este trabalho aos nossos pais, professores e aos nossos amigos companheiros, que do início ao fim estiveram ao nosso lado.

AGRADECIMENTOS DE DIOGO GUERRO

Agradeço e dedico este trabalho a minha família e a todos que estiveram comigo, que me ajudaram direta ou indiretamente a chegar ao fim desta jornada. Agradeço em especial aos professores orientadores Géri Dutra e Robison Brito, e aos amigos Edenilson da Silva, Fernando Meurer e James Rebelato.

AGRADECIMENTOS DE EDENILSON TONDO DA SILVA

Quando ingressei no curso, por ingenuidade, acreditei que a parte mais difícil havia sido completada, que a partir daquele momento estaria livre de estudar matérias de que não gostava, pois já tinha passado por esta etapa necessária, e que agora, estaria somente atuando na área que havia escolhido. Doce ilusão, logo no início pude perceber que mesmo dentro daquilo que eu gostava tanto, existiam matérias e assuntos que eu não simpatizava, e foi dentro da faculdade que percebi que a vida é assim, podemos escolher a estrada que mais nos agrada. Pode ser o caminho mais belo e fácil de seguir, mas mesmo esse, sempre terá uma pedra no caminho, uma dificuldade a ser vencida, e quando aceitamos isso, o caminho fica mais fácil de trilhar, pois já não nos deixamos abater pela pedra, a aceitamos como parte do caminho. Principalmente a isso sou grato, agradeço ao curso, por ter me ensinado aquilo que este não imaginava ensinar, todo o resto que aprendi (que não é pouco) perto desta lição que levo, é pouco.

Agradeço a minha família, principalmente aos meus pais, pelo esforço que fizeram para que eu conseguisse completar essa jornada, com certeza sem o apoio deles eu não teria chegado nem no começo desta longa caminhada. Aos meus irmãos que esta etapa da minha vida sirva de exemplo; que não desistam de seus projetos e que não os deixem para o último momento.

Aos meus amigos e colegas do curso, muito obrigado pela ajuda de vocês, em especial, ao Fernando Meurer, Diogo Guerro e James Rebelato, sem o companheirismo e amizade de vocês, esta conquista não teria valor.

A minha namorada e companheira, Maiara de Ávila, obrigado pelo apoio e compreensão, sei que muitas vezes te deixei de lado, principalmente para o desenvolvimento deste trabalho. Pelo seu suporte e carinho nesta etapa, meu muito obrigado.

RESUMO

GUERRO, Diogo; SILVA, Edenilson Tondo da. **Integração entre Software Java SE e Java ME Utilizando Bluetooth com Framework Marge**. 75 f. Monografia de Trabalho de Conclusão de Curso - Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, Universidade Tecnológica Federal do Paraná. Pato Branco, 2013.

Uma grande parcela da população preocupa-se com sua qualidade de vida, infelizmente, a prática de atividades físicas nos meios urbanos é limitada, o que faz com que essas pessoas procurem academias para a prática de exercícios físicos, buscando como resultados fortalecer os músculos, diminuir a gordura corporal, aumentar a massa muscular, entre outros benefícios. Em algumas academias são feitas avaliações periodicamente para mensurar a evolução do aluno, motivando este a continuar o treinamento. Porém, este controle é feito de forma manuscrita, pouco prática e muitas vezes o maior interessado, o aluno, não pode desfrutar de um acompanhamento efetivo de sua evolução. O presente trabalho relata a integração e aperfeiçoamento de um sistema de controle de medidas para atletas, sejam eles profissionais ou amadores em busca de uma melhor qualidade de vida ou até mesmo proprietários de academia, que pretendam ofertar e acompanhar dados sobre o desempenho de seus alunos. Para o aperfeiçoamento do aplicativo para dispositivos móveis foi utilizada a tecnologia Java ME, a qual permite desenvolver softwares para celulares. Para o desenvolvimento da aplicação *desktop*, a tecnologia utilizada foi Java, que permite a comunicação com a aplicação celular através da tecnologia *Bluetooth* com framework Marge, para fins de integração entre os softwares.

Palavras-chave: Integração, *Bluetooth*, Celular, Qualidade de Vida, Java SE, Java ME, Marge.

ABSTRACT

GUERRO, Diogo; SILVA, Edenilson Tondo da. **Software Integration Between Java SE and Java ME Using Bluetooth With Framework Marge.** 75 f. Monografia de Trabalho de Conclusão de Curso - Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, Universidade Tecnológica Federal do Paraná, Campus Pato Branco. Pato Branco, 2013.

A large portion of the population is concerned about their quality of life, but unfortunately the practice of physical activity in urban areas is limited, which makes these people look for gyms for physical exercise, seeking, as a result of their activity, to strengthen the muscles, decrease body fat, increase muscle mass, among other benefits. In some gyms, assessments are made periodically to measure the progress of members, motivating them to continue their training. However, this control is done in a handwritten form, impractical, and often the most interested part, the member, cannot enjoy an effective monitoring of his evolution. This paper describes the integration and improvement of a system to evaluate one's performance and measures, whether they be professional or amateur, looking for a better quality of life, or even gym owners who wish to submit and track data about the performance of their students. For application improvement for mobile devices Java ME technology was used, which allows developing software for mobile phones. For desktop application development, the technology used was Java, which allows communication with mobile application through Bluetooth technology with framework Marge, used for purposes of integration between softwares.

Keywords: Integration, Bluetooth, Mobile Phone, Quality of Life, Java SE, Java ME, Marge.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE SIGLAS

ANATEL	Agência Nacional de Telefonia
API	Application Programming Interface
BT-WPAN	Bluetooth Wireless Personal Network
CLDC	Connected Limited Device Configuration
FCC	Federal Communication Commission
GNU/LINUX	General Public License/Linux
GPS	Global Positioning System
IDE	Integrated Development Environment
IBGE	Instituto Brasileiro de Geografia e Estatística
IEEE	Institute of Electrical and Electronic Engineers
JAVA EE	Java Enterprise Edition
JAVA ME	Java Micro Edition
JAVA SE	Java Standard Edition
JDBC	Java Database Connectivity
JVM	Java Virtual Machine
JSR	Java Specification Request
LED	Light Emitting Diodes
LGPL	Lesser General Public License
UML	Linguagem de Modelagem Unificada.
OO	Orientação a Objeto
PAN	Personal Area Network
SGDB	Sistema de Gerenciamento de Banco de Dados
SQL	Structured Query Language
VM	Virtual Machine
TI	Tecnologia da Informação

SUMÁRIO

1 INTRODUÇÃO

Este capítulo apresenta as considerações iniciais apresentando uma visão geral do trabalho, os objetivos, a justificativa e a organização do texto.

1.1 CONSIDERAÇÕES INICIAIS

A Organização Mundial da Saúde estima que em todo o mundo mais de 2 milhões de mortes são atribuídas à inatividade física (OMS, 2002).

Os principais efeitos causados à saúde pelo sedentarismo, como consequência do excessivo uso do computador são: as dores nos joelhos, dores de cabeça, cansaço mental, dores na coluna, tendinite, preguiça, estresse, colesterol alto, peso acima dos parâmetros ideais, entre outros. Em geral, somente a partir do momento em que sintomas como os citados começam a aparecer, é que estas pessoas passam a preocupar-se com sua qualidade de vida. (Saudevidaonline.com, 2011).

Quando alguns sintomas aparecem, a prática de exercício físico passa a ser vista como prioridade. A partir deste momento, os interessados partem em busca de fortalecimento e aumento muscular, além de diminuição do colesterol. São feitas avaliações físicas periódicas para mensurar a evolução do aluno em algumas academias, contudo, o controle é feito, em geral, de forma manuscrita e pouco prática, dificultando o acesso do aluno a algo que lhe é de grande interesse.

De modo a motivar a prática de exercícios físicos e possibilitar que o praticante de atividade física e o seu instrutor possam ter um acompanhamento efetivo da evolução do progresso do praticante, faz-se oportuno o aperfeiçoamento de um sistema de controle de medidas para atletas, desenvolvido anteriormente no estágio do acadêmico Diogo Guerro.

De acordo com Guerro (2011), o sistema foi desenvolvido utilizando tecnologia móvel Java ME podendo ser acessado a partir de um aparelho celular. Assim que acessado, o sistema oferece as opções de cadastro de um novo usuário para o controle de medidas, a possibilidade de lançar novas medidas e também a visualização de informações já lançadas, com estatísticas.

Neste trabalho, objetivou-se aperfeiçoar tal sistema, de modo que ele possa

ser usado tanto pelo celular, para uso do atleta, quanto por uma aplicação *desktop*, a ser usada pelo instrutor, pois deste modo, tanto o atleta poderá ter um acompanhamento de seu desempenho e progresso na prática de atividade física, como também, o instrutor poderá, através do histórico do aluno, acompanhar o desenvolvimento do mesmo, e orientá-lo durante seu progresso, fazendo as alterações necessárias para que melhores resultados possam ser alcançados.

1.2 OBJETIVOS

Na seção a seguir serão apresentados: o objetivo geral e os objetivos específicos do sistema proposto.

1.2.1 Objetivo Geral

Aprimorar um sistema objetivando auxiliar os instrutores de academias a disponibilizarem aos seus alunos um histórico prático e simples do avanço dos mesmos na prática de exercícios, de modo que eles possam visualizar estas informações também em seu celular.

1.2.2 Objetivos Específicos

- Apresentar pesquisas acadêmicas e científicas de referência sobre a tecnologia Java ME;
- Utilizar padrões de desenvolvimento Orientados a objetos com a tecnologia Java SE;
- Realizar análise do sistema *desktop* de controle de medidas;
- Aperfeiçoar o aplicativo para celular e desenvolver o aplicativo para *desktop*;
- Trabalhar com a comunicação de dados *Bluetooth* para integrar a plataforma do celular com a do *desktop*.

1.3 JUSTIFICATIVA

O aperfeiçoamento do sistema de controle de medidas para frequentadores de academias justifica-se pela razão de tornar o aplicativo mais intuitivo e útil, devido às novas funcionalidades que visam incentivar seu uso.

Além disso, o desenvolvimento de um aplicativo para *desktop*, integrando o celular a um servidor, complementa o aplicativo e proporciona um controle automatizado para a academia, isso significa que o aluno poderá acesso fácil, rápido e cômodo ao seu histórico de medidas.

O instrutor da academia também terá acesso ao histórico dos seus alunos de maneira prática, podendo mensurar os resultados alcançados por seus alunos e possibilitando, com base nesses dados, adaptar os treinamentos.

1.4 ESTRUTURA DO TRABALHO

O trabalho está organizado em seis capítulos, sendo este o primeiro. Ele contém as considerações iniciais, a contextualização, os objetivos e a justificativa do trabalho. O restante do documento está organizado da seguinte forma:

O capítulo 2 apresenta a fundamentação teórica do trabalho, que inclui uma breve descrição a respeito de musculação, aparelhos celulares, tecnologia *Bluetooth*, orientação a objeto e modelagem de sistemas.

No capítulo 3 os materiais e métodos utilizados foram demonstrados e especificados.

No capítulo 4, são apresentados os resultados do sistema e a modelagem deste.

O capítulo 5 apresenta as conclusões obtidas, além de algumas questões em aberto e o capítulo 6 apresenta as referências bibliográficas consultadas.

2 REFERENCIAL TEÓRICO

Este capítulo apresenta um pouco da história da musculação e das academias, o controle de medidas feito nelas, as vantagens de se utilizar aplicações em aparelhos celulares e orientação a objeto.

2.1 MUSCULAÇÃO, ATIVIDADES FÍSICAS E ACADEMIAS DE GINÁSTICA

Segundo STEINHILBER (1996), a história da musculação e de exercícios para condicionamento físico mistura-se com o surgimento do próprio ser humano, pois desde que o homem colocou-se de pé ele executa os movimentos corporais mais básicos e naturais; corre, salta, arremessa, puxa, empurra, entre outros. Essas ações são respostas às necessidades sentidas pelo homem primitivo, que são de fugir, lutar, caçar, além de diversas outras atividades vitais para a sua sobrevivência. O homem antigo percebeu a importância de ter um físico que lhe ajudasse a realizar estas tarefas, isso despertou o interesse de cultivar um corpo mais forte.

Durante escavações realizadas na cidade de Olímpia, na Grécia, encontraram-se pedras com entalhes para as mãos, o que leva pesquisadores a acreditarem que serviam como anilhas. No Egito, em paredes de capelas funerárias existem gravuras mostrando que desde o ano de 4.500 a.C. homens já levantavam peso como forma de exercício físico.

A musculação teve vários registros na história, por vários locais do mundo e em diversas épocas, um dos registros mais antigos desta prática foi feito por Pausânias em seu livro “Descrição da Grécia”, nele é contada a história do atleta Milo de Crotona, nascido em torno de 560 a.C. e ganhador por 6 vezes consecutivas dos jogos olímpicos da Grécia antiga. Como forma de treinar seu corpo, Milo começou a carregar um bezerro para exercitar seus músculos, conforme o bezerro crescia e aumentava de peso, maior era o esforço necessário por Milo para carregá-lo, com este esforço crescente, o resultado de Milo foi aumento de sua força e massa muscular. Constata-se com esse relato, que as antigas civilizações possuíam conhecimentos de como aumentar sua força e tamanho muscular. Reza a lenda, que Milo deparou-se com um tronco de árvore dividido por uma fenda, testando sua força ele decidiu partí-lo, mas a fenda fechou-se, prendendo suas mãos e incapacitando-o

de defender-se contra um ataque de lobos, que o devoraram, conforme ilustrado na Figura 1, onde aparece a pintura feita por Joseph-Benoit Suvée.

Figura 1. Milo de Crotona
Fonte: Super.abril.com.br

O historiador americano Donald Kyle, da Universidade do Texas, em artigo escrito por Denis Russo Burgierman da Revista Super Interessante sobre esta lenda afirma: “Consta que Milo de Crotona morreu anos depois, foi devorado por lobos após sofrer um acidente ao quebrar um tronco com as próprias mãos... É difícil saber se a história é verdadeira. Tudo o que se sabe sobre ele vem de textos esparsos que cheiram à lenda. Talvez tenham exagerado a capacidade dos seus bíceps e do seu estômago. Mas não há dúvidas de que foi um herói do mundo grego, adorado como um semideus depois de morrer”.

De acordo com Capinussú & Costa (1989), Platão em 387 a.C. criou uma escola em homenageando o herói ateniense Academus, onde existia o ensino de práticas esportivas. Este local recebeu o nome de Akademia. Segundo artigo publicado por Alfredo Antunes, doutor em Educação Física/Ciência do Desporto Unicamp e Mestre em Ciência da Motricidade Humana em seu texto (disponível no endereço http://www.webartigos.com/artigos/academias-de-ginastica-e musculacao/29648/), as academias como são hoje, existem desde 1867 quando em

Bruxelas, Bélgica, uma instituição focou no ensino da cultura física com aparelhos. Após isso, novos estabelecimentos para prática de atividades físicas foram surgindo, primeiramente na França e em seguida nos Estados Unidos. Estas academias foram espalhando-se por todos os continentes e hoje são essenciais para a sociedade. No Brasil, a primeira academia surgiu em Belém do Pará, com a atividade de jiu-jitsu. Posteriormente, em 1925, no Rio de Janeiro, uma academia onde halterofilismo e ginástica olímpica eram ofertados foi inaugurada.

Ainda segundo Antunes, em seu artigo citado anteriormente, citando Pereira (1996), as academias originaram-se devido à necessidade de segurança: parques, ruas e praças tornavam-se perigosos, o grande crescimento populacional impedia a livre movimentação.

Praticada nas academias, a musculação é uma modalidade baseada no fisiculturismo. Sendo que a diferença entre a musculação e o fisiculturismo é o fato do primeiro ser uma atividade física, já o segundo é um esporte.

As variáveis de carga, amplitude, tempo de contração e velocidade controláveis apresentadas na musculação, proporcionam a sua prática por pessoas de várias idades e com diferentes objetivos.

A musculação diminui o percentual de gordura, deixando o corpo bonito e harmonioso, treina o coração para esforços mais intensos, aumenta a força, melhora os aspectos cognitivos, aumenta a resistência do sistema imunológico, melhora a postura, a flexibilidade e autoestima.

Os resultados obtidos com a musculação não dependem somente da maneira, tipo, ordem, frequência e intensidade dos exercícios. Outros fatores como flexibilidade, tipo de alimentação, hereditariedade e condicionamento cardiorrespiratório prévio são importantes.

Pesquisas indicam que o treinamento com pesos auxilia a emagrecer, atividades com pesos aumentam o gasto calórico diário e estimulam o metabolismo.

Periodicamente são feitas avaliações físicas, nas quais são medidas diversas partes do corpo com a finalidade de mensurar a evolução do atleta na redução do seu peso ou crescimento de um músculo que se dá a partir de atividades aeróbicas ou execução de determinados exercícios para cada um dos grupos musculares. Entre estas medidas estão: peso, braços contraídos, coxas, panturrilhas, quadril, peito e abdômen. Porém, na maioria das academias este processo é feito de forma manuscrita, e o acesso a estes números pelo maior interessado, que é o cliente da

academia, não é possível senão no momento da avaliação.

Para facilitar o processo de controle de medidas poderia ser utilizado um sistema informatizado, integrando um servidor *desktop* a um dispositivo móvel.

A maioria das pessoas, hoje, possui aparelhos celulares e estes, cada vez tem maior poder de processamento e armazenamento, o que possibilita a execução de programas específicos, como um programa para o armazenamento das medidas em uma academia.

2.2 APARELHOS CELULARES

Em meados da década de 40 já se pensava em tornar a comunicação mais eficiente, principalmente para o principal meio de comunicação existente à época: o telefone. Imaginava-se um sistema que fosse capaz de efetuar a comunicação entre telefones sem a necessidade de fios para conectar os aparelhos. Porém, a tecnologia da época não contribuía muito para execução da ideia. No trabalho de Abreu (2004), o autor descreve que o ano de 1947 foi o marco para o início da história dos celulares, com o desenvolvimento de um sistema que permitia a utilização de telefonia móvel dentro de uma determinada área utilizando o conceito de células, ou áreas de cobertura, derivando deste, o nome celular. Foi a empresa americana Bell Company que desenvolveu tal sistema. Ainda naquele ano, segundo Abreu (2004), nos Estados Unidos, a AT&T e a Bell propuseram à FCC (*Federal Communication Commission*) a alocação de um número de frequência de rádio especificamente para comunicação móvel, mas a FCC disponibilizou apenas poucas frequências, possibilitando que somente 23 pessoas se conectassem simultaneamente ao sistema de uma determinada área de cobertura. Isto, na época, tornou a tecnologia inviável comercialmente.

Passaram-se quase duas décadas para que a FCC reconsiderasse o número de frequências destinadas à telefonia móvel e aumentasse o número para suportar mais usuários. Na década de 70 foi definida a utilização de um sistema de torres para atender aos usuários por áreas, variando cada torre conforme seu deslocamento. Este sistema vigora até hoje na telefonia móvel mundial.

Ainda segundo Abreu (2004), foi em 1983 que surgiu o primeiro celular aprovado pelo FCC, o DynaTAC 8000X, da Motorola - que junto com a empresa

Ameritech iniciou o uso comercial da telefonia celular nos Estados Unidos e no mundo.

O aparelho mostrado na Figura 2, pesava cerca de 1 Kg, tinha capacidade para uma hora de conversação e oito horas com o aparelho em *stand-by*, memória para 30 números, além de display com LED (*Light Emitting Diodes*). As listas de espera para adquirir estes aparelhos chegavam aos milhares de interessados, mesmo com os preços chegando à casa dos quatro mil dólares. Atualmente, um aparelho de última geração, com diversas funcionalidades adicionais, pode ser adquirido por menos de 5% deste valor.

Figura 2. Motorola DynaTac 8000X
Fonte: Abreu (2004)

O celular tornou-se, neste contexto, uma extensão da personalidade do usuário, uma peça capaz de enriquecer relacionamentos, divertir, aumentar a produtividade e expressar individualidade. Isso significa comunicar, dividir, criar e divertir com voz, textos, imagens, músicas e vídeos. Com o barateamento da tecnologia, o número de usuários de celular no mundo passou de cerca de 300 mil, em 1984 para cerca de 4,6 bilhões atualmente, segundo o levantamento da União

Internacional de Telecomunicações, através do seu Setor de Desenvolvimento das Telecomunicações (UIT-D). Estes dados constam no relatório anual divulgado pela instituição e disponível no endereço web http://www.itu.int/dms_pub/itu-d/opb/ind/D-IND-ICTOI-2011-SUM-PDF-E.pdf.

Segundo informações da ANATEL (Agência Nacional de Telefonia)[1], o Brasil ultrapassou a marca de um celular por habitante. A superação dessa marca ocorreu em 31 de outubro de 2010, quando foram recebidos os dados sobre os celulares em operação no Brasil de outubro e dados do Instituto Brasileiro de Geografia e Estatística (IBGE) sobre a população brasileira: 194,439 milhões de celulares para uma população de 193,585 milhões de habitantes.

Os celulares agregaram, com o passar do tempo muitos recursos, tais como câmera, rádio FM e leitor MP3. Alguns telefones, inclusive, têm um computador de mão Palm ou PocketPC integrado - são os chamados *smartphones* (do inglês "*smart*", inteligente, "*phone*", telefone).

Sua principal característica está na persistência de dados, oferecendo possibilidade de instalar programas que utilizam os recursos disponíveis no aparelho. Os sistemas operacionais mais utilizados são Symbian, Windows Mobile e Android, tendo o Linux crescido também de forma muito rápida.

Nos dias de hoje, o celular já não é mais um simples telefone de bolso. Câmeras que possuem 14.1 Megapixel já foram lançadas na Europa, no Brasil encontram-se câmeras de até 12.1 Megapixel, resoluções maiores que muitas câmeras digitais. A internet já pode ser acessada via Wi-Fi e banda larga 3G e o 4G já está disponível em algumas capitais do Brasil (Brasília, Belo Horizonte, Fortaleza, Recife, Salvador e Rio). As telas que em 2003 possuíam 4 mil cores, agora chegam a mais de 16 milhões. As polegadas das telas dos celulares também cresceram. Atualmente as telas dos celulares têm em média 5 polegadas, mas fabricantes como a Samsung, já anunciaram celulares que as telas chegaram a 6.3 polegadas (Figura 3).

[1] Dados disponíveis na página da entidade, no endereço web http://www.anatel.gov.br/Portal/exibirPortalPaginaEspecialPesquisa.do?acao=&tipoConteudoHtml=1&codNoticia=21613.

Figura 3. Celular Samsung S-III com tela de 6.3 polegadas sensível ao toque
Fonte: http://www.samsungmobilepress.com/2013/04/11/Samsung-Introduces-the-GALAXY-Mega-1

O interesse no trabalho apresentado neste documento nos celulares é justificado pela tecnologia que é utilizada nestes dispositivos, que possuem a capacidade para funcionar com aplicações que suportam a linguagem Java e apresentam funcionalidades como conexões através de *Bluetooth*, itens de interesse para o desenvolvimento da pesquisa proposta, que visa integrar um software *desktop* com dispositivos móveis através do *Bluetooth*. No próximo item esta última tecnologia é explícita.

2.3 *BLUETOOTH*

Segundo Morimoto (2011), em seu livro Redes, Guia Prático, o *Bluetooth* é um padrão para redes PAN (*Personal Area Network*), isso significa que o *Bluetooth* é uma rede desenvolvida para curta distância, utilizada para interligar celulares, *tablets*, rádios e diversos outros dispositivos de uso pessoal. De acordo com Morimoto (2011), o *Bluethooth* funciona como um substituto para os cabos, ou seja, é uma tecnologia que permite interligar periféricos próximos, substituindo o uso de cabos.

Ainda conforme Morimoto (2011), o nome *Bluetooth* é uma alusão ao rei da Dinamarca e Noruega Harald Blantand (em inglês *Harold Bluetooth*), este rei é conhecido por ter unificado as tribos norueguesas, suecas e dinamarquesas. Da

mesma forma, o protocolo *Bluetooth* procura unir diferentes tecnologias, como telefones móveis e computadores.

De acordo com Siqueira (2012) o *Bluetooth* é um padrão de comunicação de baixo custo de fabricação e baixo consumo de energia, este foi inicialmente projetado para ser um padrão para substituir os cabos. O *Bluetooth* tem se tornado largamente utilizado em incontáveis dispositivos e representa uma parcela significativa das redes *wireless.*

Segundo Siqueira (2012), o *Bluetooth* é agrupado em 3 classes distintas, levando em consideração para essa distinção, o alcance das ondas de rádio dos dispositivos. A Classe 1 tem alcance máximo de no máximo 100 metros, a classe 2 tem alcance de no máximo 10 metros e a classe 3 tem alcance máximo de 1 metro.

Morimoto (2011) afirma que a versão inicial do protocolo foi criada por um grupo composto pela Ericsson, IBM, Nokia, Toshiba e Intel e foi publicada em julho de 1999. Pouco depois, o *Bluetooth* foi adotado pelo IEEE (*Institute of Electrical and Electronic Engineers*), dando origem ao padrão 802.15.1. Isso tornou a posição do *bluetooth* como um padrão aberto, o que expandiu sua adoção.

Siqueira (2012) esclarece que uma *Bluetooth Wireless Personal Network* (BT-WPAN) consiste de *piconets*, onde cada *piconet* é um conjunto de até 8 dispositivos conectados entre si, destes, um é designado como mestre e os outros escravos. Conforme Tiago, duas *piconets* podem estar conectadas (através de um *gateway*, *bridge* ou um outro dispositivo mestre) formando assim um *scatternet*, conforme ilustra a Figura 4.

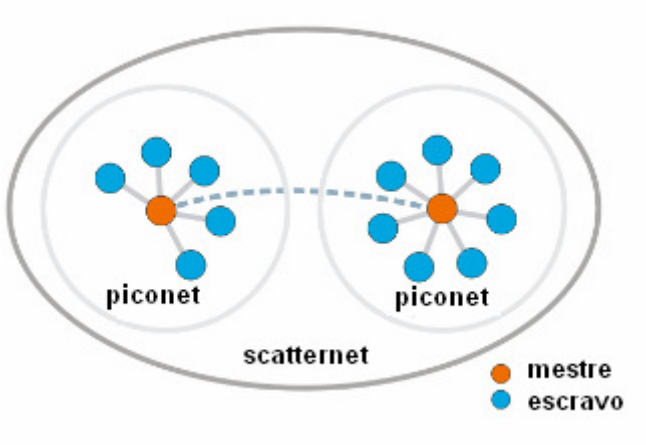


Figura 4. Topologia *Bluetooth*

Fonte: Siqueira (2012)

De acordo com Siqueira (2012), usualmente conexões *Bluetooth* são formadas de dispositivos conectando-se de forma ponto-a-ponto, entretanto, a especificação *Bluetooth* também define soluções com topologias mais complexas, formando *scatternets*, onde é possível que dois ou mais dispositivos que não estejam diretamente conectados possam comunicar-se.

Segundo Morimoto (2011), esta tecnologia é usada por um grande número de celulares, *tablets* e outros dispositivos, incluindo rádios, fones, teclados e mouses. As principais vantagens do *Bluetooth* são o baixo consumo elétrico, o que permite que os transmissores sejam usados em dispositivos pequenos demais para comportar uma interface *wireless* e o uso de *chips* mais simples, que faz com que os transmissores *Bluetooth* sejam baratos.

2.4 ORIENTAÇÃO A OBJETO

Os conceitos da orientação a objeto (OO) foram utilizados para a realização deste trabalho, estes conceitos serão descritos na sequência.

A necessidade de diminuir a complexidade elevada que o paradigma procedimental (ou estruturado) apresentava, levou a criação de uma nova abordagem: o paradigma orientado a objetos.

A forma que a programação procedimental realizava para codificar os programas era muito linear, e utiliza conceitos matemáticos para auxiliar a definir as soluções necessárias para resolver determinado tipo de problema.

O Paradigma Orientado a Objetos permite a criação de um código com maior legibilidade, onde as rotinas podiam ser mais facilmente reutilizadas e processos complexos podiam ser escritos de forma mais compreensível e de melhor manutenção (SANTOS, 2003).

A orientação a objetos (OO) torna o programador mais próximo do mundo real, no qual tudo pode ser visto como objetos. Antes da OO, o desenvolvimento se preocupava com as ações desses objetos, a programação se baseava nos verbos, como por exemplo: *cadastrarCliente, realizarVenda,* entre outros. O responsável pela codificação recebia os problemas na forma de objetos e acabava codificando em verbos. Com o advento da OO, o programador passou a codificar exatamente o que percebia do mundo real. A modelagem passou a se basear nos substantivos,

como por exemplo, o cliente, a venda etc. Os objetos, assim como no mundo real, possuem relações com outros objetos.

Ao desenvolver um sistema OO, não se analisa o problema linearmente. Primeiro se observa a interação dos entre si e qual a responsabilidade de cada um dentro do contexto do problema. Essa nova forma de raciocinar tornou sistemas grandes e complexos possíveis de serem realizados de forma mais compreensível. Para a realização de um código mais legível e reutilizável, a OO emprega alguns conceitos em seu paradigma de desenvolvimento como: herança, polimorfismo, agregação, associações, entre outros. Na sequência, são detalhados os principais componentes da OO (SANTOS, 2003):

Agregação: conceito onde um objeto contém outros objetos dentro de si. Existem dois tipos: agregação e agregação por composição. A agregação consiste no relacionamento entre dois objetos, onde um pode viver sem a existência do outro, diferentemente de composição, um relacionamento forte onde um objeto necessita da existência do outro.

Herança: o reuso de classes já existentes como instâncias de novas classes. As classes originais ficam assim contidas na nova classe, absorvendo os dados e comportamentos da classe existente anteriormente. Elementos mais específicos são completamente consistentes com o mais geral, e podem acessar seus atributos e métodos e programar novos.

Polimorfismo (“muitas formas”) permite a manipulação de instâncias de classes que herdam de uma mesma classe ancestral de forma unificada: pode-se escrever métodos que recebam instâncias de uma determinada classe e os mesmos métodos serão capazes de processar instâncias de qualquer classe que herde da classe criada já que qualquer classe que herde de suas característica será nela baseada.

A partir da orientação a objeto, desenvolve-se software a partir de agentes autônomos que interagem entre si, com isso, estabeleceram-se os princípios da orientação a objeto, que, segundo BEZERRA (2007) são:

1 – Qualquer coisa é um objeto;

2 – Objetos realizam tarefas através de requisição de serviços de outros objetos;

3 – Cada objeto pertence a uma determinada classe, uma classe agrupa objetos similares;

4 – A classe é um repositório para comportamento associado ao objeto;

5 – Classes são organizadas em hierarquias.

Utilizando a Orientação a Objeto, foi possível criar uma forma que permitiu programar softwares utilizando uma análise de como o mundo real funciona, esta implementação funciona utilizando-se de classes. Objetos e classes fazem parte do mundo, os objetos são as instâncias, ou exemplares de classes, que estabelecem quais as informações um objeto possui e como ele pode utilizá-las e manipulá-las. Os objetos são compostos de atributos e comportamentos ou métodos (WAZLAWICK, 2004).

Um sistema de software orientado a objeto consiste de objetos operando em conjunto com o objetivo final de realizar as funcionalidades deste sistema. Cada objeto é responsável por tarefas específicas, o sistema é desenvolvido através da interligação e cooperação entre os objetos.

2.4.1 Desenvolvimento orientado a objeto

O desenvolvimento de sistemas utilizando a orientação a objetos necessita de metodologias definidas e sedimentadas. A metodologia consiste na construção de um modelo de um domínio de aplicação e na posterior adição a este dos detalhes de implementação durante o projeto de um sistema.

De acordo com WAZLAWICK (2004), uma metodologia orientada a objeto para desenvolvimento de sistemas consiste de: ferramentas, notações gráficas e processos. Ainda segundo Waslawick, uma metodologia boa deve possuir os seguintes atributos:

- Aspectos de gerenciamento de projetos;
- Descrições de papéis e programas de treinamento detalhados;
- Exemplos completos;
- Exercícios de treinamento;
- Medidas para controle correticidade;
- Orientação para estimativa de custos;
- Procedimentos e políticas para garantia de qualidade do software;
- Técnicas definidas para utilização dos métodos.

Como o desenvolvimento de software orientado a objetos tornou-se um padrão, faz-se necessário a criação de ferramentas de modelagem para este paradigma. Após muitos debates e análises sobre as propostas de modelagem orientada a objetos, chegou-se a um padrão de modelagem universalmente adotado definido como Linguagem de Modelagem Unificada (UML).

2.5 LINGUAGEM DE MODELAGEM UNIFICADA (UML)

A UML define-se em uma linguagem para o desenvolvimento da estrutura de um projeto de software, ela serve para visualizar, especificar, construir e documentar os artefatos de um sistema complexo (BOOCH, JACOBSON e RUMBAUGH, 2006).

A modelagem auxilia no entendimento dos processos de um sistema, porém, para que isso seja possível, vários modelos interligados e com uma linguagem unificada como a UML são necessários, com isso, é possível criar modelos bem formados e complexos.

Por ser uma linguagem padrão, a UML trás uma grande vantagem, um desenvolvedor pode documentar a sua modelagem e depois a mesma pode ser interpretada sem duplicidade de sentidos por outro desenvolvedor, independentemente da linguagem que ele programe.

Criar uma modelagem é desenvolver um modelo simplificado da realidade e, com a UML, é possível construir modelos a partir de blocos de construção básicos. Estes blocos de construção são conhecidos como diagramas e correspondem a um grupo gráfico de vários conjuntos de elementos interligados. Todavia, é trabalhoso compreender um sistema complexo através de uma única perspectiva, por isso, os diagramas são divididos e permitem dirigir o foco para vários aspectos do sistema.

Diagramas que são bem elaborados facilitam a compreensão do sistema de forma geral, por isso, esta etapa merece atenção redobrada. No próximo capítulo serão apresentados alguns dos principais diagramas da UML, assim como um resumo dos mesmos.

2.5.1 Diagrama de classes

Segundo Eduardo Bezerra (2007), em seu livro Princípios de Análise e Projeto de Sistemas com UML, o diagrama de classes é usado no desenvolvimento do modelo de classes desde o estado de análise até o estado de especificação. De todos os diagramas da UML, este é o mais amplo em termos de notação.

Os diagramas de classes são utilizados para fazer a modelagem da visão estática do sistema, oferecendo suporte aos seus requisitos funcionais, isto é, as funcionalidades que o sistema deverá oferecer aos usuários finais.

A Figura 5 exemplifica um diagrama de classes, apresentando os itens e a estrutura que são: interface, colaborações, relacionamentos de dependência, generalização e associação, classes com nome, atributos e operações.

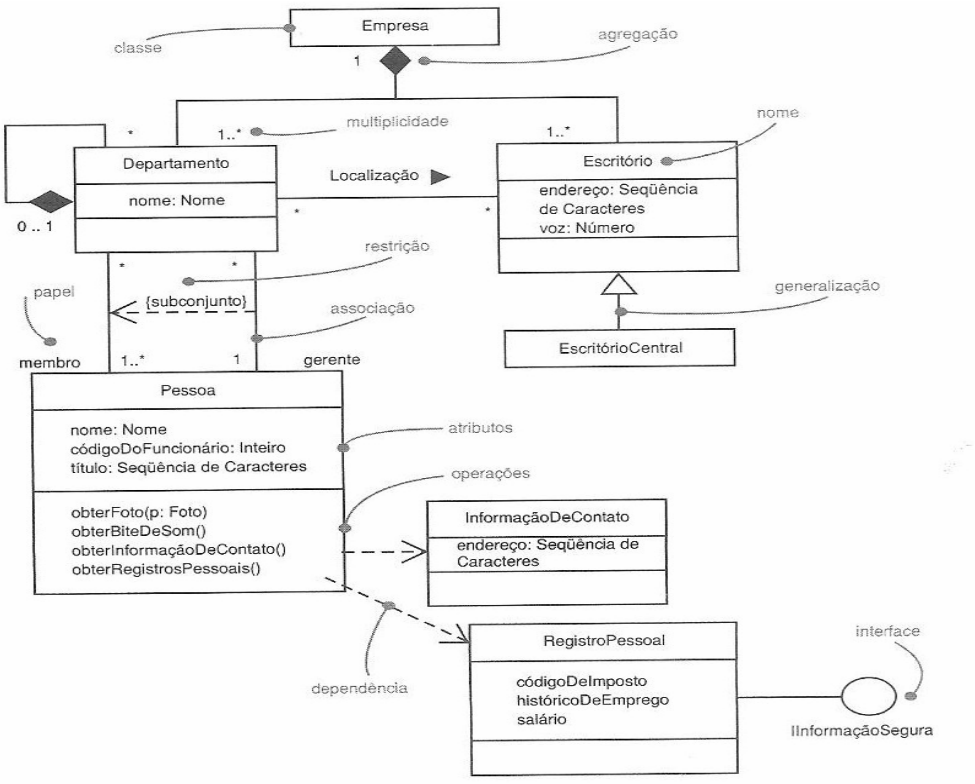


Figura 5. Diagrama de Classes
Fonte: JACOBSON, 2006

2.5.2 Diagrama de casos de uso

Ainda conforme Bezerra (2007), o diagrama de casos de uso corresponde a uma visão externa do sistema e representa graficamente os atores, casos de uso e relacionamentos entre esses elementos. O diagrama de casos de uso tem o objetivo de ilustrar em um nível alto de abstração quais os elementos externos interligam com que funcionalidades do software. Deste modo, a finalidade de um diagrama de caso de uso é ilustrar uma espécie de “diagrama de contexto” que apresenta os elementos externos de um sistema e as maneiras segundo as quais eles as utilizam.

A figura Figura 6 ilustra a notação da UML para exemplificar atores, casos de uso e relacionamentos de comunicação. Esses três elementos são os mais comumente utilizados.

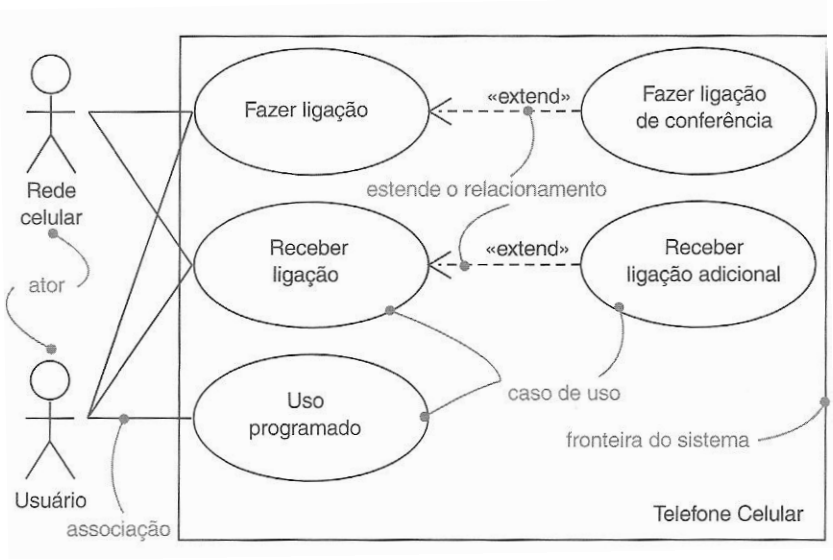


Figura 6. Diagrama de casos de uso
Fonte: JACOBSON, 2006

2.5.3 Diagramas de interação

Segundo Bezerra (2006) a ligação entre objetos para dar base à funcionalidade de um caso de uso denomina-se realização de um caso. A realização de um caso de uso descreve o comportamento de um ponto de vista de dentro do sistema. A realização de um caso de uso é apresentada através de diagramas de interação.

Os diagramas de interação auxiliam a documentar e a entender os aspectos dinâmicos do sistema de software, ou seja, eles mostram a sequência de mensagens enviadas e recebidas pelos objetos que participam em um caso de uso.

Ainda conforme Bezerra (2006), diagramas de interação representam como o sistema age internamente para que um ator atinja seu objetivo na realização de um caso de uso. A modelagem de um sistema de software orientado a objetos normalmente possui diversos diagramas de interação, o conjunto de todos os diagramas de interação de um sistema origina seu modelo de interações.

Existem dois tipos de diagrama de interação: diagrama de sequência e diagrama de colaboração.

2.5.4 Diagramas de sequência

O diagrama de sequência possui um conjunto de elementos gráficos, o foco está em apresentar como as mensagens são enviadas no decorrer do tempo. As mensagens são colocadas em ordem crescente, com o objetivo de dar ao leitor uma visão clara do fluxo de controle. Na Figura 7 são apresentados os elementos básicos de um diagrama de sequência.

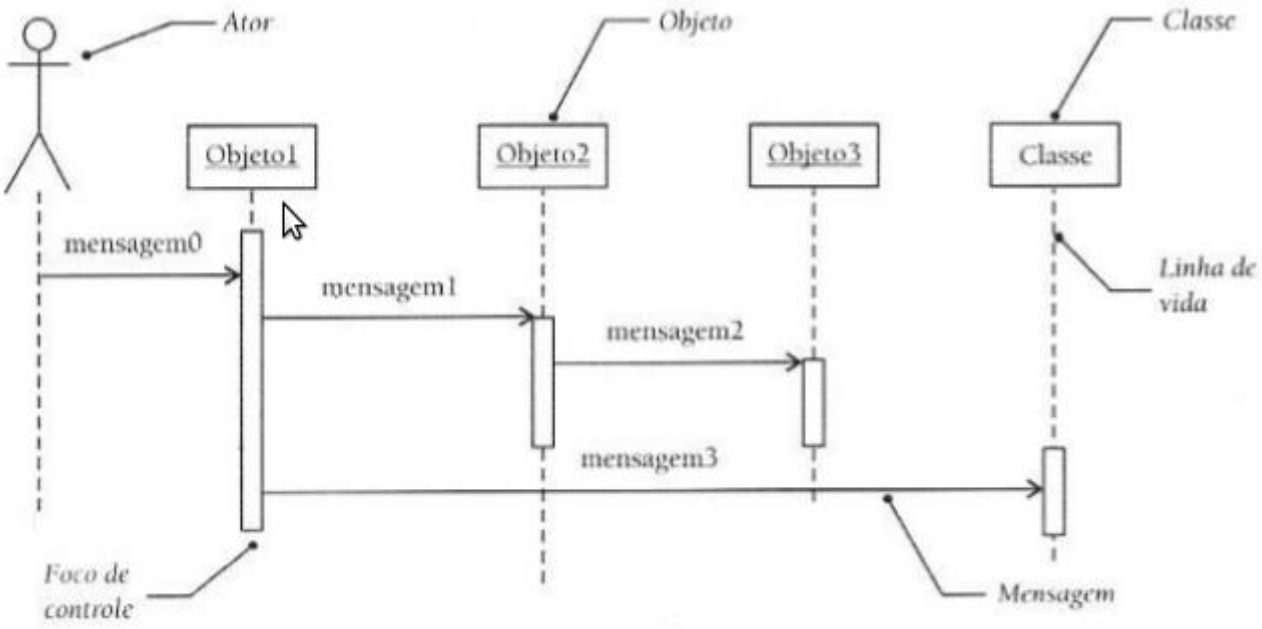


Figura 7. Diagrama de sequência
Fonte: Eduardo Bezerra (2006)

2.6 TRABALHOS E PESQUISAS DE REFERÊNCIA SOBRE JAVA ME

No meio acadêmico e científico diversos são os trabalhos e pesquisas que utilizaram a tecnologia Java ME como meio para alcançarem seus objetivos. O trabalho de Valvik (2012) busca estender o protótipo de um sistema em uma API abrangente (secureXdata), que pôde ser desenvolvida e testada em testes de campo e estudos de usabilidade, permitindo a criação de aplicativos Java ME seguros, conforme alguns requisitos de segurança utilizados. Segundo o autor, seu objetivo foi alcançado, gerando além da API objeto do estudo, um capítulo inteiro em sua dissertação sobre a programação em Java ME. O trabalho de Valvik está disponível em http://www.uib.no/filearchive/masters_thesis.pdf.

Outro trabalho que discute o esforço para fornecer aos desenvolvedores Java ME ferramentas para criar aplicações móveis, combinando a simplicidade e familiaridade do modelo de programação Ajax com a riqueza e ambiente seguro das APIs MAS (Mobile Service Architecture - JSR-248) é o trabalho de Arora e Hardy (2007), pesquisadores da Sun MicroSystems. Trata-se de uma biblioteca de código aberto que pode ser adicionada a qualquer aplicação Java ME. O trabalho dos autores descreve brevemente esta biblioteca, em conjunto com alguns casos de uso como exemplo. As conclusões deste trabalho afirmam que para aplicações que necessitam de utilizar as capacidades de um dispositivo móvel que não estão disponíveis em um navegador (como câmeras, *Bluetooth*, etc), uma pequena biblioteca que oferece o modelo de programação Ajax pode facilitar a tarefa para um desenvolvedor *web* que necessite programar neste tipo de dispositivo. O artigo encontra-se disponível na página da W3C (http://www.w3.org/2007/06/mobile-ajax/papers/sun.hardy.mobileAjaxJavaME.pdf).

Em (GEJIBO et. al., 2011), são analisados os desafios de implementação de um protocolo de segurança proposto com base na plataforma Java ME. O protocolo apresenta uma solução segura que encapsula os dados para armazenamento e transmissão sem exigir alterações significativas na aplicação móvel existente. Os autores disponibilizam sua pesquisa em http://www.uib.no/filearchive/challenges-in-implementing-end-to-end-secure-protocol-for-java-me-based-mobile.pdf.

A pesquisa proposta por WANG, DU, CHEN, (2009) analisa os mais populares frameworks de testes Java ME (JMUnit, J2MEUnit e Mobile JUnit,),

abordando os seus defeitos. O Framework JT Harness/ME também é analisado e discutido, mesmo este sendo um pouco ignorado pela comunidade acadêmica e científica. Como resultado, um esquema de design de um sistema de teste que integra o JT Harness/ME na plataforma NetBeans é proposto. Segundo os autores, os resultados dos testes mostram que este método aproveita as vantagens do JT Harness/ME e do NetBeans, oferecendo uma abordagem de teste conveniente e flexível com a capacidade de gerenciamento de testes. O artigo encontra-se disponível em http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5341593.

Outro trabalho interessante é o proposto por GRONLI, HANSEN, GHINEA (2010), onde é apresentado o desenvolvimento e avaliação de um protótipo de uma sala de reuniões inteligente que, através de diferentes plataformas móveis permite que os participantes e apresentadores compartilhem informações por meio da arquitetura de nuvem do Google, através de um grupo heterogêneo de dispositivos móveis, rodando em plataformas populares como Java ME, Windows Mobile e também Android. Os resultados mostram que a sala de reuniões inteligente, com sucesso, facilita a participação do usuário e sua interação, bem como a troca de informações no ambiente proposto, independentemente da plataforma móvel empregada pelos usuários. Neste trabalho a tecnologia Java ME é utilizada como meio para acessar a sala de reuniões inteligente proposta pelos autores. O trabalho pode ser encontrado e acessado no seguinte endereço eletrônico da IEEE, http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5615815&isnumber=5614033.

No trabalho de EKLER, NURMINEN, e KISS (2008), investiga-se como a tecnologia *peer-to-peer* (no caso em questão, o BitTorrent) pode ser utilizada em dispositivos móveis portáteis. Como este tipo de dispositivo está sendo cada vez mais utilizado para download de música e vídeo, os autores consideram interessante como um dispositivo portátil pode ser utilizado pelas comunidades existentes para acessar e reproduzir conteúdo multimídia. O documento é baseado nas experiências de implementação do BitTorrent para os telefones celulares (Nokia Series 40) sob a plataforma Java ME. O documento analisa as questões de implementação importantes, tais como grandes aplicações *peer-to-peer* em dispositivos móveis e compartilha as experiências práticas deste trabalho. O desempenho da aplicação é avaliado por medições de velocidade de utilização e de memória.

Por fim, outro trabalho recente, produzido por KAI et. al, (2010), investiga o

reconhecimento facial baseado em abordagens de controle de acesso projetado para câmeras equipadas com dispositivos móveis. Os autores desenvolveram uma Hierarquia de Correlação baseada na Verificação de Face, chamada de HCFV, que é aplicável para a maioria dos dispositivos móveis, com recursos limitados, como telefones móveis com o tamanho da memória disponível menor do que 1MB. Os resultados experimentais demonstram que a abordagem proposta conseguiu melhor desempenho do que o uso de esquemas de correlação convencionais. Além disso, a fim de investigar os problemas de aplicabilidade e exequibilidade, o HCFV proposto pelos autores foi utilizado no emulador Nokia S60 CLDC usando a tecnologia Java ME. A experiência mostra que a memória HFVC consome aproximadamente 1/10 de memória, 1/6 do espaço de armazenamento e tempo de processamento de 3,5% em comparação com os resultados utilizando os métodos convencionais. Este trabalho é interessante porque mostra outra forma de utilização do Java ME, e o documento pode ser acessado na biblioteca digital da IEE, no endereço http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5516852.

3 MATERIAIS E MÉTODOS

Este capítulo contém os materiais e o métodos utilizados para a modelagem e implementação do sistema. Os materiais se referem às ferramentas e as tecnologias, incluindo linguagem de programação, banco de dados e interface de desenvolvimento. O método se refere aos procedimentos utilizados para o desenvolvimento do sistema.

3.1 MATERIAIS

As seguintes ferramentas e tecnologias foram utilizadas para desenvolvimento do sistema.

3.1.1 Netbeans

Netbeans IDE é uma ferramenta de código aberto feita para auxiliar os desenvolvedores na criação de aplicativos em diferentes plataformas, utilizando tecnologia Java para isto. Nela são encontradas todas as ferramentas necessárias para projetos profissionais em Java, tanto para as plataformas *Web, Desktop* e *Mobile*, sendo, estas duas últimas utilizadas no desenvolvimento dos aplicativos. A IDE possui um ambiente visual de desenvolvimento móvel, conforme exemplifica a Figura 8.

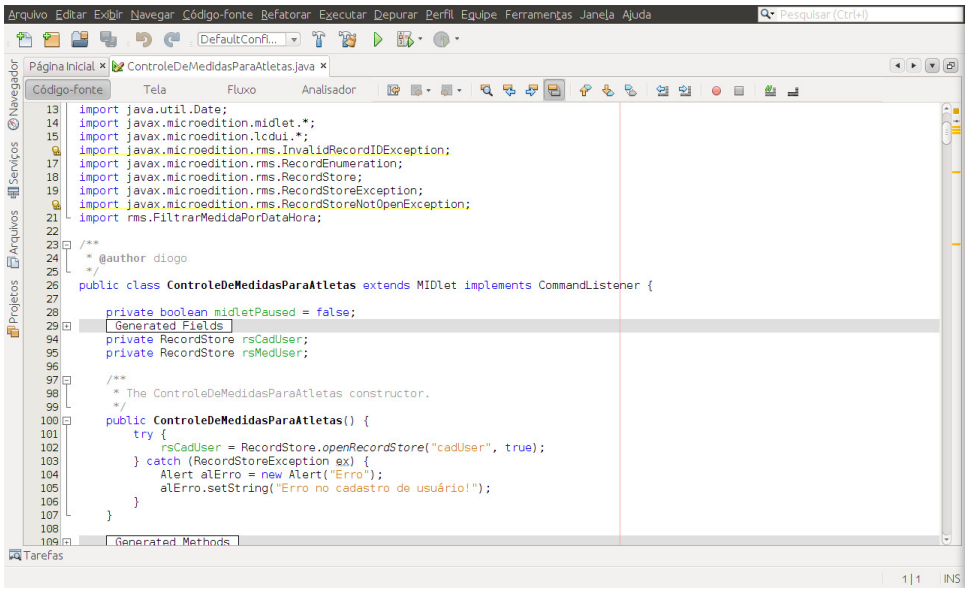


Figura 8. Ambiente Netbeans

3.1.2 MySQL

Para o desenvolvimento do software, é necessário um modo de gravar os dados, no caso, as medidas dos usuários do sistema, para que a linguagem de programação possa selecioná-los e processá-los conforme requisição enviada pelo utilizador. A melhor forma de armazenar estes dados é utilizando um SGDB (Sistema de Gerenciamento de Banco de Dados).

O MySQL é um SGDB que permite ao JAVA utilizar e exibir dados em um formato legível para o software desenvolvido. Ele é um servidor SQL (*Structured Query Language*) projetado para cargas altas de processamento de consultas complexas. Como um sistema de banco de dados relacional, o MySQL permite que várias tabelas diferentes sejam unidas para que a máximo eficiência e velocidade seja alcançada.

O MySQL foi desenvolvido pela empresa TCX em 1996, mas, nos dias atuais a Oracle mantém o programa. Segundo a Oracle, o MySQL é um dos SGBD mais rápidos disponíveis no mercado, dispondo quase todas as funcionalidades dos grandes bancos de dados. O MySQL é uma linguagem simples, em que você facilmente pode gravar, alterar e recuperar informações num banco de dados com segurança e rapidez. Em geral, o MySQL é utilizado em sistemas que participam da filosofia GNU/Linux, embora outros sistemas operacionais também ofereçam suporte, como o Windows da Microsoft, por exemplo.

Segundo Suehring (2002), no MySQL é possível escolher o tipo de tabela no momento de criação desta, já o formato de armazenamento dos dados, assim como alguns recursos do banco de dados dependem do tipo de tabela escolhido.

MySQL foi escolhido pois possui suporte ao JAVA e ao NetBeans, além de ser um banco de dados *Open Source*. Para realizar a conexão entre banco de dados e a aplicação foi utilizado o *Java Database Connectivity* ou JDBC, este é um conjunto de classes e interfaces desenvolvidas em Java que realizam o envio de instruções SQL para qualquer banco de dados relacional.

3.1.3 Visual Paradigm

Para o desenvolvimento dos diagramas UML, o software utilizado foi o Visual Paradigm, uma ferramenta CASE, voltada para UML, capaz de criar diversos diagramas, como diagramas de caso de uso, de atividades, de classes, de sequência, entre outros. A Figura 9 ilustra a interface do Visual Paradigm.

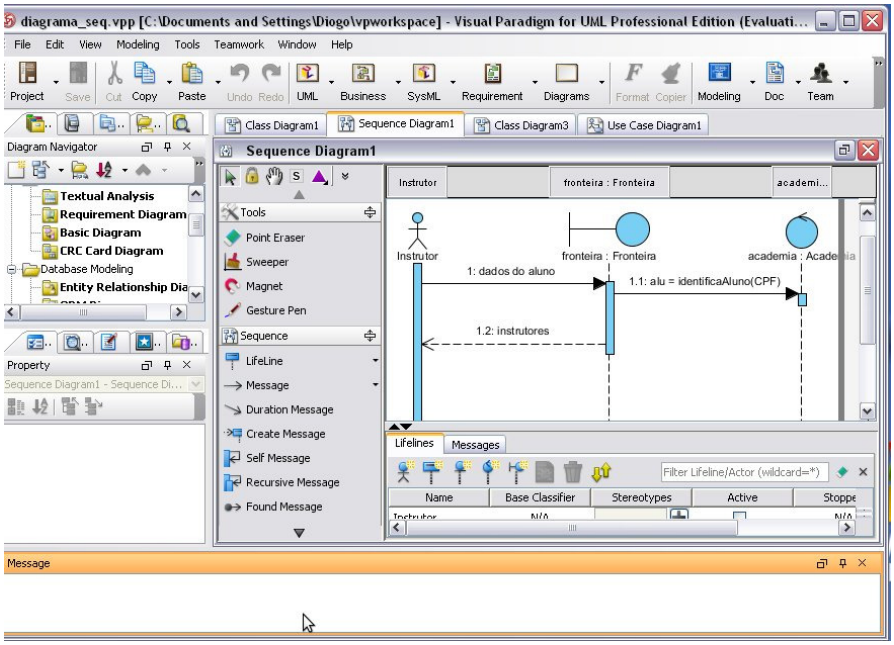


Figura 9. Ambiente Visual Paradigm

O aplicativo possui um ambiente de trabalho intuitivo, o que facilita a visualização e manipulação do projeto de modelagem, ele é uma aplicação comercial que oferece suporte a transformações de códigos-fonte de algumas linguagens de programação como, por exemplo, o Java, em diagramas UML. A ferramenta pode ser utilizada para diversos tipos de diagramas, podendo também ser utilizado como ferramenta case para criação e visualização de banco de dados. Os ícones e menus da ferramenta são bastante intuitivos, o que facilita a integração de novos usuários. Para quem trabalha com outros aplicativos gráficos, é possível exportar os diagramas e utilizá-los em outras plataformas salvando-a em formatos de imagem.

3.1.4 Java

Java é uma linguagem de programação orientada a objeto desenvolvida na década de 90 por uma equipe de programadores chefiada por James Gosling, na empresa Sun Microsystems. Diferentemente das linguagens convencionais, que são compiladas para código nativo, a linguagem Java é compilada para um código que é executado por uma máquina virtual.

A linguagem Java permite o desenvolvimento de aplicações para uma série de plataformas. É possível existirem software Java desde dispositivos pequenos, como telefones celulares, até computadores de grande porte, como os *mainframes*, por exemplo. Devido a essa característica, a linguagem Java conta com três conhecidos ambientes de desenvolvimento: o Java SE, o Java EE e o Java ME. No desenvolvimento do software para academia foram utilizadas as plataformas Java SE e Java ME.

3.1.5 Java SE

De acordo com Lindholm, et. al. (2013), entre as plataformas de desenvolvimento Java, o Java SE (*Java Standard Edition*) é a mais utilizada. Isso porque seu uso é voltado a PCs, onde há bem mais necessidade de aplicações. Além disso, pode-se dizer que essa é a plataforma principal, já que, de uma forma ou de outra, o Java EE e o Java ME tem sua base aqui. Pode-se dizer também que esses ambientes de desenvolvimento são versões aprimoradas do Java SE para as aplicações a que se propõem.

Por ser a plataforma mais abrangente do Java, o Java SE é a mais indicada para quem quer aprender a linguagem.

3.1.6 Java ME

De acordo com Muchow (2004), o Java ME (*Java Micro Edition*) é a plataforma de desenvolvimento para dispositivos móveis ou portáteis, como telefones celulares e *palmtops*. Como a linguagem Java já era conhecida e a adaptação ao Java ME não é complicada, logo surgiram diversos tipos de aplicativos

para tais dispositivos, como jogos e agendas eletrônicas. As empresas saíram ganhando com isso porque, desde que seus dispositivos tenham uma JVM (*Java Virtual Machine* - Máquina Virtual Java), é possível, com poucas modificações, programar os aplicativos em qualquer aparelho, sendo o único limite a capacidade do hardware.

A plataforma Java ME contém configurações e bibliotecas trabalhadas especialmente para a atuação em dispositivos portáteis. Assim, o desenvolvedor tem maior facilidade para lidar com as limitações de processamento e memória, por exemplo. Um exemplo disso é a configuração chamada CLDC (*Connected Limited Device Configuration*), destinada aos dispositivos com recursos de *hardware* bastante limitados, como processadores de 16 bits e memórias com 512 KB de capacidade. Essa configuração contém uma JVM e um conjunto básico de bibliotecas que permite o funcionamento da aplicação Java em dispositivos com tais características. Ela permite trabalhar com recursos como interface gráfica, armazenamento, acesso a rede de dados, acesso a recursos externos como GPS, sensores, *Bluetooth*, etc.

3.1.7 JFreeChart

JFreeChart é um *framework* de código aberto para a linguagem de programação Java, ele permite a criação fácil de gráficos, tanto interativos quanto não interativos, suportando variados gráficos, incluindo gráficos combinados.

É possível colocar vários marcadores e anotações sobre o traçado do gráfico. JFreeChart desenha automaticamente as escalas dos eixos e legendas.

De acordo com o endereço eletrônico oficial do projeto, que pode ser acessado através do endereço http://www.jfree.org/jfreechart/, o projeto JFreeChart foi iniciado há 13 anos atrás, em fevereiro de 2000, por David Gilbert. Hoje, JFreeChart é o *framework* de geração de gráficos mais utilizado para Java, com a versão 1.0.13 atingindo mais de 500.000 *downloads*. O projeto continua a ser desenvolvido por David Gilbert, com contribuições de uma comunidade diversificada de desenvolvedores.

3.1.8 Gráficos em Java ME com MECHART

O aumento na utilização de dispositivos portáteis, como, por exemplo telefones celulares, implica no crescimento da demanda por soluções móveis, exigindo cada vez mais dos desenvolvedores e de suas aplicações. Isso pode ser visto nas inovações proporcionadas na MIDP 2.0, no entanto, para a criação de interfaces gráficas somente com a *API* presente na biblioteca padrão, ainda existe um certo grau de dificuldade, além de um número bastante reduzido de componentes gráficos.

Funcionalidades consideradas comuns em outras plataformas, ainda são um desafio através do Java ME, por exemplo: a construção de relatório, persistência de dados, dentre outros. Porém, a comunidade Java ME nos fornece novidades, uma delas é a API MECHART, que fornece uma biblioteca para construção de três tipos de gráficos: gráficos de linhas, de setores e de barras. Como a *API* está registrada sobre GPL espera-se que mais colaboradores ajudem no crescimento dessa biblioteca, de modo que, futuramente, esta possa oferecer mais tipos de gráficos.

O projeto MECHART foi descontinuado, por isso, não existe um portal para o armazenamento do mesmo. A biblioteca foi baixada de portais independentes para compartilhamento de arquivos, onde alguns desenvolvedores disponibilizam uma biblioteca de classes para download.

MECHART é uma *API* que fornece uma biblioteca para construção de três tipos de gráficos: gráficos de linhas, gráficos em pizza e gráfico de barras.

3.1.9 Bluecove

Para permitir a integração entre as tecnologias *Bluetooth* e Java foi criada a Java APIs for *Bluetooth*, conhecida por JSR 82. A JSR 82 atualmente está disponível em diversos dispositivos e define a especificação para comunicação *Bluetooth* em diversos níveis.

De acordo com o endereço oficial do projeto Bluecove (http://www.bluecove.org), ele é uma biblioteca que programa o protocolo *Bluetooth* (implementação a especificação JSR-82) que é utilizada para a comunicação entre aplicação J2ME e J2SE.

Além do fato de ser gratuito (O Bluecove está sob a licença Apache License, Version 2.0, licença esta que permite a distribuição comercial de softwares com o Bluecove, desde que mencionada as fontes), o Bluecove é uma biblioteca para a linguagem Java de fácil aprendizagem que programa as principais funções para a comunicação com dispositivos *Bluetooth*, fazendo com que não haja a necessidade da criação de métodos para conexão diretamente dos celulares.

O Bluecove trabalha junto com a Java VM, utilizando uma pilha *Bluetooth* que existe no Windows, através de simples funções, fazem todo o trabalho de se comunicar com os aparelhos de celulares via *Bluetooth*, deste modo não é necessário trabalhar diretamente com o hardware do celular.

Na aplicação *desktop* o Bluecove foi adicionado para que a aplicação tivesse as funções pertinentes ao protocolo *Bluetooth*. Após isso, a comunicação com o celular tornou-se possível.

3.1.10 Projeto Marge

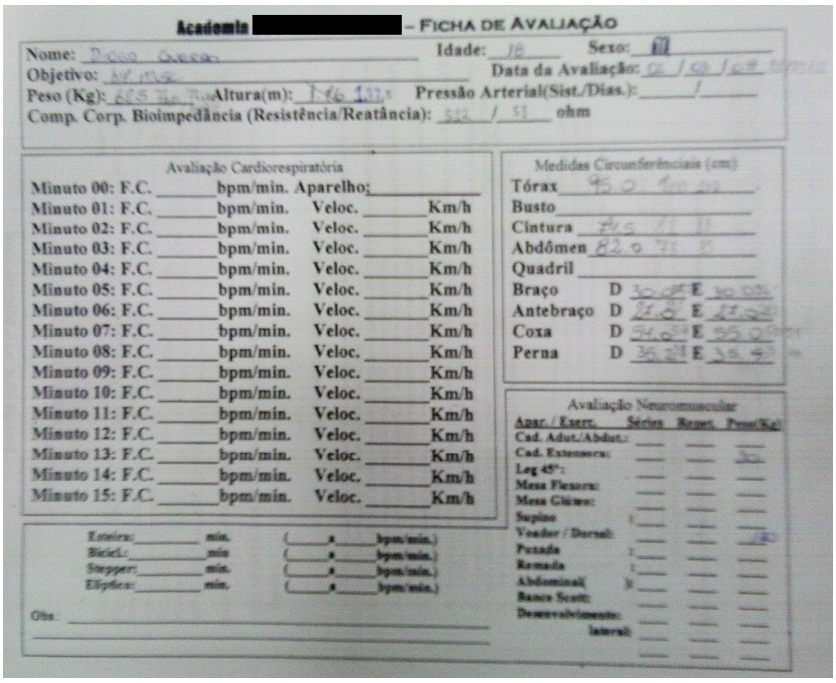
Segundo Ghisi (2007), o projeto Marge é um *framework* desenvolvido em Java que utiliza a biblioteca JSR 82 objetivando auxiliar na criação de softwares que utilizam a tecnologia *Bluetooth*. Sendo mantido por colaboradores de várias partes do mundo, foi criado com o intuito de facilitar o desenvolvimento de aplicações em Java que façam uso do *Bluetooth*, como por exemplo: *games*, controles remoto, aplicações de marketing, soluções para automação residencial, entre outras. O foco do projeto é acelerar o desenvolvimento das aplicações com *Bluetooth*, objetivando simplificar essa tarefa e concentrando o foco do desenvolvimento apenas na parte lógica do Software.

O projeto Marge não propõe apenas uma biblioteca para facilitar o uso de *Bluetooth* em Java, mas sim toda uma arquitetura definida pelo framework, que será responsável por integrar as outras partes relacionadas à comunicação *Bluetooth*, como os protocolos, as trocas de mensagens, pesquisa por dispositivos e mais serviços de maneira simples e suportando várias configurações. O projeto Marge é *Open Source* registrado sobre a licença LGPL (*Lesser General Public License*).

3.2 MÉTODO

Um dos principais objetivos da realização deste trabalho foi o aprendizado da intercomunicação de aplicações para aplicativos móveis e *desktop* utilizando a tecnologia *Bluetooth*. As fases deste trabalho foram: análise, codificação, sincronização e teste.

Com o trabalho de estágio de Diogo Guerro, um dos autores deste trabalho de conclusão de curso, foi desenvolvido um aplicativo móvel para cadastro de medidas para atletas, visando criar um sistema informatizado que pudesse substituir a ficha de avaliação, que era preenchida manualmente. A ficha pode ser visualizada na Figura 10.


Academia [redacted] – FICHA DE AVALIAÇÃO

Nome: Diogo Guerro Idade: 18 Sexo: M

Objetivo: Massa Data da Avaliação: 02 / 08 / 2014

Peso (Kg): 68,5 Altura(m): 1,76 Pressão Arterial(Sist./Dias.): ____ / ____

Comp. Corp. Bioimpedância (Resistência/Reatância): 512 / 51 ohm

Avaliação Cardiorespiratória

Minuto 00: F.C. _____ bpm/min. Aparelho: _____
Minuto 01: F.C. _____ bpm/min. Veloc. _____ Km/h
Minuto 02: F.C. _____ bpm/min. Veloc. _____ Km/h
Minuto 03: F.C. _____ bpm/min. Veloc. _____ Km/h
Minuto 04: F.C. _____ bpm/min. Veloc. _____ Km/h
Minuto 05: F.C. _____ bpm/min. Veloc. _____ Km/h
Minuto 06: F.C. _____ bpm/min. Veloc. _____ Km/h
Minuto 07: F.C. _____ bpm/min. Veloc. _____ Km/h
Minuto 08: F.C. _____ bpm/min. Veloc. _____ Km/h
Minuto 09: F.C. _____ bpm/min. Veloc. _____ Km/h
Minuto 10: F.C. _____ bpm/min. Veloc. _____ Km/h
Minuto 11: F.C. _____ bpm/min. Veloc. _____ Km/h
Minuto 12: F.C. _____ bpm/min. Veloc. _____ Km/h
Minuto 13: F.C. _____ bpm/min. Veloc. _____ Km/h
Minuto 14: F.C. _____ bpm/min. Veloc. _____ Km/h
Minuto 15: F.C. _____ bpm/min. Veloc. _____ Km/h

Esteira: _____ min. (_____ a _____ bpm/min.)
Bicicl.: _____ min. (_____ a _____ bpm/min.)
Stepper: _____ min. (_____ a _____ bpm/min.)
Elíptica: _____ min. (_____ a _____ bpm/min.)

Obs.: _____

Medidas Circunferenciais (cm)

Tórax 95,0 100,0
Busto _____
Cintura 84,5 77,0
Abdômen 82,0 77,0
Quadril _____
Braço D 30,0 E 30,0
Antebraço D 27,0 E 27,0
Coxa D 54,0 E 55,0
Perna D 35,5 E 35,5

Avaliação Neuromuscular

Aparelho / Exercício	Séries	Repetições	Peso(Kg)
Cad. Adut./Abdut.:	___	___	___
Cad. Extensora:	___	___	30
Leg 45°:	___	___	___
Mesa Flexora:	___	___	___
Mesa Glútea:	___	___	___
Supino	___	___	___
Voador / Dorsal:	___	___	___
Puxada	___	___	___
Remada	___	___	___
Abdominal	___	___	___
Banco Scott:	___	___	___
Desenvolvimento:	___	___	___
Lateral:	___	___	___

Figura 10. Ficha de avaliação do aluno

Esta forma de avaliação traz consigo algumas limitações: a ficha suja com o tempo, o histórico de evolução é perdido, o aluno só tem acesso à ficha na própria academia e com permissão da secretária, impossibilitando um acompanhamento constante e é necessário apagar uma medida para inserir outra, como mostrado na Figura 11.

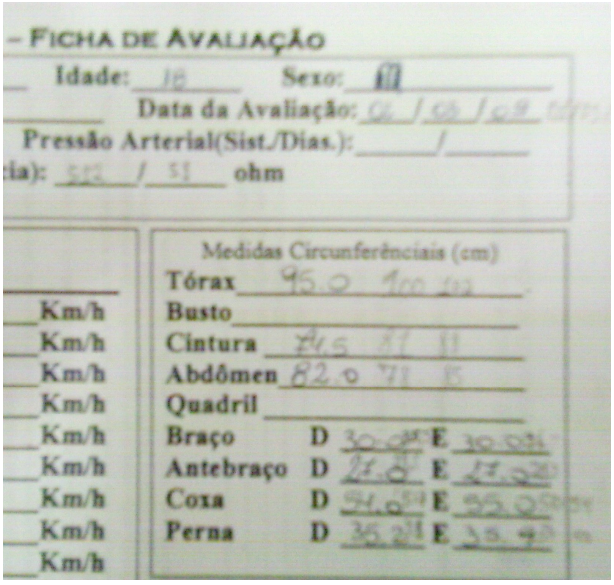


Figura 11. Ficha de avaliação – medidas

A partir deste *software*, elaborou-se a ideia de criar um aplicativo *desktop*, que pudesse ser usado como servidor para o aplicativo móvel, sendo a integração realizada a partir da tecnologia *Bluetooth*. A seguir uma descrição sucinta das etapas realizadas:

a) Análise

Na análise foram definidos os limites do protótipo do *software* e o que seria necessário para demonstrar a utilização da tecnologia *Bluetooth* seguindo os princípios da modelagem descritos no capítulo 2.

Para o aperfeiçoamento do aplicativo foi utilizada a ferramenta de análise Visual Paradigm (3.1.3 Visual Paradigm), já que o sistema passou a ser uma aplicação móvel com integração com o servidor *desktop*, onde é possível a sincronização dos dados.

b) Codificação e Sincronização

Na fase de Codificação e Sincronização o protótipo *desktop* foi desenvolvido e a importação de dados entre os aplicativos através do *Bluetooth* foi realizada, conforme apresentado na seção 4.4.

c) Testes

Os testes realizados pelos autores deste trabalho visaram encontrar erros de codificação e transmissão dos dados.

4 RESULTADOS E DISCUSSÕES

Neste capítulo serão apresentados os resultados obtidos tais como o padrão das telas, dos relatórios e das codificações utilizadas.

4.1 DESCRIÇÃO DO SISTEMA

O software proposto tem por meta proporcionar um fácil acesso ao histórico de medidas de usuários de academia. Este histórico deve ser acessível tanto pelo usuário quanto pelo instrutor da academia. Para tornar isso possível, se faz necessário o desenvolvimento de duas aplicações, uma delas para ser utilizada pelo praticante de atividades físicas através de seu celular para fins de consultas, e outra aplicação *desktop* que deve ser utilizada pelo instrutor para fins de lançamento de medidas do usuário e também de pesquisa e consulta de lançamentos realizados.

No sistema *desktop,* os instrutores serão os responsáveis pelo lançamento das medidas dos praticantes, permitindo que o aluno sincronize estes dados com a aplicação celular. Os instrutores podem alterar, inserir ou excluir usuários e lançamentos de medidas, além de poderem gerar gráficos referente às medidas dos usuários em relação ao tempo.

A aplicação celular deve manter apenas o histórico de medidas do próprio usuário, para que isso seja possível, é necessária a importação destes dados para a aplicação *desktop*, para tanto, o usuário precisa autenticar-se com a aplicação *desktop* através de um login/senha e solicitar seu histórico de medidas. Uma vez que estes dados estejam também presentes no aplicativo celular, a própria aplicação celular poderá gerar gráficos referentes às medidas do usuário em relação ao tempo. Os usuários também podem alterar sua senha através do celular.

4.2 MODELAGEM DO SISTEMA

Neste capítulo serão apresentados a modelagem do sistema e seus requisitos.

4.2.1 Documentos de requisitos

Os requisitos são as funcionalidades que devem ser formadas de modo a atender as necessidades do usuário. Este processo deve ser dividido em pequenas fases, com o objetivo de evitar que o processo seja complexo serão seguidas as seguintes etapas:

- Preparar uma lista contendo os requisitos do sistema;
- Detalhar de maneira resumida os principais requisitos;
- Organizar os requisitos em casos de uso.

4.2.2 Listagem de requisitos do sistema

A seguir a lista de requisitos do sistema de controle de medidas na aplicação celular:

F1 – Manter alunos;

F2 – Manter instrutores;

F3 – Manter histórico de medidas do utilizador;

F4 – Gerar gráfico do desenvolvimento do aluno;

F5 – Manter login no sistema;

F6 – Sincronizar dados dos alunos;

F7 – Consultar lançamentos de medidas.

4.2.3 Casos de uso do sistema

Na sequência são exemplificados os casos de uso do software, estes possuem uma especificação mais detalhada de cada funcionalidade que é vital para o sistema.

Nestes quadros são apresentados o código e o nome da funcionalidade, uma breve descrição, os códigos e as regras de negócios que estas deverão seguir. Também são apresentadas características da regra, ou seja, se ela é desejável ou obrigatória e se ela é permanente ou transitória.

F1 – Manter alunos	**Oculto ( )**			
Descrição: Através da aplicação *desktop* é possível cadastrar, alterar e excluir alunos através de um *login* com perfil de instrutor previamente cadastrado.				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
NF 1.1 – Nível de acesso.	Só poderão acessar essa funcionalidade os usuários com perfil de instrutor.	Segurança	( )	( )
NF 1.2 – Cadastro do usuário.	Instrutores poderão cadastrar os usuários no sistema através de informações como RG, nome, etc. Instrutores também podem excluir e alterar posteriormente informações referentes ao cadastro dos usuários.	Interface	( )	( )

Quadro 1. Manter alunos

F2 – Manter instrutores	**Oculto ( )**			
Descrição: Através da aplicação *desktop* é possível cadastrar, alterar e excluir instrutores através de um *login* com perfil de instrutor já previamente cadastrado.				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
NF 2.1 – Nível de acesso.	Só poderão acessar essa funcionalidade os funcionários com perfil de instrutor.	Segurança	( )	( )
NF 2.2 – Cadastro do instrutor.	Instrutores poderão cadastrar os instrutores no sistema através de informações como RG, nome, etc. Instrutores também podem excluir e alterar posteriormente informações referentes ao cadastro dos usuários.	Interface	( )	( )

Quadro 2. Manter instrutores

F3 - Manter histórico de medidas do utilizador.	**Oculto ( )**			
Descrição: O sistema deve manter o histórico de medidas do praticante de forma que o usuário possa acessar esta informação com facilidade e rapidez através do celular. O instrutor deve ter acesso aos dados de todos os alunos através do sistema *desktop*.				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
NF 3.1 – Nível de acesso.	O usuário tem total acesso para excluir e alterar as medidas em sua aplicação, mas isso não reflete na aplicação *desktop*.	Segurança	( )	( )
NF 3.2 – Manter medidas	O instrutor irá manter as medidas do aluno no aplicativo *desktop* que poderão ser sincronizadas com o aplicativo móvel posteriormente.	Interface	( )	( )
NF 3.3 – Identificação do professor.	O professor será identificado pelo seu código.	Interface	( )	( )
NF 3.4 - Identificação do cliente	O cliente será identificado pelo seu código.	Interface	( )	( )

Quadro 3. Manter histórico de medidas do utilizador

F4 – Gerar gráfico do desenvolvimento do aluno	**Oculto ( )**			
Descrição: O aplicativo *desktop* e o aplicativo móvel devem gerar gráficos baseados nas medidas lançadas pelo instrutor.				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
NF 4.1 – Níveis de acesso.	O instrutor pode gerar gráficos referentes às medidas de todos os alunos. Os alunos poderão gerar somente gráficos de suas medidas.	Segurança	( )	( )

Quadro 4. Gerar gráfico do desenvolvimento do aluno

F5 – Manter *login* no sistema	**Oculto ( )**			
Descrição: O sistema deve permitir que o usuário acesse a aplicação *desktop* através da aplicação celular mediante autenticação. Após isso, deve ser possível a importação dos dados da aplicação *desktop* para a aplicação celular. O usuário poderá alterar sua senha através do celular sem precisar do instrutor.				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
NF 5.1 – Nível de acesso	O usuário deve poder autenticar-se com seu *login* e senha na aplicação *desktop* através da aplicação celular	Segurança	( )	( )
NF 5.2 – Alteração de senha	O instrutor poderá alterar através da aplicação *desktop* a senha de qualquer usuário. Já o usuário, apenas poderá alterar a sua senha através da aplicação celular.	Segurança	( )	( )
NF 5.3 – Sincronização de dados	Após a autenticação o usuário poderá sincronizar os dados presentes na aplicação *desktop* para o seu celular.	Interface	( )	( )

Quadro 5. Manter *login* no sistema

F6 – Sincronizar dados dos alunos	**Oculto ( )**			
Descrição: Após autenticação do aluno no sistema através do aplicativo móvel, os dados devem ser sincronizados de modo que o aluno tenha em seu celular os mesmo dados lançados no aplicativo *desktop*.				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
NF 6.1 – Nível de acesso	O usuário deve poder autenticar-se com seu *login* e senha na aplicação *desktop* através da aplicação celular	Segurança	( )	( )
NF 6.2 – Sincronização de dados	Após a autenticação o usuário poderá sincronizar os dados presentes na aplicação *desktop* para o seu celular.	Interface	( )	( )

Quadro 6. Sincronizar dados dos alunos

F7 – Consultar lançamentos de medidas	**Oculto ( )**			
Descrição: Alunos e instrutores poderão consultar as medidas lançadas.				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
NF 7.1 – Visualizar medidas no celular	O usuário poderá visualizar as medidas sincronizadas para o aplicativo celular.	Segurança	( )	( )
NF 7.2 – Visualizar medidas no deskop.	O instrutor poderá visualizar todas as medidas de todos os usuários através da aplicação *desktop*.	Segurança	( )	( )

Quadro 7. Consultar lançamentos de medidas

4.2.4 Conceitos do Sistema

Na sequência são apresentadas as funcionalidades de nível médio representadas como conceitos do sistema. Estas terão codificação como classes. Após isso, são apresentados os conceitos, sendo que as letras “A", "C", "E" e "I" significam alteração, consulta, exclusão e inclusão respectivamente e estabelecem como devem ser realizadas as manutenções destas classes. A coluna referências cruzadas representa as relações existentes entre as funcionalidades. Através destas referências os desenvolvedores poderão saber qual a relação entre as funcionalidades.

Conceito	A	C	E	I	Observação	Referências Cruzadas
Medidas	X	X	X		As medidas serão incluídas pelo caso de uso F3 - Inserir medidas do aluno. Estas poderão ser alteradas ou excluídas.	F3
Aluno		X			O aluno não pode ser excluído se tiver medidas lançadas.	F4, F5, F7
Instrutor	X	X	X	X	O instrutor não poderá ser excluído se tiver qualquer vínculo com atividades e clientes da academia.	F1, F2, F3, F4, F5, F6, F7
Sincronização	X	X	X	X	Os dados devem ser sincronizados de modo que o aluno tenha em seu celular os mesmo dados lançados no aplicativo *desktop*.	F6

Quadro 8. Conceitos do sistema

4.2.5 Diagramas de casos de uso do sistema

Neste item são apresentados os diagramas de casos de uso utilizados para a modelagem do sistema proposto. A Figura 12 mostra o diagrama de casos de uso que representa as funcionalidades, nele também são apresentados os atores responsáveis por executá-las.

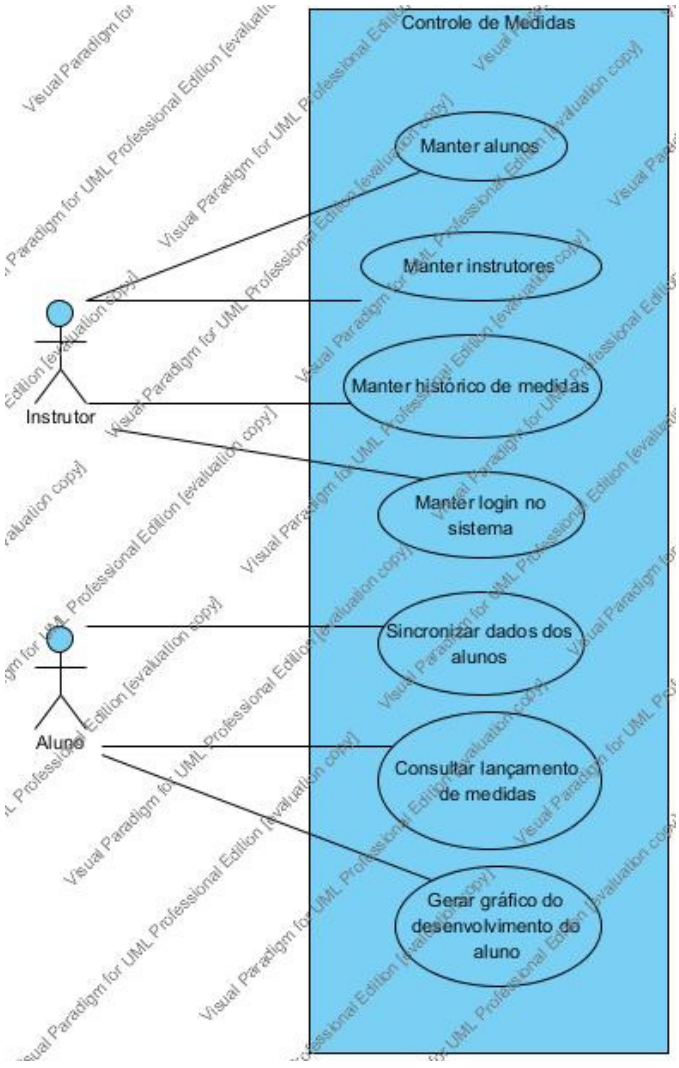


Figura 12. Diagrama de casos de uso das funcionalidades

4.2.6 Diagrama de classe

Na Figura 13 são apresentados os conceitos do sistema. Os conceitos apresentados passarão a ser tratados como classes após a construção do diagrama de sequência.

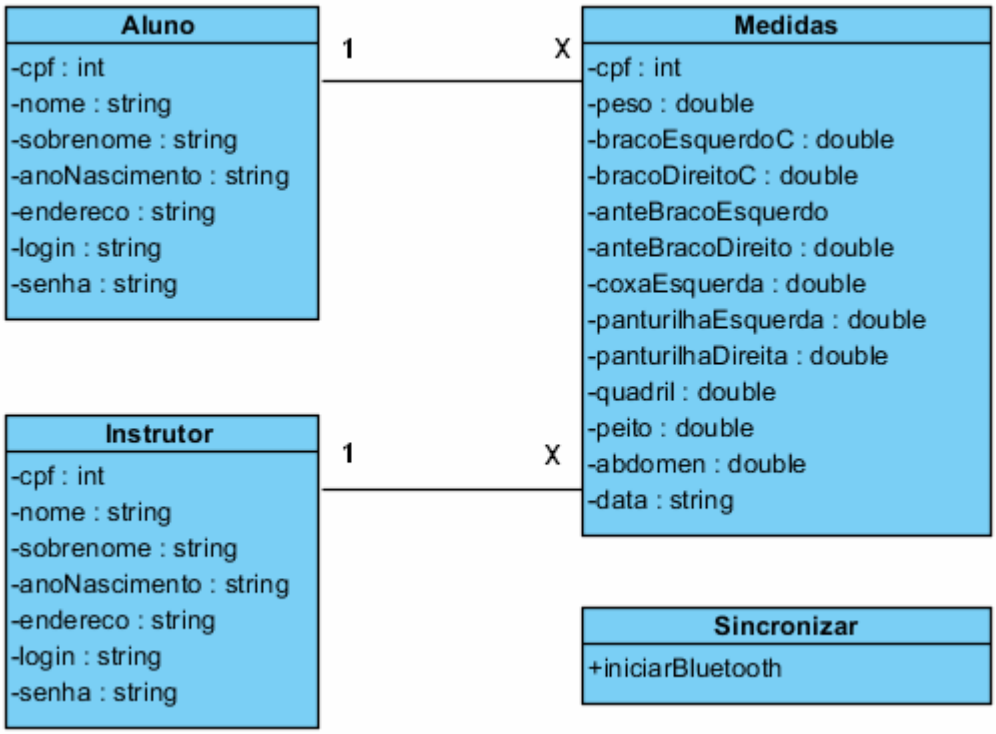


Figura 13. Diagrama de classe

4.2.7 Diagrama de sequência

Na Figura 14 é apresentado o diagrama de sequência, sendo apresentados os atores que se relacionam com as telas do sistema, quais as entradas de dados que farão, quais os métodos serão utilizados na classe principal para possibilitar a realização do caso de uso e as respostas que o sistema gerará.

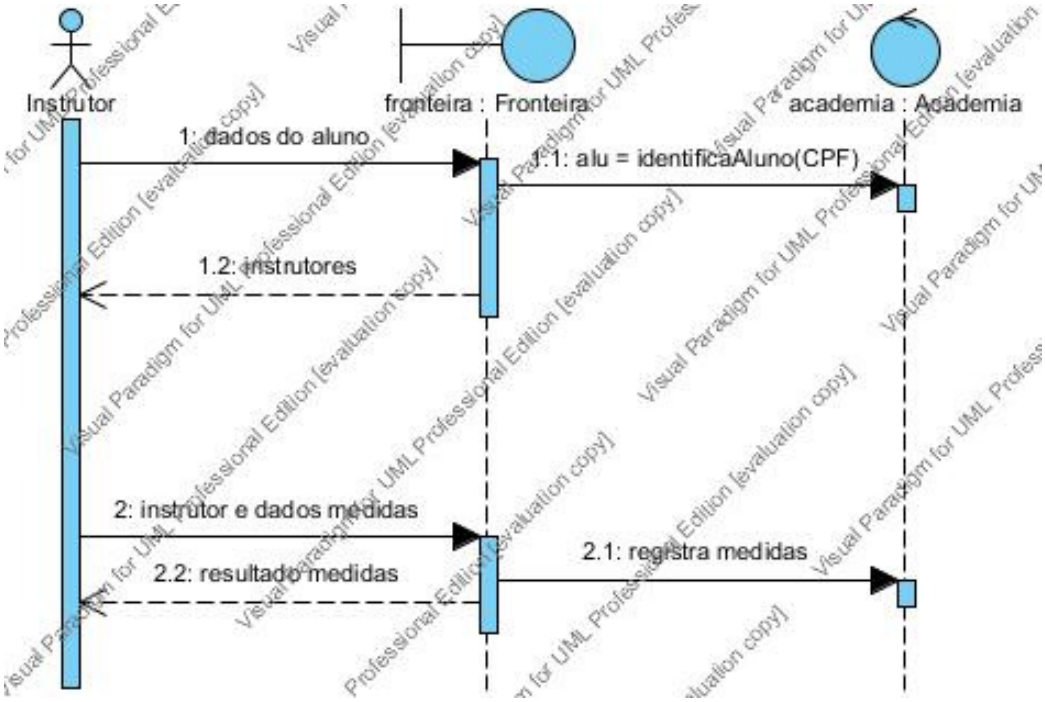


Figura 14. Diagrama de sequência

4.3 RELATÓRIO DO SISTEMA PRECEDENTEMENTE ÀS IMPLEMENTAÇÕES

Na sequência segue cada uma das telas do aplicativo precedentemente ao que foi proposto neste trabalho, com a descrição de suas funcionalidades.

Na Figura 15 é apresentada a tela inicial do aplicativo.

Figura 15. Tela inicial

Clicando no botão “Menu”, são mostradas três opções para o usuário: “1 Cadastrar Usuário”, “2 Lançar Medida” e “3 Visualizar Medida”.

Figura 16. Menu de opções

No sistema anteriormente desenvolvido, conforme mostrado na Figura 17 o usuário deveria se cadastrar no próprio celular, preenchendo o campo “Nome”, “Data de Nascimento” e “Altura” e clicando no botão “Salvar”.

Figura 17. Tela de cadastramento de usuário

Após a inclusão do registro, é mostrada uma tela informativa, conforme aparece na Figura 18.

Figura 18. Tela informativa

Caso o usuário acesse a opção 2 do Menu lançar medida, os campos para inserção das medidas do corpo serão mostrados: peso, braço esquerdo contraído, antebraço esquerdo, antebraço direito, coxa esquerda, coxa direita, panturrilha esquerda, panturrilha direita, quadril, peito e abdômen. Através deste Menu, o usuário pode atualizar suas medidas.

Na Figura 19, também aparece uma mensagem que solicita que o usuário preencha os campos com KG ou CM, que são as medidas padrão para o que é exigido no formulário.

Figura 19. Menu lançar medida

Após preencher todos os campos de acordo com o solicitado, aparece uma tela de confirmação, como mostrado na Figura 20.

Figura 20. Registro incluído com sucesso

O usuário poderá visualizar todas as medidas lançadas no celular até a presente data e poderá visualizar cada uma delas individualmente, de acordo com o mostrado na Figura 21.

Figura 21. Medidas lançadas

Ao clicar em “Visualizar”, conforme mostrado na Figura 21, aparecerão todas as medidas lançadas na data selecionada para fins de comparação, de acordo com a Figura 22.

Figura 22. Edição de medidas

Também há a possibilidade de alterar ou excluir o registro, que também aparece na Figura 22. Após executar qualquer uma destas ações, é mostrada uma Tela de informação.

Figura 23. Tela de informação

Finalizado o uso do programa o usuário pode voltar para a Tela inicial e sair deste.

4.4 APERFEIÇOAMENTO E IMPLEMENTAÇÃO DO SISTEMA

Na sequência segue cada uma das telas do aplicativo que foi desenvolvido com o objetivo de ampliar o que foi desenvolvido no trabalho de estágio de Guerro (2011), tendo em vista que o controle de medidas era feito pelo próprio usuário, no aplicativo móvel, conforme é mostrado na seção 4.3 RELATÓRIO DO SISTEMA PRECEDENTEMENTE ÀS IMPLEMENTAÇÕES. Há também a descrição das funcionalidades do sistema, implementação e integração entre os dispositivos.

Na Figura 24 há o nome do programa, “Controle de Medidas”, uma imagem de fundo que representa o programa, e três botões para acessar os menus com as funcionalidades do mesmo.

Figura 24. Tela inicial do aplicativo *desktop*

Clicando no menu “Cadastros”, depois em “Instrutor”, é mostrada a tela onde é possível manter o cadastro de instrutores, inserindo, alterando e excluindo os dados, conforme ilustra a Figura 25. Também é possível navegar entre os registros utilizando os botões de navegação.

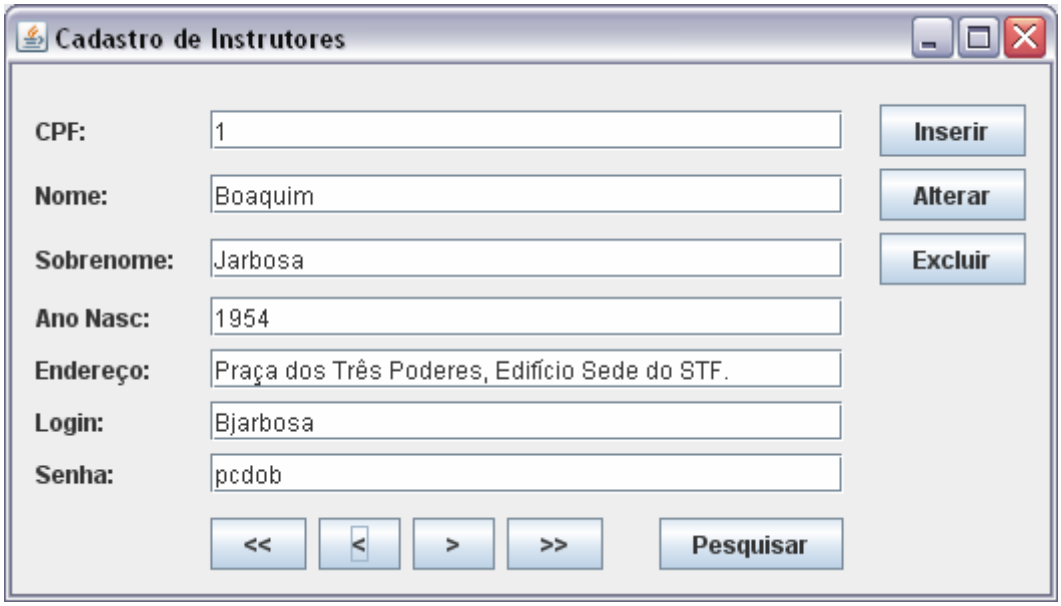


Figura 25. Cadastro de instrutores *desktop*

Conforme o que é exibido na Figura 26 o instrutor poderá realizar o cadastro de alunos, preenchendo os campos de acordo com os dados deste, podendo inserir, alterar ou excluir os dados de um aluno existente. Também é possível navegar entre os registros utilizando os botões de navegação.

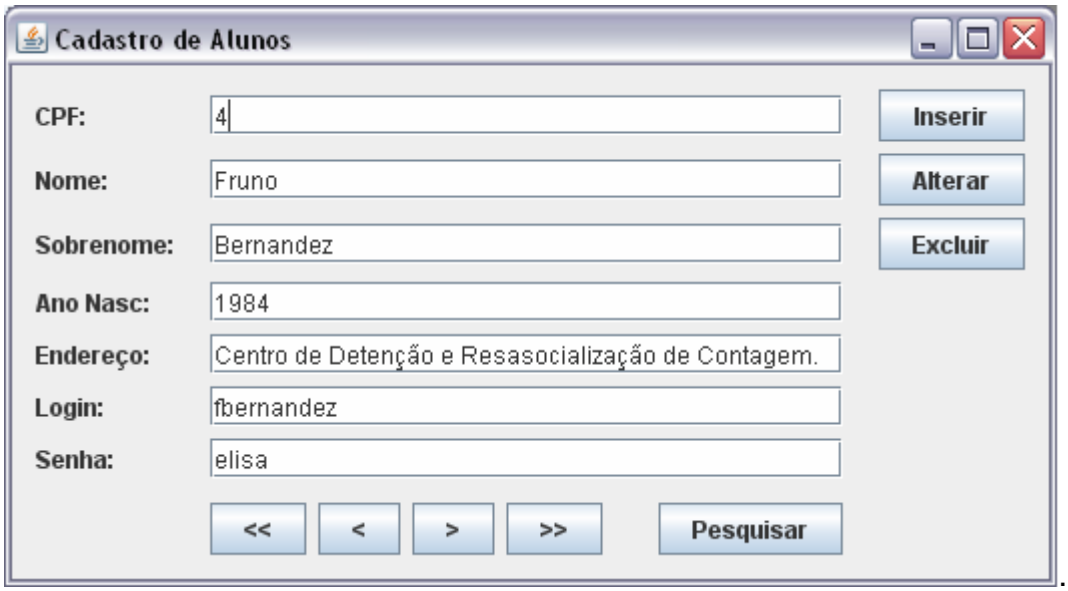


Figura 26. Cadastro de alunos *desktop*

Na Figura 27 aparece a tela de lançamento de medidas, onde o instrutor pode lançar as medidas de cada um dos músculos que serão salvas em uma determinada data, para um determinado aluno portador de um número de CPF.

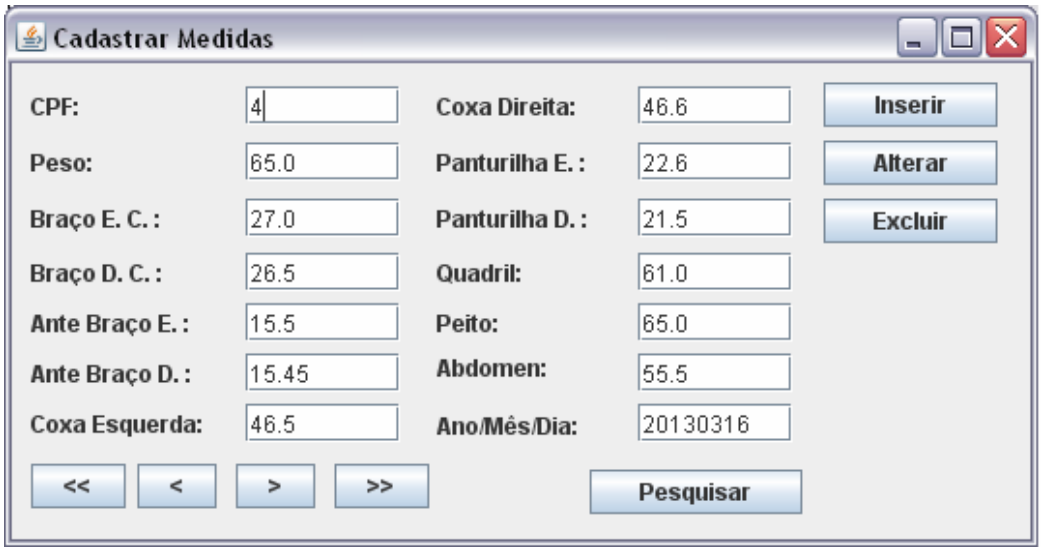


Figura 27. Medidas lançadas

Na Figura 28 são apresentadas as telas que podem aparecer a partir do clique do botão Pesquisar, ilustrado na Figura 27.

Figura 28. Pesquisar

A tela que aparece na Figura 29 pode ser acessada na Tela inicial do aplicativo *desktop* através da opção “GRÁFICOS” > “Visualizar”, ou então por meio da tela da Figura 29, clicando no botão “Pesquisar”.

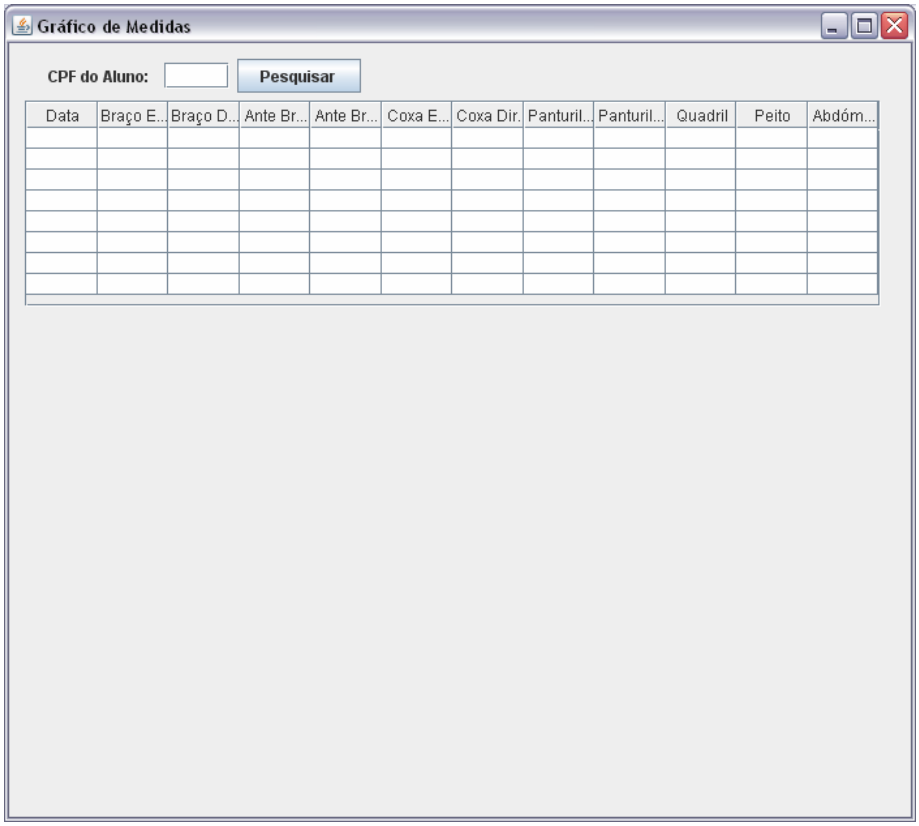


Figura 29. Gráficos

Na figura Figura 30 é mostrado o gráfico gerado com base nas medidas lançadas de um determinado aluno, através de seu CPF.

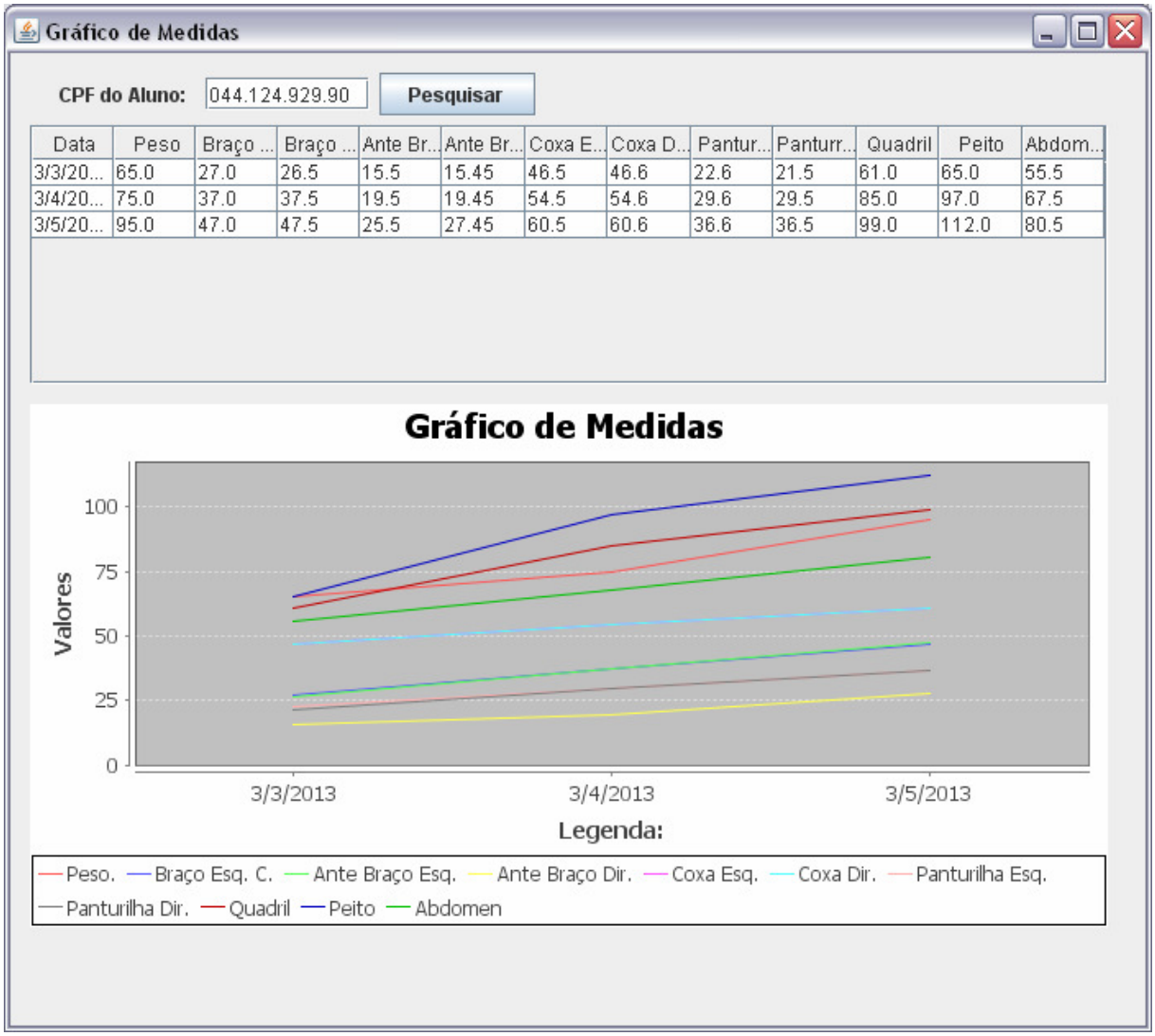


Figura 30. Gráfico gerado

O Quadro 9 mostra como é realizada a criação da tabela e dos gráficos apresentadas na Figura 30. Algumas variáveis são criadas e instanciadas, como a variável "dados" que armazena os dados referentes às medidas de um determinado aluno e será utilizada na montagem da tabela, e a variável “títulos” que será a legenda da tabela.

Para cada medida registrada, o sistema percorre o banco de dados e recupera as informações gravadas nas variáveis “dados” e “dadosGrafico”, que serão utilizadas para o preenchimento da tabela e do gráfico.

```
Object[][] dados = new Object[medidasArray.size()][];
         String[] titulos = new String[]{"Data", "Peso", "Braço Esq. C.",
             "Braço Dir. C.", "Ante Braço Esq.", "Ante Braço Dir.", "Coxa Esq.",
         "Coxa Dir.", "Panturrilha Esq.", "Panturrilha Dir.", "Quadril",
         "Peito", "Abdomen"};

         DefaultCategoryDataset dadosGrafico = new DefaultCategoryDataset();

         for (int i = 0; i < medidasArray.size(); i++) {
         MedidasBean medidas = medidasArray.get(i);

         String dataMedida = medidas.getData();
         String dataMedidaTratada = dataMedida.substring(3, 4) + "/" +
                 dataMedida.substring(5, 6) + "/" + dataMedida.substring(0, 4);

         String[] dadosLinha = new String[]{dataMedidaTratada,
String.valueOf(medidas.getPeso()),
         String.valueOf(medidas.getBracoEsquerdoC()),
String.valueOf(medidas.getBracoDireitoC()),
         String.valueOf(medidas.getAnteBracoEsquerdo()),
String.valueOf(medidas.getAnteBracoDireito()),
         String.valueOf(medidas.getCoxaEsquerda()), String.valueOf(medidas.getCoxaDireita()),
         String.valueOf(medidas.getPanturilhaEsquerda()),
String.valueOf(medidas.getPanturilhaDireita()),
         String.valueOf(medidas.getQuadril()), String.valueOf(medidas.getPeito()),
String.valueOf(medidas.getAbdomen())};
         dados[i] = dadosLinha;

         dadosGrafico.addValue(medidas.getPeso(), "Peso.", dataMedidaTratada);
         dadosGrafico.addValue(medidas.getBracoEsquerdoC(), "Braço Esq. C.",
dataMedidaTratada);
         dadosGrafico.addValue(medidas.getBracoDireitoC(), "Ante Braço Esq.",
dataMedidaTratada);
         dadosGrafico.addValue(medidas.getAnteBracoDireito(), "Ante Braço Dir.",
dataMedidaTratada);
         dadosGrafico.addValue(medidas.getCoxaEsquerda(), "Coxa Esq.", dataMedidaTratada);
         dadosGrafico.addValue(medidas.getCoxaDireita(), "Coxa Dir.", dataMedidaTratada);
         dadosGrafico.addValue(medidas.getPanturilhaEsquerda(), "Panturilha Esq.",
dataMedidaTratada);
         dadosGrafico.addValue(medidas.getPanturilhaDireita(), "Panturilha Dir.",
dataMedidaTratada);
         dadosGrafico.addValue(medidas.getQuadril(), "Quadril", dataMedidaTratada);
         dadosGrafico.addValue(medidas.getPeito(), "Peito", dataMedidaTratada);
         dadosGrafico.addValue(medidas.getAbdomen(), "Abdomen", dataMedidaTratada);

         }
         DefaultTableModel model = new DefaultTableModel(dados, titulos);
         tbMedidas.setModel(model);

         criaGrafico(dadosGrafico);
         }
```

Quadro 9. Geração de gráfico

Na classe “criaGrafico”, que pode ser visualizada no Quadro 10, as variáveis de configuração para geração do gráfico são instanciadas. Esta classe recebe as informações contidas na variável “dadosGrafico” e gera o gráfico na tela.

```
public void criaGrafico(DefaultCategoryDataset dataset) {
        CategoryDataset cds = dataset;
        String titulo = "Gráfico de Medidas ";
        String eixoy = "Valores";
        String txt_legenda = "Legenda:";
        boolean legenda = true;
        boolean tooltips = true;
        boolean urls = true;
        JFreeChart graf = ChartFactory.createLineChart(titulo, txt_legenda,
                eixoy, cds, PlotOrientation.VERTICAL, legenda, tooltips, urls);
        ChartPanel painelGrafico = new ChartPanel(graf, true);
        painelGrafico.setSize(pnGrafico.getWidth(), pnGrafico.getHeight());
        painelGrafico.setVisible(true);
        pnGrafico.removeAll();
        pnGrafico.add(painelGrafico);
        pnGrafico.revalidate();
        pnGrafico.repaint();
    }
```

Quadro 10. Classe para geração do gráfico

Na Quadro 11 é apresentado o código da conexão do banco MySQL na aplicação *desktop*, nele são definidas as variáveis referentes às configurações de acesso ao banco MySQL (usuário, senha e URL).

```
public static Connection getConnection() {

      if ( con == null ) {
            try {
                Class.forName("com.mysql.jdbc.Driver");
                String url = "jdbc:mysql://127.0.0.1:3306/banco";
                String usuario = "root";
                String senha = "root";
                con = DriverManager.getConnection(url, usuario, senha);
                return con;
            } catch (Exception ex) {
                return null;
            }
      } else {
        return con;
      }
  }
}
```

Quadro 11. Codificação da conexão MySQL

O código Quadro 12 mostra as variáveis criadas na aplicação *desktop* para facilitar a manutenção das tabelas. Dentro das variáveis são definidos os comandos MySQL que serão utilizados na manutenção da tabela. Neste caso, o código se refere à inserção dos dados no banco MySQL referente a tabela aluno.

```
public void conectarBanco() {

        try {

            Connection con = ConexaoFactory.getConnection();

            stmtInsert = con.prepareStatement(
                    "insert into alunos values ( ?, ?, ?, ?, ?, ?, ?)");
            stmtDelete = con.prepareStatement(
                    "delete from alunos where cpf = ? ");
            stmtUpdate = con.prepareStatement(
                    "update alunos set nome=?, sobrenome=?, anoNascimento=?, endereco=?,"
                    + " login=?, senha=? where cpf=?");
            stmtSelect = con.prepareStatement(
                    "select * from alunos where cpf=?");

            stmtNavegar = con.createStatement();
            rsNavegar = stmtNavegar.executeQuery("select * from alunos");

        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        }

    }
```

Quadro 12. Variáveis da conexão com o banco

O Quadro 13 mostra a codificação referente ao botão “<<” da Figura 26, nela é possível observar a utilização do código MySQL definido anteriormente, não sendo necessária a reprogramação de código MySQL para a manutenção das tabelas.

```
public AlunosBean primeiro() {

        try {

            rsNavegar.first();

            AlunosBean aluno = new AlunosBean();

            aluno.setCpf(rsNavegar.getInt("cpf"));
            aluno.setNome(rsNavegar.getString("nome"));
            aluno.setSobrenome(rsNavegar.getString("sobrenome"));
            aluno.setAnoNascimento(rsNavegar.getInt("anoNascimento"));
            aluno.setEndereco(rsNavegar.getString("endereco"));
            aluno.setLogin(rsNavegar.getString("login"));
            aluno.setSenha(rsNavegar.getString("senha"));

            return aluno;

        } catch (SQLException ex) {
            System.out.println("erro " + ex.getMessage());
            return null;
        }
    }
```

Quadro 13. Utilização do código MySQL

Ao iniciar o aplicativo *desktop*, a classe 'iniciarBluetooth' é ativada. Nela, como se pode observar no Quadro 14, a conexão *Bluetooth* inicia-se, utilizando os recursos da biblioteca Bluecove. No código apresentado na sequência, recursos físicos do computador são alocados com a instanciação das variáveis de comunicação. O número máximo de clientes é estabelecido e o servidor espera pela conexão por parte do cliente.

```
public void iniciarBluetooth() {
    communicationFactory = new RFCOMMCommunicationFactory();
    ServerConfiguration config = new ServerConfiguration(this);
    config.setMaxNumberOfConnections(1);
    communicationFactory.waitClients(config, this);
}
```

Quadro 14. Classe iniciar Bluetooth

O Bluecove trás alguns métodos abstratos, estes são: 'connectionEstablished', 'errorOnConnection', 'receiveMessage', 'errorOnReceiving' e 'errorOnSending'. Destes, alguns foram codificados no desenvolvimento desse trabalho.

O método 'connectionEstablished' é iniciado quando o cliente solicita um serviço para aplicação servidor, neste passo, como pode ser visto no Quadro 15, a conexão é realizada. Após este procedimento, uma mensagem é enviada do cliente para o servidor, essa mensagem é tratada no método 'receiveMessage'. Os métodos 'errorOnConnection', 'errorOnReceiving' e 'errorOnSending' são executados quando ocorrem erros na conexão, recebimento ou envio de dados. No desenvolvimento deste trabalho, devido aos fins serem estritamente acadêmicos, não houve necessidade da utilização de tais recursos. Porém, para aplicações que têm fins comerciais, a codificação desses métodos pode evitar que erros da comunicação do B*luetooth* reflitam na aplicação.

```
public void connectionEstablished(ServerDevice local, RemoteDevice rd) {
        this.local = local;

        local.startListening();
        local.setEnableBroadcast(true);
    }

    @Override
    public void errorOnConnection(IOException ioe) {
        System.out.println("Error: " + ioe.getMessage());
    }

    @Override
    public void receiveMessage(byte[] bytes) {***}

    @Override
    public void errorOnReceiving(IOException ioe) {
        System.out.println("Error: " + ioe.getMessage());
    }

    @Override
    public void errorOnSending(IOException ioe) {
        System.out.println("Error: " + ioe.getMessage());
    }
```

Quadro 15. Métodos abstratos Bluecove

No Quadro 16 as medidas cadastradas no *desktop* são sincronizadas com o celular, utilizando-se o método abstrato 'receiveMessage'. Tal código recebe o CPF do usuário que é enviado pelo celular, após isso, ele pesquisa todas as medidas que este usuário possui e as adiciona em uma string separando-as por grupos musculares e medidas de datas diferentes com o caractere "/". Após isso a string é enviada para o celular e a conexão *Bluetooth* é reiniciada para novas sincronizações.

```
public void receiveMessage(byte[] bytes) {

        String message = new String(bytes);
        ArrayList<MedidasBean> medidasBlue = dao.pesquisar(Integer.parseInt(message));
        String enviarDados = "";

        for (int i = 0; i < medidasBlue.size(); i++) {
            MedidasBean medidas = medidasBlue.get(i);
            enviarDados = enviarDados + String.valueOf(medidas.getData()) + ";" +
String.valueOf(medidas.getPeso()) + ";"
                    +      String.valueOf(medidas.getBracoEsquerdoC())      +      ";"      +
String.valueOf(medidas.getBracoDireitoC()) + ";"
                    +     String.valueOf(medidas.getAnteBracoEsquerdo())     +     ";"     +
String.valueOf(medidas.getAnteBracoDireito()) + ";"
                    +       String.valueOf(medidas.getCoxaEsquerda())       +       ";"       +
String.valueOf(medidas.getCoxaDireita()) + ";"
                    +     String.valueOf(medidas.getPanturilhaEsquerda())     +     ";"     +
String.valueOf(medidas.getPanturilhaDireita()) + ";"
                    +        String.valueOf(medidas.getQuadril())        +        ";"        +
String.valueOf(medidas.getPeito()) + ";"
                    + String.valueOf(medidas.getAbdomen());
            if ((i + 1) < medidasBlue.size()) {
                enviarDados = enviarDados + "/";
            }
        }
        local.send(enviarDados.getBytes());
        communicationFactory = null;
        iniciarBluetooth();
    }
```

Quadro 16. Envio de medidas para o celular

Na Figura 31, é possível observar a tela inicial do aplicativo móvel, esta possui uma imagem de fundo, um título e alguns menus, sendo estes: "Sair", "Cadastrar Usuário", "Lançar Medidas", "Visualizar Medidas" e "Sincronizar". Ao pressionar a opção "Sair", o aplicativo é finalizado.

Figura 31. Tela inicial

Conforme mostrado abaixo Figura 32, é possível visualizar a tela de cadastro de utilizador do aplicativo móvel. Nesta tela algumas informações do usuário são solicitadas para cadastramento do mesmo. Para demonstração do recebimento e envio de informações pelo *Bluetooth* através do aplicativo, apenas o CPF está sendo utilizado, porém, é possível enviar mais informações para validação do usuário, como por exemplo, *login* e senha.

Figura 32. Cadastro de usuário

Na Figura 33 é apresentada uma lista das medidas lançadas, tal tela pode ser acessada através da opção presente na Tela inicial.

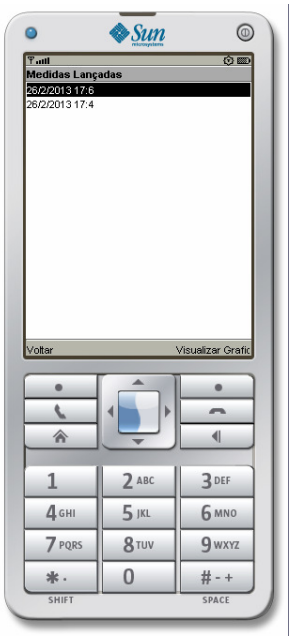


Figura 33. Medidas lançadas

Conforme mostrado na Figura 34, também é possível visualizar evolução individual de cada músculo. Clicando na região desejada o gráfico poderá ser visualizado.

Figura 34. Evolução individual

A tela que aparece na Figura 35 exemplifica um gráfico gerado no celular, com base em dados lançados de uma determinada medida no aplicativo *desktop* pelo instrutor e sincronizada para o aplicativo móvel.

Figura 35. Gráfico na aplicação móvel

Para a conexão *Bluetooth* pelo celular, foi utilizada a Biblioteca *Marge*, esta dispõe de alguns métodos abstratos que foram codificados no desenvolvimento deste trabalho. Ao selecionar o menu sincronizar disponível na aplicação, a classe "ConnectionBluetoothThread" inicia a pesquisa por dispositivos *Bluetooth,* como pode ser visto no Quadro 17.

```
  class ConectionBluetoothThread extends Thread {

        public void run() {
            try {
                LocalDevice.getLocalDevice().setDiscoverable(DiscoveryAgent.GIAC);
                factory = new RFCOMMCommunicationFactory();
                devices = new Vector();
                DeviceDiscoverer.getInstance().startInquiryGIAC(
ControleDeMedidasParaAtletas.this );

            } catch (BluetoothStateException ex) {

                System.out.println( "Error: " + ex.getMessage() );
            }
        }
    }
```

Quadro 17. Pesquisa por dispositivos *Bluetooth*

Após a inicialização da classe exibida anteriormente, o método abstrato 'deviceDiscovered' é executado e procura pelos serviços *Bluetooth* disponíveis. Adicionando o nome do serviço à tela do dispositivo móvel. Ao finalizar a pesquisa, o método 'inquiryCompleted' é executado, atualizando o título da aplicação celular com a informação de finalização da busca e quantos dispositivos foram localizados. Caso algum erro ocorra no meio deste processo o método 'inquiryError' será executado informando que houve falha. Como pode ser visto no Quadro 18.

```
public void deviceDiscovered(RemoteDevice rd, DeviceClass dc) {
        try {

            devices.addElement( rd );
            getLsPrincipal().append( rd.getFriendlyName(false), null );
            getLsPrincipal().setTitle( "Device: " + devices.size() );

        } catch( Exception e ) {
            System.out.println( "Error: " + e.getMessage() );
        }
    }

    public void inquiryCompleted(RemoteDevice[] rds) {
        getLsPrincipal().setTitle( "Completo - Dispositivos: " + devices.size()  );
    }

    public void inquiryError() {
        System.out.println( "Error on inquiry" );
```

Quadro 18. Métodos abstratos Marge

Após isso, o serviço servidor é escolhido pelo usuário no dispositivo móvel. Ocorrendo a confirmação de dados no celular, conforme ilustra a Figura 37, o método “serviceSearchCompleted” é iniciado e as medidas recebidas no servidor são tratadas no método demonstrado no Quadro 19.

```
public void serviceSearchCompleted(RemoteDevice rd, ServiceRecord[] srs) {
        try {
            ClientConfiguration config = new ClientConfiguration( srs[0], this );
            ClientDevice clientDevice = factory.connectToServer(config);

            ByteArrayInputStream bin = new ByteArrayInputStream(rsCadUser.getRecord(1));
            DataInputStream din = new DataInputStream(bin);
            String nome = din.readUTF();
            String dataNasc = din.readUTF();
            double altura = din.readDouble();
            String cpf = din.readUTF();

            clientDevice.startListening();

            getLsPrincipal().deleteAll();

            clientDevice.send( cpf.getBytes() );
            config.setCommunicationListener( this );

        } catch (Exception e) {
            System.out.println("Erro na sincronização");
        }
    }
```

Quadro 19. Solicitação de dados para o servidor

A figura Figura 36 apresenta o dispositivo móvel procurando pelo serviço Bluetooth e a mensagem que ele mostra quando não encontra nenhum serviço disponível.

Figura 36. Busca por dispositivos

As telas que aparecem na Figura 37 se referem à sincronização do aparelho celular com a aplicação *desktop*, a imagem exemplifica o serviço *desktop* encontrado, a mensagem de que a sincronização foi realizada com sucesso, e a verificação das medidas na aplicação celular.

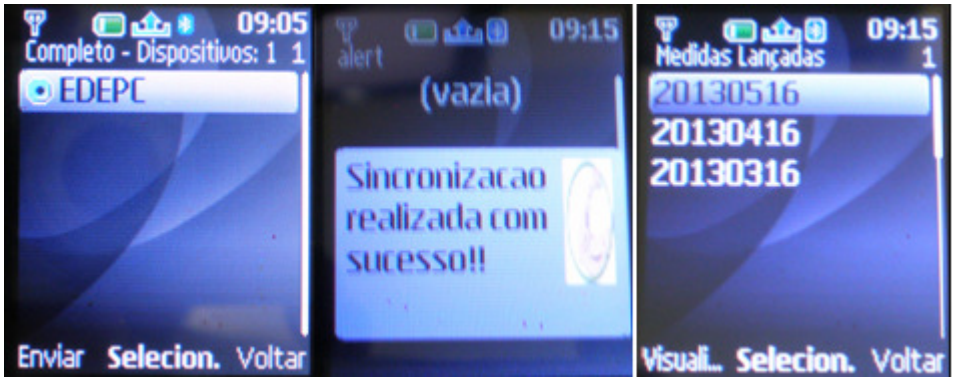


Figura 37. Sincronização *Bluetooth*

O Quadro 20 mostra o procedimento realizado para decodificação da string recebida pela aplicação *desktop*. Tendo em vista que o Java ME não possui a função "*split*", presente no Java SE, houve a necessidade da utilização de uma classe que possibilitasse a quebra da *string* por caractere definido. O código possibilita a quebra da *string* e a gravação no banco de dados.

```
StringTokenizer primeiroDivisor = new StringTokenizer(resp, '/');
                String[][] conjuntoDados = new String[primeiroDivisor.size()][13];
                int contadorPrimeiroArray = 0;
                while (primeiroDivisor.hasNext()) {
                    String aux = primeiroDivisor.next();
                    StringTokenizer segundoDivisor = new StringTokenizer(aux, ';');
                    for (int i = 0; i < 13; i++) {
                        conjuntoDados[contadorPrimeiroArray][i] = segundoDivisor.next();
                    }
                    contadorPrimeiroArray++;
                }
                for (int i = 0; i < contadorPrimeiroArray; i++) {
                    ByteArrayOutputStream bout = new ByteArrayOutputStream();
                    DataOutputStream dout = new DataOutputStream(bout);
                    for (int j = 0; j < 13; j++) {
                        if (j == 0) {
                            dout.writeUTF(conjuntoDados[i][j]);
                        } else {
                            dout.writeDouble(Double.parseDouble(conjuntoDados[i][j]));
                        }
                    }
                    rsMedUser.addRecord(bout.toByteArray(), 0, bout.toByteArray().length);
                }
                getAlStatus().setString("Sincronizacao realizada com sucesso!!");
```

Quadro 20. Recebimento das medidas pelo celular

5 CONSIDERAÇÕES FINAIS

Neste capítulo serão apresentadas as conclusões e as perspectivas futuras do sistema desenvolvido.

5.1 CONCLUSÃO

Com o fim do desenvolvimento deste trabalho, concluiu-se que o objetivo principal foi atingido. O sistema para o controle de medidas com seus requisitos, funcionalidades e conceitos planejados foi desenvolvido com sucesso. Pôde-se perceber que o ambiente *desktop* ainda tem muito a ser ampliado, porém, todos os recursos propostos estão funcionando adequadamente, servindo de guia para trabalhos futuros.

Algumas dificuldades foram encontradas, principalmente no que se refere a troca de dados entre as plataformas através da tecnologia bluetooth. Inicialmente tentou-se conexão direta entre o *desktop* e aparelho móvel pelo bluetooth, através da biblioteca Bluecove. Não obtendo o resultado esperado, o projeto Marge foi utilizado, com esse, foi possível realizar a troca de mensagens entre as plataformas.

A criação de gráficos com a biblioteca MECHART foi difícil, pois o projeto está interrompido, e sua ultima versão não foi encontrada para *download*, o link no site oficial do projeto está desabilitado.

O objetivo deste trabalho estava centrado no aperfeiçoamento de um sistema que possa ser útil para pessoas em busca de melhor qualidade de vida e também para atletas profissionais, fornecendo um controle fácil e de rápido acesso às medidas destes nos vários períodos de prática de atividade física, servindo também como incentivo para tal prática.

5.2 TRABALHOS FUTUROS

Desenvolvimento do software para *smartphones* e aperfeiçoamento do ambiente para *desktop*, ampliando as funções nele contidas.

No software desenvolvido neste trabalho, é realizada a comunicação através do protocolo *Bluetooth* para que haja a comunicação entre arquiteturas de hardware

distintas, um servidor *desktop* e um celular que comporta aplicações Java ME. Porém, alguns *smartphones,* com o sistema operacional Android por exemplo, não suportam esta tecnologia, limitando o software a determinados celulares.

Para aplicações futuras o aplicativo móvel poderia ser desenvolvido para o sistema operacional Android.

Além disso, novas funcionalidades no aplicativo *desktop* poderiam ser inseridas, como por exemplo: montagem de dietas, treinos e cálculo de gordura corporal.

6 **REFERÊNCIAS**

ABREU, Leonardo Marques de. **Usabilidade de Telefones Celulares com Base em Critérios Ergonômicos.**: Dissertação de mestrado da PUCRio - Departamento de Artes & Design, 294 f. 2004

ARORA Akhil, HARDY Vincent. **Mobile Ajax for Java ME Technology.** Sun MicroSystems Engineering. CiteSeerX - **10.1.1.168.217** 2007

BEZERRA, Eduardo. **Princípios de Análise e Projeto de Sistemas com UML**. 2a Ed. Rio de Janeiro: Elsevier, 2007.

BOOCH Grady; JACOBSON Ivar; RUMBAUGH James. **UML: Guia do Usuario.** Elsevier Editora. 474 p. 2006.

BURGIERMAN, Denis Russo. **Era Uma Vez na Olímpia**. Revista Super Interessante, São Paulo, n. 156, p. 37-43, out./dez. 2000. Disponível em < http://super.abril.com.br/esporte/era-vez-olimpia-441615.shtml >. Acesso em 22/06/2012.

CAPINUSSÚ Maurício; COSTA Lamartine. **Administração e Marketing nas Academias de Ginástica e Musculação.** São Paulo, Ibrasa, 1989.

EKLER, P.; NURMINEN, J.K.; KISS, A., **Experiences of Implementing BitTorrent on Java ME Platform.** Consumer Communications and Networking Conference. pp.1154-1158. ISBN:978-1-4244-1456-7. 2008

GEJIBO Samson, MANCINI Federico, MUGHAL Khalid A., VALVIK Remi B., KLUNGSØYR Jørn. **Challenges in Implementing an End-to-End Secure Protocol for Java ME-Based Mobile Data Collection in Low-Budget Settings.** Disponível on-line no endereço eletrônico http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.224.7407. 2011

GHISI, Bruno Cavaler. **Marge: Framework Para Desenvolvimento de Aplicações em Java Que Façam Uso da Tecnologia Bluetooth**. Trabalho de Conclusão de Curso - Graduação em Sistemas De Informação) - Universidade Federal de Santa Catarina. Disp. em http://pt.scribd.com/doc/38839698/tcc-brunoghisi-marge. 2007.

GRONLI, T. HANSEN, J. GHINEA, G. **A Context-Aware Meeting Room: Mobile Interaction and Collaboration Using Android, Java ME and Windows Mobile**. Computer Software and Applications Conference Workshop. ISBN:

978-1-4244-8089-0 Page(s): 311- 316. 2010

GUERRO, Diogo. **Desenvolvimento de um Software para Controle de Medidas de Atletas Usando Java ME**. UTFPR, 2011.

ISBN-13: 9780133260441. Disponível no endereço eletrônico http://docs.oracle.com/javase/specs/jvms/se7/html/. 2013

JSR-82. **JSR-82:** Java Bluetooth, 2006. Disponível em: < http://www.jsr82.com >. Acesso em 10/12/2012.

KAI Xi; YAN Tang; JIANKUN Hu; FENGLING Han. **A correlation based face verification scheme designed for mobile device access control: From algorithm to Java ME implementation.** 5th IEEE Conference on Industrial Electronics and Applications (ICIEA). pp.317,322. ISBN: 978-1-4244-5045-9. 2010

KRUCHTEN, P. **Introdução ao RUP: Rational Unified Process.** Rio de Janeiro: Ciência Moderna, 2003.

KRUCHTEN, P. **The Rational Unified Process: An Introduction**. Addison Wesley, 3ª Edição. 2003.

LINDHOLM, T.; YELLIN, F.; BRACHA, G. e BUCKLEY, A. **The Java™ Virtual Machine Specification - Java SE 7 Edition**. Addison-Wesley Professional

MORIMOTO, Carlos E. **Redes, Guia Prático.** 560 p. ISBN: 978-85-99593-09-7. Editora Meridional. 2011.

MUCHOW John W. **Core J2me: Tecnologia e MIDP**. Editora MAKRON BOOKS - GRUPO PEARSON, Série Java. ISBN: 9788534615228. 588 p. 2004

ORGANIZAÇÃO MUNDIAL DE SAÚDE, **Riscos a saúde.** Disponível no endereço eletrônico http://www.who.int/. Acesso em 11/01/2013.

PEREIRA, M.M.F. **Academia: Estrutura Técnica e Administrativa**. Rio de Janeiro: Sprint, 1996.

SANTOS Rafael. **Introdução à Programação Orientada a Objetos Usando Java.** Editora: CAMPUS Pgs: 352. ISBN 853521206X. 2003.

SIQUEIRA, Thiago Senador de. **Bluetooth – Características, protocolos e funcionamento**. Disponível no endereço eletrônico http://www.ic.unicamp.br/~ducatte/mo401/1s2006/T2/057642-T.pdf. Último acesso em 11 de outubro de 2012.

STEINHILBER, Jorge. **Profissional de Educação Física** Ed. Sprint, Rio de Janeiro R.J. 1996.

SUEHRING, Steve. **MySQL, A Bíblia.** Rio de Janeiro: Elsevier, 2002.

VALVIK Remi A. B. **Security API for Java ME: secureXdata.** Long Master Thesis – Institute of Informatics of University of Bergen - Normay 110p. 2012.

WANG Zhenglei, DU Zhenjun, CHEN Rong. **A Testing Method for Java ME Software.** Proceedings of the 2009 International Conference on Scalable Computing and Communications; Eighth International Conference on Embedded Computing Pages 58-62 IEEE Computer Society. ISBN: 978-0-7695-3825-9. 2009

WAZLAWICK, Raul Sidnei. **Análise e Projeto de Sistemas de Informação Orientados a Objetos**. Editora Campus, 2004.