UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE SISTEMAS DE INFORMAÇÃO – BACHARELADO

PROTÓTIPO DE UM SISTEMA DE RASTREAMENTO PARA COMPUTADORES

BÁRBARA DIAS PEREIRA

BLUMENAU
2013

2013/2-02

BÁRBARA DIAS PEREIRA

PROTÓTIPO DE UM SISTEMA DE RASTREAMENTO PARA COMPUTADORES

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

Prof. Miguel Alexandre Wisintainer, Mestre – Orientador

BLUMENAU
2013

2013/2-02

PROTÓTIPO DE UM SISTEMA DE RASTREAMENTO PARA COMPUTADORES

Por

BÁRBARA DIAS PEREIRA

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

Presidente:	Prof. Miguel Alexandre Wisintainer, Mestre – Orientador, FURB
Membro:	Prof. Wilson Pedro Carli, Mestre – FURB
Membro:	Prof. Jacques Robert Heckmann, Mestre – FURB

Blumenau, 28 de novembro de 2013.

AGRADECIMENTOS

A Deus, por me fortalecer, sustentar, amar e capacitar todos os dias.

À minha família, que sempre me mostrou que o impossível não existe quando temos determinação e interesse pelo que almejamos atingir.

Ao meu amigo e namorado Thiago Henrique Teixeira por todo auxílio, paciência, compreensão, carinho e companheirismo.

Ao meu orientador, professor Miguel Alexandre Wisintainer, por ter acreditado na conclusão deste trabalho.

A Lucas Zampar Bernardi, pelo ato de bondade de emprestar sua placa para conclusão desse projeto.

Aos professores do Departamento de Sistemas e Computação da Universidade Regional de Blumenau por suas contribuições durante os semestres letivos.

O universo desorienta-me. Não posso imaginar que exista um relógio sem relojoeiro.

Voltaire

RESUMO

O presente trabalho apresenta um protótipo de um sistema de rastreamento para computadores que utiliza um hardware com software integrado para fazer o rastreamento e exibe seus resultados em uma página *web*. O rastreador é capaz de determinar a posição de um computador desde que o rastreador consiga receber sinal e que o computador em questão esteja devidamente cadastrado na aplicação *web*. Para transformar o hardware utilizado em um rastreador utilizou-se uma placa com Linux embarcado e adicionou-se à mesma um módulo GPS. O envio das coordenadas detectadas ao servidor da aplicação *web* é feito através da tecnologia 3G. No desenvolvimento do software embarcado utilizou-se a linguagem de programação Python e para construção da aplicação *web* utilizou-se os *frameworks* JSF2 e Hibernate juntamente com servidor de aplicação Jboss e banco de dados MySQL. Como resultado obteve-se um sistema de rastreamento para computadores que além de proporcionar mais segurança aos proprietários destes, também pode auxiliar na recuperação desses equipamentos em caso de perda ou roubo.

Palavras-chave: Sistema de Rastreamento. Segurança de Computadores.

ABSTRACT

This paper presents a tracking system prototype for computers using hardware with integrated software to trace and displays the results on a web page. The tracker is able to determine the position of a computer since the tracker can receive signal and the computer is properly registered in the web application. To transform the hardware in a tracker a board with embedded Linux was used added with a GPS module. The sending coordinates detected to the web application server is done through 3G technology. In the development of embedded software was used the Python programming language. In the web application building was used frameworks JSF2 and Hibernate with Jboss application server and MySQL database. As a result it was obtained a tracking system for computers wich provid more security to the owners of the computers, and also can help in the recovery of such equipment in case of loss or theft.

Key-words: System Tracking. Computer Security.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE SIGLAS

3G – Terceira Geração

CDMA – *Code Division Multiple Access*

FDMA – *Frequency Division Multiple Access*

FTP – *File Transfer Protocol*

GPS – *Global Positioning System*

GUI – *Graphical User Interface*

HTTP – *Hypertext Transfer Protocol*

HSDPA – *High-Speed Downlink Packet Access*

IP – *Internet Protocol*

JCP – *Java Community Process*

JSF – *JavaServer Faces*

J2EE – *Java Platform Enterprise Edition*

Kbits – *Quilobit* por segundo

MAC – *Media Access Control*

Mbps – *Megabit* por segundo

ODBC – *Open Database Connectivity*

SGBD – Sistema Gerenciador de Banco de Dados

SMTP – *Simple Mail Transfer Protocol*

SQL – *Structure Query Language*

TCP – *Transmission Control Protocol*

TDMA – *Time Division Multiple Access*

UMTS – *Universal Mobile Telecommunications System*

URL – *Uniform Resource Locator*

WCDMA – *Wide-Band Code-Division Multiple Access*

Wi-fi – *Local Area Network*

SUMÁRIO

1 INTRODUÇÃO

Nos últimos tempos, têm-se descoberto muitos conceitos, tecnologias, soluções e inovações. As pessoas estão procurando cada vez mais conhecimento e domínio das tecnologias – principalmente dos equipamentos que as incumbam e que estão sendo disponibilizados no mercado.

Segundo o *site* de notícias Convergência Digital (2013), a maioria das pessoas com idade entre 18 e 30 anos, chamada geração do milênio é a grande entusiasta de tudo o que a tecnologia tem a oferecer. No Brasil, 92% deles acreditam que a tecnologia facilita a transposição de barreiras de linguagem, em comparação a 87% em todo o mundo. Oitenta e cinco por cento creem que a tecnologia tornou mais fácil encontrar um emprego, comparado a 83% em nível global.

À medida que a tecnologia vem avançando nas aplicações para facilidade e agilidade dos processos diários, também vem crescendo no âmbito de segurança. Com a digitalização das informações, passou-se a desenvolver mais aplicativos voltados para a segurança e integridade da informação. Porém, em paralelo a esse crescimento, também houve avanços negativos, pois pessoas motivadas pelos seus interesses e má fé passaram a desenvolver agentes maliciosos com o objetivo de corromper a segurança de informações.

O Departamento de Segurança da Informação e Comunicações (DSIC) homologou em abril de 2013 uma série de diretrizes com o objetivo de orientar o servidor público federal, que trabalha com segurança da informação, a garantir maior eficiência em sua atuação. A iniciativa do governo vem em um momento em que é comum ver notícias de *sites* e sistemas eletrônicos com problemas relacionados à segurança da informação causados por ataques que comprometem a indisponibilidade, confidencialidade e integridade dos seus dados (REZENDE, 2013).

Contudo, proteger a informação não inclui somente o cuidado para com aquilo que se mantém dentro dos equipamentos que armazenam as informações, mas também para com o meio externo. Uma pesquisa realizada pela empresa de segurança *F-Secure* em 14 países ao redor do mundo, incluindo o Brasil, revelou que 74% dos entrevistados já perderam documentos importantes como fotos, e-mails, arquivos sensíveis, entre outros (COMPUTERWORLD, 2012).

Os softwares presentes no mercado, que têm a finalidade de localizar máquinas perdidas ou roubadas, estão direcionados para a localização do computador somente a partir

do momento que o mesmo estiver conectado à internet, além de que por serem apenas softwares, também estão vulneráveis à desinstalação.

Devido ao pouco investimento na segurança física de computadores e limitações nos processos de rastreamento existentes, verificou-se a necessidade de desenvolver um método que não dependa de conexões via internet do tipo *Local Area Network* (Wi-fi) padronizada pelo *Institute of Eletrical and Eletronics Engineers* (LAN IEEE 802.11) e do tipo via cabos – *Local Area Network* padronizada pelo *Institute of Eletrical and Eletronics Engineers* (LAN IEEE 802.3) para fins de rastreio e que assegure que as funções deste serão executadas livre do que possa acontecer com o computador, como formatação e desinstalação de softwares.

Com base nesta análise, identificou-se a necessidade de desenvolver um sistema que seja voltado para a segurança física de computadores. O objetivo é aumentar a probabilidade de que em caso de perda ou roubo de computadores, os mesmos possam ser recuperados com o auxílio de um rastreador que estará implantado no interior da máquina e monitorará o deslocamento da mesma.

1.1. OBJETIVOS DO TRABALHO

O objetivo do trabalho é apresentar um protótipo de sistema para rastreamento de computadores.

Os objetivos específicos são:

a) registrar as informações referentes à localização do computador em qualquer lugar do mundo que possua cobertura 3G;
b) permitir que o usuário cadastre informações de identificação em uma página *web* para futura consulta da localização de seu computador;
c) disponibilizar, através de uma página *web*, um histórico de localizações do computador, registradas pelo rastreador proposto.

1.2. ESTRUTURA DO TRABALHO

No primeiro capítulo tem-se a introdução ao tema principal deste trabalho com a

apresentação da justificativa e dos objetivos.

No segundo capítulo apresenta-se a fundamentação teórica pesquisada sobre software embarcado, tecnologia de terceira geração (3G), *Hypertext Transfer Protocol* (HTTP), segurança da informação, *Transmition Control Protocol* (TCP) e *Internet Protocol* (IP), sistema operacional Linux, *Global Positioning System* (GPS) e trabalhos correlatos.

O terceiro capítulo apresenta o desenvolvimento do protótipo iniciando-se com o levantamento de informações, tendo na sequência a especificação, implementação e por fim resultados e discussão.

No quarto capítulo tem-se as conclusões deste trabalho bem como apresentam-se sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

O presente capítulo aborda assuntos necessários para o entendimento do trabalho, tais como conceitos e informações sobre segurança da informação, sistema operacional Linux, software embarcado, tecnologia 3G, sistema GPS, protocolo TCP/IP, protocolo HTTP e trabalhos correlatos.

2.1. SEGURANÇA DA INFORMAÇÃO

Os avanços da tecnologia têm permitido que, cada vez mais, processos sejam informatizados e automatizados. Informações importantes que eram depositadas em papéis e armazenadas em grandes porta-arquivos, passaram a ser registradas e conservadas em computadores.

O uso de equipamentos de informática para armazenamento de informações oferece praticidade e facilidade ao usuário. O fato de manter todos os registros em um só lugar torna as buscas muito mais rápidas e permite que as informações estejam muito mais acessíveis. Porém, independente da forma como os dados são guardados, é necessário que os mesmos estejam seguros.

Segundo Oliveira (2001, p. 9), “segurança das informações define-se como o processo de proteção de informações e ativos digitais armazenados em computadores e redes de processamento de dados.”.

Atualmente, existem vários agentes que ameaçam a segurança da informação armazenada em computadores e em seus programas. São diversos tipos de vírus, *spywares*, ataques de *hackers*, *spams*, entre outros. Porém, é possível dificultar o agir desses agentes através de aplicativos como antivírus e *antispywares*, já presentes no mercado, além de configurações de segurança do próprio computador, como o *firewall*, e políticas de segurança.

Contudo, os possíveis problemas que podem colocar a segurança das informações em risco, não se limitam a apenas vírus, *spywares*, *hackers* e invasões de rede, mas também a possíveis transtornos físicos que podem ocorrer com o equipamento que é responsável por manter tais informações. Alguns dos possíveis problemas são incêndios, causas da natureza, desabamentos e roubo.

Oliveira (2001, p. 75) destaca a importância não só da segurança lógica das informações, mas também da segurança física, que está relacionada ao equipamento utilizado. Medidas de proteção física, como serviços de guarda, alarmes, fechaduras e outros são parte da segurança de dados.

Ainda segundo Oliveira (2001, p. 75), as técnicas de proteção de dados por mais sofisticadas que sejam acabam não sendo completamente eficientes se a segurança física não for garantida. Não são apenas os ambientes que necessitam de segurança, os equipamentos, e principalmente os que armazenam informações importantes – computadores, em sua maioria, também precisam estar seguros e preparados para situações de risco.

2.2. SISTEMA OPERACIONAL LINUX

O Linux é um sistema operacional completo que é um clone livre do sistema operacional UNIX eficiente e estável. Ele cuida do funcionamento do computador e gerencia aspectos do mesmo como processador, memória, dispositivos, sistemas de arquivos e segurança (NEGUS, 2007, p. 7).

Linus Torvalds foi o criador do sistema operacional Linux, em 1991. Tudo começou quando o estudante finlandês encontrava-se frustrado com as carências do Minix – sistema operacional baseado no Unix, criado na época por um professor holandês chamado Andrew S. Tanenbaum para objetivos acadêmicos, como o de ensinar o funcionamento de um sistema operacional. Linus começou a idealizar como seria bom ter um sistema operacional que, além de gratuito, pudesse executar tarefas como transferência e armazenamento de arquivos e emulação de terminal. Em setembro de 1991, Linus libertou a primeira versão de seu sistema, o Linux versão 0.01 (PEREIRA, 2010).

Corrêa (2004) destaca que o ponto forte do Linux é a segurança, é o segundo sistema operacional mais seguro do mundo. Este é um dos motivos mais favoráveis a sua utilização, principalmente em servidores. Outro fator que comprova a segurança, é que das invasões executadas com sucesso, apenas 1% está relacionado com o Linux como alvo. A cada mil vírus criados, apenas um é para Linux e quando uma falha ou vírus é encontrado no sistema, o mesmo é eliminado em questão de poucas horas devido ao grande número de programadores que o Linux possui.

Com o passar do tempo, devido ao uso do *kernel* do Linux por empresas e pessoas,

estas começaram a criar programas para interagir com ele, disponibilizando-os para outras pessoas e assim foram surgindo milhares de distribuições. Para o Linux ser uma alternativa, ou uma solução, para empresas e pessoas, ele teria de ser fácil de instalar e configurar, sendo assim, visando facilitar esta tarefa, foram criadas distribuições pré-compiladas e disponíveis em CD-ROM, ou via *download*, em *sites* na internet (MATOS, 2007, p. 14).

Ainda de acordo com Matos (2007, p. 16), uma das distribuições criadas foi a Debian, que é uma das poucas distribuições que não é mantida por nenhuma empresa, mas sim por um conjunto de desenvolvedores. Esta distribuição desenvolveu ferramentas interessantes, como por exemplo, o APT para instalar pacotes, e atualmente conta com mais de 37.500 pacotes contendo softwares pré-compilados e distribuídos em um bom formato.

2.3. SOFTWARE EMBARCADO

Um sistema embarcado pode ser conceituado como um sistema microprocessado no qual o computador é completamente encapsulado ou dedicado ao dispositivo ou sistema que ele controla. Diferente de computadores de propósito geral, como o computador pessoal, um sistema embarcado realiza um conjunto de tarefas predefinidas, geralmente com requisitos específicos (EMBEDDED ARCHITECTS, 2013).

Segundo Taurion (2005, p. 14), em praticamente todas as atividades humanas é possível identificar a presença de softwares embarcados, embora a grande maioria deles passe despercebida. Os exemplos são muitos, e no dia-a-dia os mesmos são usados em tarefas cotidianas, como celulares, automóveis (sistemas de freio, por exemplo), catracas eletrônicas e elevadores inteligentes.

O núcleo de um sistema embarcado é o seu microcontrolador, que é o processador, a memória, e as interfaces de entrada e saída todas dentro de um mesmo chip, tornando muito mais simples a construção de um sistema. Softwares escritos para esse tipo de sistema são chamados de *firmware*, e armazenados em uma memória ROM ou memória *flash*. Muitas vezes o sistema é executado com recursos computacionais limitados: sem teclado, sem tela e com pouca memória (GERMANO, 2013).

Sistemas embarcados que geralmente contam com uma quantidade reduzida de recursos como memória, poder de processamento e outros requisitos não são projetados para utilizar sistemas operacionais como os de computadores pessoais. Comumente, são utilizados

sistemas operacionais que além de consumirem muito menos memória e processamento, são muito mais estáveis e confiáveis (EMBEDDED ARCHITECTS, 2013).

De acordo com Taurion (2005, p. 15), os sistemas operacionais são o cerne dos sistemas embarcados, pois em cima deles é que são construídas as funcionalidades dos artefatos. O sistema operacional Linux tem grande oportunidade de expansão no setor de software embarcado. A seguir são listadas algumas razões disso, segundo Taurion (2005, p. 16):

a) custo de entrada baixo e poucas restrições de direito autoral: pode ser baixado gratuitamente da internet e distribuído livremente, sem restrições. Não existem *royalties* a serem pagos pelo uso do Linux embarcado nos milhões de produtos a serem vendidos no mercado consumidor. Um comprador de uma geladeira não estará pagando *royalties* pela funcionalidade embutida proporcionada pelo sistema operacional;
b) confiabilidade: é um sistema confiável e estável, importante em um ambiente onde a atualização ou inserção de *patches* não é uma tarefa simples;
c) flexibilidade: pode ser customizado às necessidades específicas de cada aplicação. Fundamental em ambiente embarcado, onde cada dispositivo demanda características funcionais diferentes entre si. Um sistema eletrônico de injeção de combustível em um motor de automóvel não precisa das mesmas características operacionais de um forno de micro-ondas doméstico, por exemplo;
d) disseminação de conhecimento: pelo acesso ao fonte do Linux o desenvolvedor pode não apenas estudar profundamente os algoritmos implementados no *kernel*, mas fazer experiências e testes com variações de código. O aprendizado em software baseia-se fundamentalmente na prática, facilidade que o Linux e o conceito de software livre permitem e incentivam;
e) disponibilidade de plataformas: o Linux é executado em praticamente todas as plataformas de sistemas embarcados existentes e, pelo acesso ao seu código fonte, permite uma implementação mais rápida em qualquer nova plataforma que venha a existir. Permite, portanto, maior flexibilidade dos projetos no referente à escolha de hardwares.

2.4. TECNOLOGIA 3G

A 3G é a terceira geração de tecnologia de telefonia móvel e permite a transmissão de dados em alta velocidade sem a utilização de qualquer cabo ou rede Wi-fi. Atualmente, esta tecnologia é usada principalmente por celulares e *tablets* que podem ter acesso à internet e fornecer a visualização de conteúdos como notícias, vídeos, músicas, entre outros (CÂMARA, 2012).

Apesar de grande parte dos *smartphones* e *tablets* atuais possuir compatibilidade com a tecnologia de terceira geração, para fazer uso da mesma é necessário que o usuário contate um plano de alguma empresa de telefonia. A associação do usuário ao plano é dada através de um número de telefone celular que está vinculado a um *chip*, popularmente conhecido como *Subscriber Identity Module* (SIM) card. Os modems 3G oferecidos para uso desta tecnologia em computadores pessoais requerem o mesmo processo para sua utilização. É importante ressaltar também que a tecnologia 3G também pode ser usada fora do país do usuário, contudo, nessas condições é necessário solicitar a operadora o serviço de *roaming* internacional (TURRM, 2011).

Segundo Morimoto (2011), a tecnologia 3G utiliza o *Universal Mobile Telecommunications System* (UMTS), ou simplesmente sistema UMTS. Este sistema permite o uso de dois protocolos de transporte, que são o *Wide-Band Code-Division Multiple Access* (WCDMA) e o *High-Speed Downlink Packet Access* (HSDPA) e são usados de acordo com a disponibilidade, qualidade da recepção e do modo suportado pelo equipamento.

O protocolo WCDMA oferece taxas de transmissão de até 384 *quilobits* por segundo (kbits) e oferece tempos de latência muito melhores e uma conexão muito mais utilizável. Já o protocolo HSDPA, reduz a latência e aumenta a taxa de *download* da rede de forma expressiva. Usando o HSDPA como protocolo de transporte, o UMTS suporta taxas de 1.8, 3.6, 7.2 e 14.4 *megabits*. Evidentemente, a velocidade real varia de acordo com a qualidade do sinal e o número de usuários conectados à mesma estação de transmissão, mas ela é sempre bem mais alta que no WCDMA (MORIMOTO, 2011).

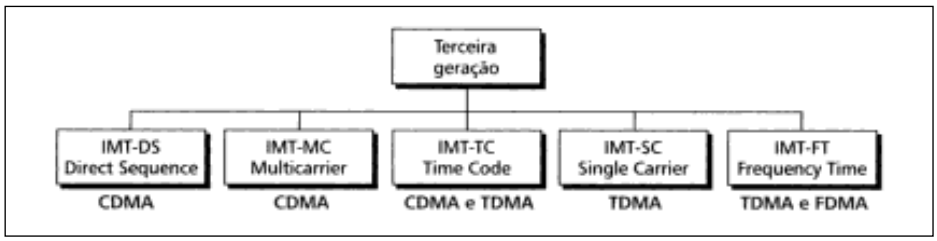
De acordo com Forouzan (2004, p. 378), o conceito de 3G surgiu em 1992, quando a União Internacional de Telecomunicações (UIT) publicou uma proposta de projeto chamada *Internet Mobile Communication for year 2000* (IMT-2000). O documento definiu os seguintes critérios para a tecnologia 3G:

a) qualidade de voz comparável à dos serviços de telefonia fixo;

b) taxas de transmissão de dados de 144 kbits para acesso a partir de veículo móvel (carros), 384 kbits para acesso partindo de usuário em movimento (pedestres) e 2 *megabits* por segundo (Mbps) para usuários fixos (casa ou escritório);
c) suportar serviços de comutação de pacotes e comutação de circuitos;
d) banda total de 2 gigahertz (GHz);
e) largura de banda de 2 megahertz (MHz);
f) interface com a internet.

A Figura 1 apresenta as interfaces de rádio (padrão *wireless*) adotadas pelo IMT-2000. As cinco interfaces foram geradas a partir das tecnologias de segunda geração. As duas primeiras evoluíram da tecnologia *Code Division Multiple Access* (CDMA). A terceira evoluiu simultaneamente da combinação CDMA e da combinação *Time Division Multiple Access* (TDMA). A quarta evoluiu da TDMA e a última evoluiu tanto da *Frequency Division Multiple Acces* (FDMA) quanto da TDMA (FOROUZAN, 2004, p. 379).

Figura 1 - Interfaces de rádio do padrão IMT-2000



Fonte: Forouzan (2004, p. 379).

Forouzan (2004, p. 378), também previu que quando a terceira geração de telefonia estivesse disponível, ela proporcionaria tanto comunicação de dados, como comunicação de voz digitalizada. Além disso, os usuários poderiam fazer *downloads*, assistir filmes, navegar ou jogar na internet e manter seus dispositivos portáteis sempre conectados, ou seja, não seria necessário discar um número para estar conectado à internet.

Segundo o *site* de notícias Olhar Digital (2013), a maioria dos usuários de internet móvel, inclusive os usuários brasileiros, encontra-se atualmente na tecnologia 3G. Segundo o padrão do IMT-2000, a rede de terceira geração oferece velocidades mínimas de 200 kbits, mas ainda promete velocidades muito superiores. Contudo, a tecnologia de quarta geração (4G), é a novidade do momento, e no Brasil, as operadoras de celular estão correndo contra o tempo para conseguir cumprir os prazos impostos pela Agência Nacional de Telecomunicações (Anatel) para implantação da tecnologia no país.

2.5. SISTEMA GPS

O Sistema de Posicionamento Global foi desenvolvido pelo Departamento de Defesa americano, com o intuito de que os Estados Unidos pudessem assim orientar suas movimentações aéreas, bombardeios e lançamentos de mísseis. Este sistema é mantido pelo Departamento de Defesa estadunidense e conta com um total de 24 satélites e mais 4 sobressaltantes, em 6 planos orbitais, a uma altitude de 19.000 quilômetros (km).

> O GPS é considerado, atualmente, a mais moderna e precisa forma de determinação da posição de um ponto na superfície terrestre. O receptor capta os sinais de quatro satélites para determinar as suas próprias coordenadas - e depois calcula a distância entre os quatro satélites pelo intervalo de tempo entre o instante local e o instante em que os sinais foram enviados. (DECICINO, 2009).

Segundo Dilão (2013), a infraestrutura tecnológica associada ao sistema GPS é formada por três subsistemas:

a) subsistema de satélites: é constituído pelos 24 satélites que percorrem o equivalente a duas voltas no planeta Terra por dia. As órbitas dos satélites foram escolhidas de modo que, de qualquer ponto da Terra, possa-se ver entre 5 e 8 satélites. Para calcular com precisão a posição de determinado objeto/pessoa, basta apenas receber em boas condições o sinal de quatro destes satélites;
b) subsistema de controle: é formado por várias estações terrestres, nestas são observadas as trajetórias dos vários satélites GPS e o tempo é atualizado com precisão. Esta informação é transmitida aos satélites e com esses dados, o sistema informático recalcula e corrige em cada um dos mesmos a posição absoluta do respectivo satélite que é enviada para a Terra;
c) subsistema do utilizador: é composto por um receptor de rádio com uma unidade de processamento capaz de decifrar em tempo real a informação enviada por cada satélite e calcular a posição. Cada satélite envia sinais de características diferentes em intervalos de 30 em 30 segundos e de 6 em 6 segundos. Para haver uma determinação precisa da posição são necessários pelo menos de 12 minutos e 30 segundos de boa recepção dos vários tipos de sinais enviados.

A Figura 2 apresenta o sistema GPS.

Figura 2 - Sistema GPS

Fonte: Portogente (2013).

2.6. PROTOCOLO TCP/IP

Segundo Sportack (2007, p. 4), “o TCP/IP é um conjunto de mecanismos de comunicação de dados, embutidos em software, que permitem que você use a Internet e inúmeras redes privadas.”.

Sportack (2007, p. 4) ainda explica que os mecanismos, na verdade são protocolos que são projetados para executar uma função específica. O TCP se concentra no processamento e manipulação de dados e aplicativos, por isso é chamado de Protocolo de Controle de Transmissão (*Transmission Control Protocol*). Quanto ao IP, é orientado e projetado para acomodar a transmissão e recebimento de dados pela rede. É chamado de Protocolo da Internet (*Internet Protocol*).

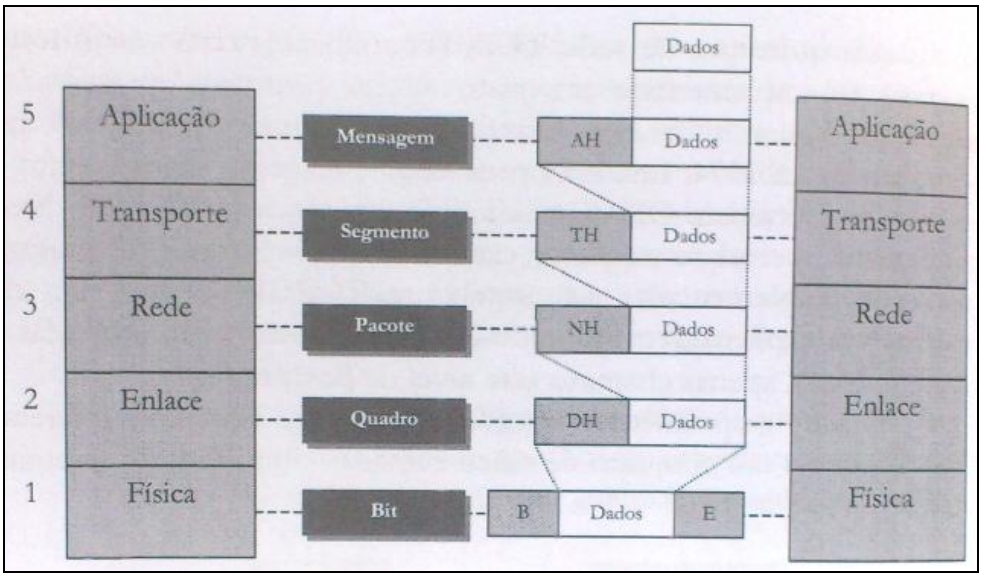
O TCP/IP surgiu em 1 de Janeiro de 1983, no *Department of Defense/Advanced Research Project Agency* (DOD/ARPA), nos Estados Unidos. Nesta data, haviam aproximadamente 400 computadores conectados a uma rede denominada ARPAnet, que começaram a se comunicar uns com os outros com uma coleção de protocolos informalmente

conhecidos como TCP/IP (SPORTACK, 2007, p. 5).

O TCP/IP possui um modelo de arquitetura que é um conjunto de camadas, onde cada camada representa um grupo de tarefas específicas e facetas da comunicação. Ressalta-se que o modelo TCP/IP é teórico, portanto as camadas não existem fisicamente e não realizam qualquer função. As implementações de protocolo, na verdade são uma combinação de hardware e software que realizam as funções associadas com as camadas correspondentes (SCRIMGER et al., 2002, p. 33).

A Figura 3 apresenta o modelo de referência TCP/IP.

Figura 3 - Modelo de Referência TCP/IP



Fonte: Péricas (2012, p. 22).

A comunicação entre as camadas do modelo de referência TCP/IP inicia na camada de mais alto nível, que é camada de aplicação. Esta contém os mecanismos de suporte às aplicações de alto nível, como por exemplo, os protocolos SMTP para encaminhamento de mensagens de correio eletrônico e HTTP para navegação na internet. A camada seguinte é a de transporte, que permite que as entidades da origem e do destino mantenham um canal de comunicação. Na sequência, a camada de rede entrega os pacotes emitidos a qualquer destino, independentemente das tecnologias de transmissão utilizadas. Então a camada de enlace permite que quadros enviados pela camada de rede sejam transportados livres de erros. Finalmente, a camada física trata a transmissão de *bits* através de um canal de comunicação (PÉRICAS, 2012, p. 22).

2.7. PROTOCOLO HTTP

Segundo Péricas (2012, p. 171), o *Hypertext Transfer Protocol* (HTTP), definido na *Requests for Comments* (RFC 2616), é o protocolo de aplicação utilizado para transferência de páginas internet entre sistemas computacionais.

De acordo com Forouzan (2010, p. 609), o protocolo HTTP é um protocolo usado principalmente para acessar dados da internet e funciona como uma junção dos protocolos *File Transfer Protocol* (FTP) e *Simple Mail Transfer Protocol* (SMTP). É similar ao protocolo FTP porque transfere arquivos e usa os serviços do protocolo *Transmission Control Protocol* (TCP). Porém, é mais simples do que o FTP, pois usa apenas uma conexão TCP, não há conexão de controle separada, somente dados são transferidos entre o cliente e o servidor.

O protocolo HTTP é fundamentado em requisições e respostas entre clientes e servidores. O cliente, que pode ser um navegador ou dispositivo que fará a requisição, solicita um determinado recurso enviando um pacote de informações, contendo alguns cabeçalhos, a uma *Uniform Resource Locator* (URL). O servidor recebe essas informações e envia uma resposta, que pode ser um recurso ou outro cabeçalho (VIEIRA, 2007).

Uma mensagem HTTP de requisição é composta por uma linha inicial denominada linha de requisição, várias linhas adicionais contendo opções específicas para a requisição e uma linha final em branco. Enquanto uma mensagem HTTP de resposta é composta por uma linha inicial denominada linha de *status*, várias linhas adicionais contendo opções específicas para a resposta, uma linha em branco e o conteúdo do objeto propriamente dito (PÉRICAS, 2012, p. 192).

O Quadro 1 apresenta os sete tipos de mensagens de requisição a um servidor de internet que o protocolo HTTP especifica.

Quadro 1 - Tipos de mensagens de requisição

Comando	**Significado**
GET	Solicita um arquivo
HEAD	Solicita apenas o cabeçalho da mensagem de resposta sem o arquivo
PUT	Envia um arquivo
POST	Anexa dados a um arquivo no servidor.

DELETE	Exclui um arquivo no servidor.
TRACE	Envia de volta a requisição com propósito de depuração.
OPTIONS	Permite consultar o servidor sobre suas propriedades ou sobre as de um arquivo.

Fonte: Péricas (2012, p. 193).

2.8. TRABALHOS CORRELATOS

Beszczynski (2008) desenvolveu um protótipo de um sistema de rastreamento veicular baseado no módulo Telit para conclusão do curso de Ciência da Computação na Universidade Regional de Blumenau.

O principal objetivo do trabalho de Beszczynski (2008) era oferecer uma nova possibilidade de localização para diversos fins, como ferramenta para auxiliar na tomada de decisão de empresas de transportes ou para localizar um automóvel ou carga que fora roubada e até mesmo para auxiliar autoridades na localização de rotas de quadrilhas de roubo de veículos, bem como na identificação do motorista.

Beszczynski (2008) desenvolveu um rastreador que registrava as informações coletadas em uma base de dados e disponibilizava as mesmas para consultas em uma página *web*. O protótipo fazia conexão através de um Serviço de Rádio de Pacote Geral (*General Packet Radio Service* – GPRS) com um servidor usando um modem interno.

De acordo com Prass (2011), o software Prey Project é um aplicativo de código aberto que permite rastrear computadores que tenham sido roubados ou perdidos. A localização do equipamento em questão fica disponível em um serviço na internet, em que o usuário deve estar previamente cadastrado.

O aplicativo, depois de instalado no computador, passa a enviar a localização do mesmo para um painel de monitoramento, caso o usuário tenha registrado que o dispositivo foi roubado ou perdido, no *site*. Esta localização aparece no painel de monitoramento através de mapas do Google Maps (PRASS, 2011).

É importante ressaltar que o rastreamento através deste aplicativo só é possível se o dispositivo, após ter sido roubado ou perdido, for conectado à internet e o Prey Project ainda estiver instalado no mesmo.

3 DESENVOLVIMENTO DO PROTÓTIPO

Neste capítulo são apresentadas as particularidades do protótipo através das sub-seções levantamento de informações, especificação e implementação. Além disso, também são mostrados os resultados obtidos e discussão sobre os mesmos.

3.1. LEVANTAMENTO DE INFORMAÇÕES

O protótipo de sistema de rastreamento para computadores foi desenvolvido com a intenção de auxiliar usuários a encontrar seus respectivos computadores que tenham sido perdidos ou roubados.

Para melhor organização e compreensão dividiu-se o protótipo em duas partes: uma aplicação embarcada em uma placa (hardware) e uma aplicação *web*, onde o usuário pode visualizar a localização de seu computador mediante cadastro pessoal e do próprio computador.

O software embarcado tem como principal objetivo rastrear a localização atual do computador. Isso é feito através de um módulo que possui um Sistema de Posicionamento Global (*Global Positioning System* – GPS) e está ligado a placa utilizada. As coordenadas encontradas são enviadas para o servidor da aplicação *web* por meio de uma conexão 3G e registradas na base de dados da aplicação *web*.

A aplicação *web* é a interface de interação que o usuário possui com o protótipo. É através dessa aplicação que o usuário acompanhará os locais em que seu computador esteve, bem como ativará e desativará notificações por *e-mail* de novas localizações. Podem ser cadastrados vários computadores para o mesmo usuário desde que cada computador possua sua respectiva placa.

Para a construção da aplicação embarcada e da aplicação *web*, foram utilizadas as respectivas ferramentas:

a) IDLE Python para codificação do software embarcado;
b) Terminal RealTerm para interação com o sistema operacional da placa;
c) MySQL com banco de dados para armazenar as informações obtidas através da aplicação *web*;

d) JavaServer Faces 2.0 (JSF2), Hibernate e JBoss para desenvolvimento da aplicação *web*;
e) Sparx Systems Enterprise Architect para modelagem de diagramas;
f) ferramenta MySqlWorkbench para criação do Modelo Entidade Relacionamento (MER).

3.2. ESPECIFICAÇÃO

Esta seção apresenta os requisitos do protótipo, diagramas de casos de uso, bem como o diagrama de atividades, e o modelo de entidade relacionamento. A especificação dos principais casos de uso está descrita no Apêndice A.

3.2.1 Requisitos do Protótipo

Para melhor compreensão dos requisitos do protótipo, dividiu-se o mesmo em dois módulos:

a) módulo da aplicação *web*: refere-se à parte responsável por disponibilizar, visualmente para o usuário, a localização do computador. É composto por páginas *web* e servidor;
b) módulo da aplicação embarcada: refere-se à parte responsável por coletar as coordenadas da posição do computador e enviá-las ao servidor. É composto pelo rastreador (placa, módulo GPS e modem 3G) e software embarcado.

O Quadro 2 apresenta os requisitos funcionais previstos para o módulo da aplicação *web* e sua rastreabilidade, ou seja, vinculação com o(s) caso(s) de uso associado(s).

Quadro 2 - Requisitos Funcionais do módulo da aplicação web

Requisitos Funcionais	Caso de Uso
RF01: O sistema permitirá ao usuário manter-se.	UC01.01
RF02: O sistema permitirá ao usuário consultar os lugares em que o	UC02.01

computador esteve.	
RF03: O sistema permitirá ao usuário ativar/desativar notificações.	UC03.01
RF04: O sistema permitirá enviar notificações por *e-mail*.	UC04.01
RF05: O sistema permitirá ao usuário manter computadores.	UC05.01
RF06: O sistema permitirá armazenar as coordenadas recebidas no servidor.	UC06.01

O Quadro 3 apresenta os requisitos funcionais previstos para o módulo da aplicação embarcada e sua rastreabilidade, ou seja, vinculação com o(s) caso(s) de uso associado(s).

Quadro 3 - Requisitos Funcionais do módulo da aplicação embarcada

Requisitos Funcionais	Caso de Uso
RF01: O sistema permitirá coletar coordenadas da localização do computador.	UC01.02
RF02: O sistema permitirá enviar as coordenadas para o servidor.	UC02.02

O Quadro 4 lista os requisitos não funcionais previstos para o módulo da aplicação *web*.

Quadro 4 - Requisitos Não Funcionais do módulo da aplicação *web*

Requisitos Não Funcionais
RNF01: O sistema deverá ser executado a partir do servidor de aplicação JBoss 7.1.
RNF02: O sistema deverá utilizar o *framework* Hibernate 4.2.1 para persistência e manipulação de seus dados.
RNF03: O sistema deverá utilizar o *framework* JavaServer Faces 2.0 (JSF2) para a construção de interfaces.
RNF04: O sistema deverá utilizar banco de dados MySQL 5.6.

O Quadro 5 lista os requisitos não funcionais previstos para o módulo da aplicação embarcada.

Quadro 5 - Requisitos Não Funcionais do módulo da aplicação embarcada

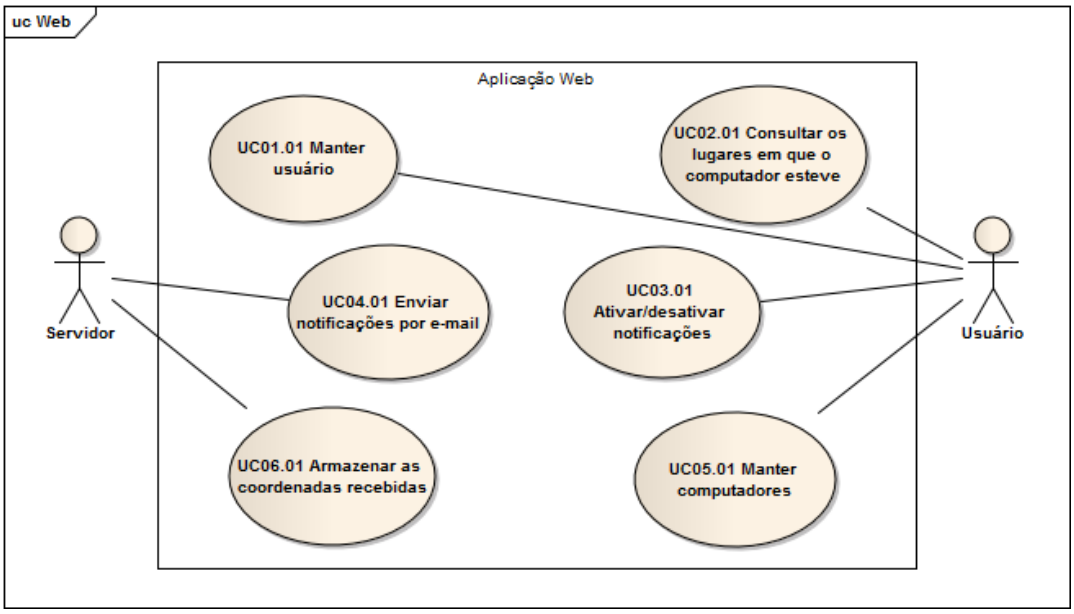
Requisitos Não Funcionais
RNF01: O sistema deverá utilizar as tecnologias UMTS, WCDMA e HSPA da rede de terceira geração (3G) para envio das coordenadas encontradas pelo GPS ao servidor.
RNF02: O sistema deverá utilizar linguagem de programação Python 2.7 para codificação do software embarcado.
RNF03: O sistema deverá utilizar sistema operacional Linux, distribuição Debian, versão Wheezy na placa para executar o software embarcado.

3.2.2 Diagramas de Casos de Uso

Esta subseção apresenta os diagramas de caso de uso necessários para o entendimento do funcionamento do protótipo, sendo que as descrições dos mesmos estão no Apêndice A.

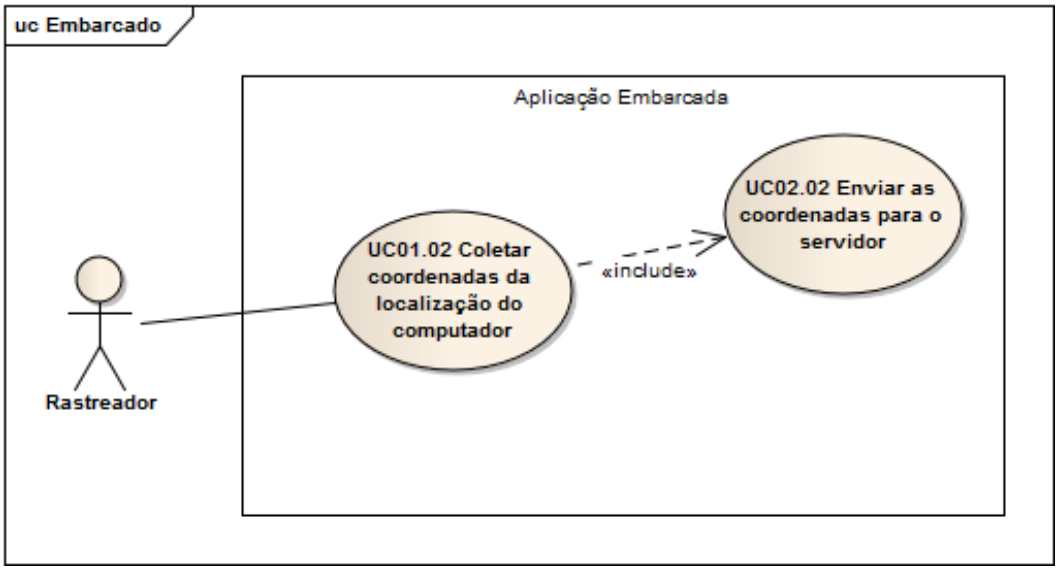
A Figura 4 apresenta as atividades executadas pelo usuário e pelo servidor no módulo da aplicação *web*.

Figura 4 - Diagrama de Casos de Uso do módulo da aplicação *web*



Na Figura 5 têm-se as atividades executadas pela aplicação embarcada.

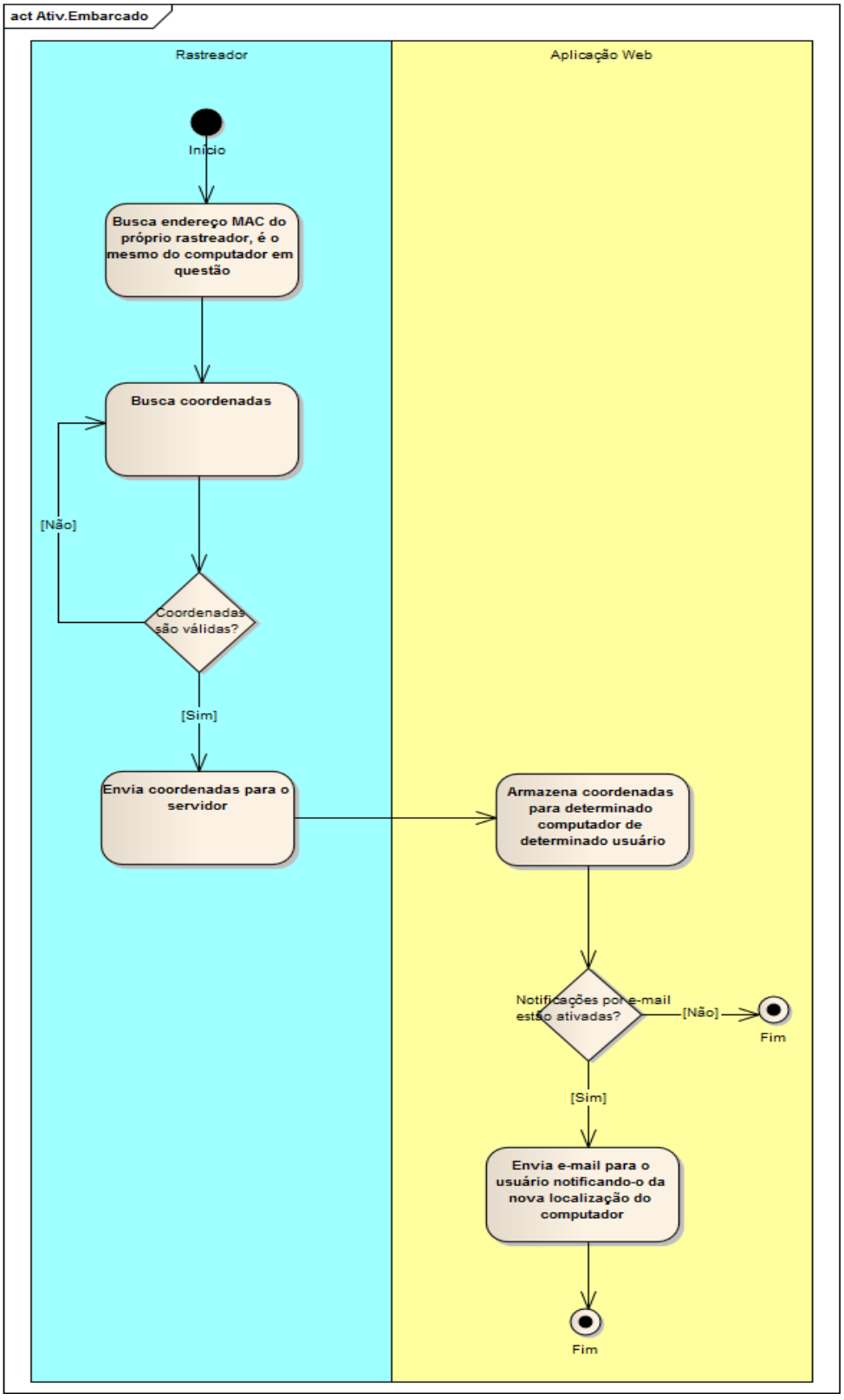
Figura 5 - Diagrama de Casos de Uso da aplicação embarcada



3.2.3 Diagrama de Atividades

A Figura 6 apresenta o diagrama de atividades que descreve o processo de rastreamento de determinado computador. Inicialmente, o rastreador busca o próprio endereço físico (*Media Access Control* – MAC), que é o mesmo endereço da placa de rede do computador rastreado. Em seguida, o rastreador busca as coordenadas do local atual e verifica se as coordenadas estão válidas visto que as mesmas podem não existir se o GPS não estiver recebendo sinal. Na sequência, se as coordenadas forem válidas são enviadas para o servidor da aplicação *web*, senão o rastreador continua buscando-as. Ao armazenar as coordenadas no servidor, a aplicação *web* verificará se o usuário ativou a opção de ser notificado por *e-mail* sobre novas localizações. Se as notificações por *e-mail* estiverem ativas, o usuário receberá um *e-mail* sobre a nova localização, além de também ter essa informação disponível na página *web* do protótipo.

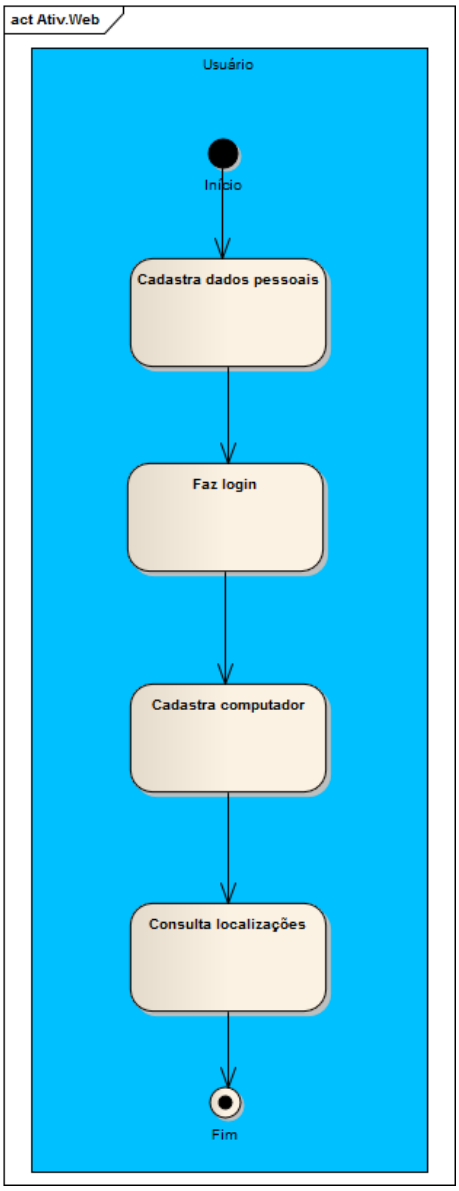
Figura 6 - Diagrama de atividades rastreador



A Figura 7 apresenta o diagrama de atividades que expõe as funcionalidades da aplicação *web*. Para ter acesso às localizações encontradas pelo rastreador, o usuário,

primeiramente, cadastra seus dados pessoais e a partir de então tem acesso à página *web* do protótipo através do *e-mail* e senha cadastrados. Após fazer *login*, o usuário cadastra o computador desejado que será rastreado e a partir do momento que houver localizações registradas para tal computador, o usuário poderá visualizá-las na página *web*.

Figura 7 - Diagrama de atividades aplicação *web*

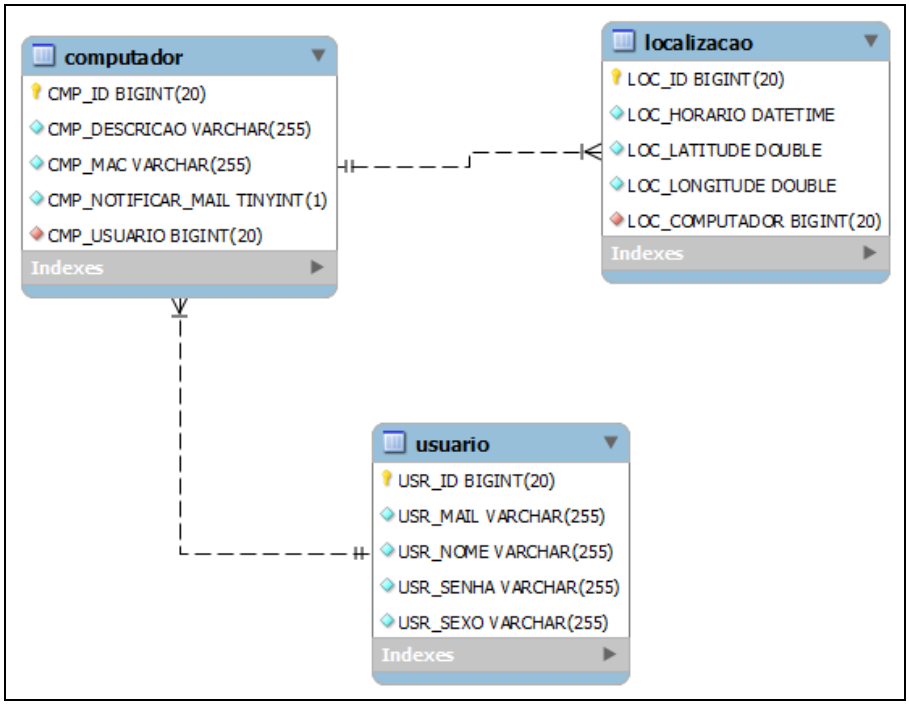


3.2.4 Modelo de Entidade e Relacionamento

A Figura 8 define o diagrama de entidade e relacionamento que contém as entidades que serão persistidas no banco de dados. O dicionário de dados está sendo detalhado no

Apêndice B.

Figura 8 - Diagrama de entidade e relacionamento



A seguir, apresenta-se resumidamente a descrição das entidades que compõem o protótipo:

a) computador: entidade responsável por armazenar os computadores cadastrados através da aplicação *web*;
b) localização: entidade responsável por armazenar as localizações de determinado computador enviadas pelo rastreador;
c) usuário: entidade responsável por armazenar os usuários do protótipo.

3.3. IMPLEMENTAÇÃO

A seguir são mostradas as técnicas e ferramentas utilizadas e a operacionalidade da implementação.

3.3.1 Técnicas e ferramentas utilizadas

O software embarcado foi desenvolvido utilizando a linguagem de programação Python através do interpretador IDLE Python. A placa empregada para servir de rastreador foi a Terra G25, da Acme Systems, empresa italiana especializada no desenvolvimento de projetos, produção e montagem de placas de microprocessadores de baixo custo, desenvolvidos para prototipagem rápida em ambientes públicos e industriais (ACME SYSTEMS, 2013, tradução nossa).

No desenvolvimento da aplicação *web* foram usados os *frameworks* JavaServer Faces 2.0 (JSF2) e Hibernate, servidor de aplicação JBoss e o banco de dados MySQL.

3.3.1.1 Linguagem de Programação Python e IDLE Python

A linguagem de programação Python é uma linguagem interpretada, isto é, seu código é executado pelo interpretador e não compilado para uma linguagem de máquina, para então ser executado no sistema, como acontece em outras linguagens. Uma grande vantagem disso é a possibilidade de serem realizados testes com partes do código no interpretador sem criar-se um código e compilá-lo, para então testá-lo (ALVES, 2009).

De acordo com Lutz e Ascher (2007, p. 31), o Python é uma linguagem de programação frequentemente aplicada em funções de *script* e é comumente definida como uma linguagem de *script* orientada a objetos.

A linguagem de programação Python começou a ser desenvolvida em dezembro de 1989 por Guido van Rossum, em um Centro para Matemática e Ciência da Computação (CWI), localizado em Amsterdam. Na época, também ocorreu o desenvolvimento de várias outras linguagens de programação dinâmicas como *Tool Command Language* (TCL) e Ruby (DRUMOND, 2009).

Ainda segundo Lutz e Ascher (2007, p. 64), o IDLE é uma *Graphical User Interface* (GUI) que permite editar, executar, navegar e depurar programas em Python partindo de uma única interface. Para muitas pessoas, o IDLE é uma alternativa fácil de usar para digitar linhas de comando e uma alternativa menos propensa a erros para clicar em ícones.

Neste trabalho, foi utilizada a versão 2.7 do Python para desenvolver o software embarcado, que na verdade é um *script* que é executado dentro do rastreador e é responsável

por identificar o endereço MAC do próprio, validar as coordenadas recebidas através do GPS que está junto ao rastreador e, se válidas, enviar tais coordenadas ao servidor da aplicação *web* através da biblioteca *requests* do Python.

Requests é uma biblioteca para envio e recebimento de informações utilizando o protocolo HTTP através de uma URL, licenciada sob o servidor HTTP Apache2 e escrita em Python (STUMM JUNIOR, 2013).

A Figura 9 apresenta parte do código do software embarcado. Na linha 205, é chamado o método que busca o endereço MAC do próprio rastreador. O trecho que se inicia a partir da linha 207 e termina na linha 216 é responsável por configurar a comunicação com o GPS que está integrado ao rastreador, através da biblioteca *serial* do Python. Na continuação, o *script* entra em uma estrutura de repetição, a partir da linha 224, sem fim previsto. Esta é a parte de controle e é a mais importante do *script*, pois é dentro desta estrutura que é chamado o método que busca as coordenadas retornadas pelo GPS – linhas 234 e 236, valida as mesmas, linhas 235 e 237, e faz uma requisição HTTP ao servidor a cada trinta minutos, enviando as coordenadas através de uma URL, linha 253. As requisições são feitas a cada trinta minutos porque o GPS tem pequenas variações nos valores das coordenadas que não são suficientes para considerar-se que esta é uma nova localização, ou seja, não é necessário registrar várias vezes o mesmo lugar em um intervalo de tempo tão curto.

Figura 9 - Parte do código fonte do software embarcado

```
end_mac = get_macaddress_linux('eth0')

# configuracao para comunicacao com GPS
serialdevice = "/dev/ttyS1"
ser = serial.Serial(
  port=serialdevice,
  baudrate=4800,
  timeout=1,
  parity=serial.PARITY_NONE,
  stopbits=serial.STOPBITS_ONE,
  bytesize=serial.EIGHTBITS)

# limpa o buffer de entrada descartando tudo o que esta no buffer
ser.flushInput()

ultima_latitude = '0'
ultima_longitude = '0'

while 1:
    posicao_atual = get_coordenadas(ser)

    if  not posicao_atual:
        continue

    try:
        # se o GPS estiver sem sinal, as coordenadas estarao vazias
        if (posicao_atual[0] != '') and (posicao_atual[2] != '') :
            if  (posicao_atual[0] != ultima_latitude) or (posicao_atual[2] != ultima_longitude) :
                ultima_latitude = posicao_atual[0]
                l_latitude_ofc = calcula_latitude(ultima_latitude)
                ultima_longitude = posicao_atual[2]
                l_longitude_ofc = calcula_longitude(ultima_longitude)
                print(l_latitude_ofc)
                print(l_longitude_ofc)

                if  posicao_atual[1] == 'S':
                    l_latitude_ofc = '-' + l_latitude_ofc
                print(l_latitude_ofc)
                if  posicao_atual[3] == 'W':
                    l_longitude_ofc = '-' + l_longitude_ofc
                print(l_longitude_ofc)

                # montar url a ser passada para a aplicacao
                link = 'http://201.54.196.2:5999/xereta/pages/modulos/mdLocalizacao/novaLocalizacao.xhtml?mac='+end_mac
                     +'&latitude='+l_latitude_ofc+'&longitude='+l_longitude_ofc
                print '>>>', link

                xereta = requests.get(link, verify=False)
```

3.3.1.2 Placa Terra G25

Inicialmente foi definido que a placa que seria utilizada para execução deste projeto seria a Aria G25. Contudo devido a motivos ainda não identificados, a placa teve seu processador queimado, o que impossibilitou completamente o uso da mesma. Então, adquiriu-se temporariamente a placa Terra G25.

Fabricada e vendida pela empresa italiana Acme Systems, a placa Terra foi idealizada para prototipagem rápida e produção final de dispositivos personalizados, baseada no sistema

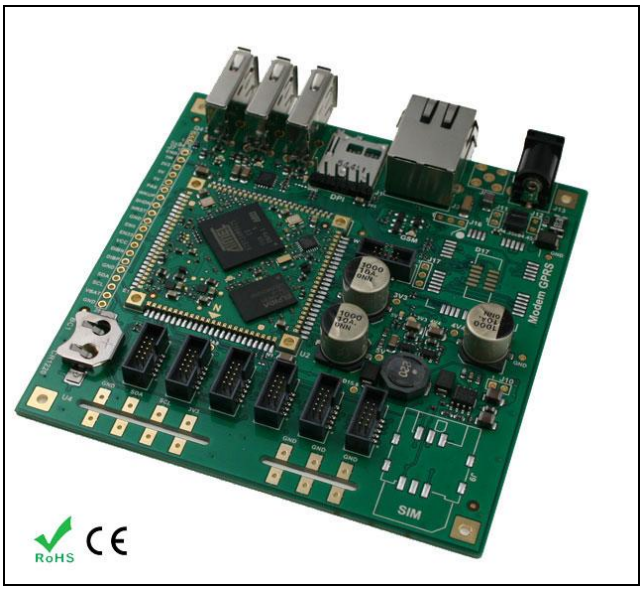
operacional Linux embarcado compilado para arquitetura ARM (ACME SYSTEMS, 2013, tradução nossa).

A Terra G25 possui as seguintes características:

a) módulo Aria G25 (System-on-Module) SoM com ARM9 400Mhz e 256 *megabytes* (Mbyte) de memória *Random Access Memory* (RAM);
b) três portas USB 2.0 com controle de alimentação separado;
c) suporte para microSD bootável com Debian Linux embarcado compilado para arquitetura ARM;
d) porta *Ethernet*;
e) dimensões da placa: 10x10 centímetros;
f) energia prolongada, faixa de entrada de alimentação de 9 a 28 Volts.

A Figura 10 apresenta a placa Terra G25 e seus componentes.

Figura 10 - Placa Terra G25

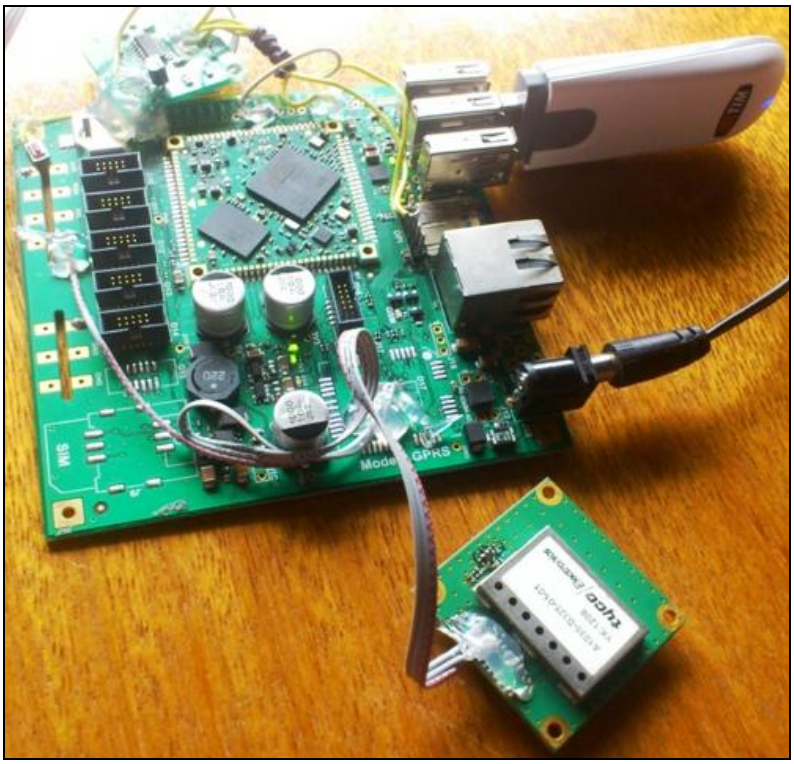


Fonte: ACME SYSTEMS (2013).

Na construção desse projeto, para fazer com que a placa obtivesse localizações, foi adicionado à mesma um módulo GPS, modelo A1035-D da Tyco Electronics. Para envio das coordenadas, detectadas pelo GPS, ao servidor da aplicação *web* usou-se um modem 3G Huawei E330S conectado a placa via USB e para manter a placa ligada usou-se uma bateria de chumbo, de 12V por 2,3A. Inicialmente definiu-se que seria utilizada a tecnologia GSM para comunicação com o servidor da aplicação *web*, contudo optou-se pela tecnologia 3G por esta ser mais recente e oferecer um pouco mais de agilidade.

A Figura 11 mostra a placa Terra G25 com os componentes adicionais – GPS e modem 3G, necessários para alcançar os objetivos desse projeto.

Figura 11 – Placa Terra G25 com GPS e Modem 3G



Fonte: Autoria própria.

3.3.1.3 *Framework* JavaServer Faces 2.0 (JSF2)

Segundo Paganini (2010, p. 22), "*JavaServer Faces* (JSF) é uma especificação técnica do *Java Community Process* (JCP), publicada em 2004, com o objetivo de padronizar um *framework* para desenvolvimento da camada de apresentação em aplicações web.".

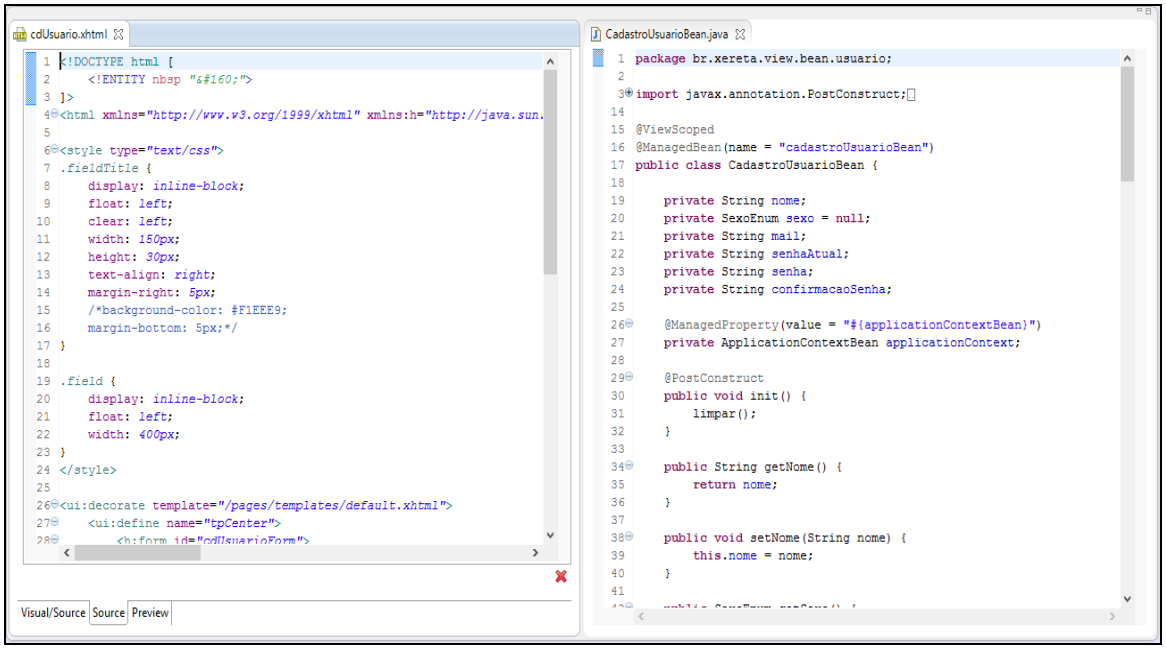
Esta especificação busca maximizar a produtividade no desenvolvimento de aplicações *web*, minimizar a complexidade de manutenção, possibilitar melhor integração com outras tecnologias *web*, dentre outros objetivos. Além disso, também oferece uma infraestrutura para criação de componentes interativos integrados a todas as soluções oferecidas pela especificação, por exemplo: conversores, validadores, eventos, vínculo a dados e a métodos (PAGANINI, 2010, p. 23).

Godoy (2011), também conceitua JSF como um *framework* que permite a elaboração

de interfaces de usuário *web* dispondo componentes em um formulário e ligando-os a objetos Java, permitindo assim a separação entre lógica e regras de negócio, navegação, conexões com serviços externos e gerenciamento de configurações. O ponto forte deste *framework* é o grande número de componentes e o *design* muito flexível que o permitiu crescer acomodando novas tecnologias.

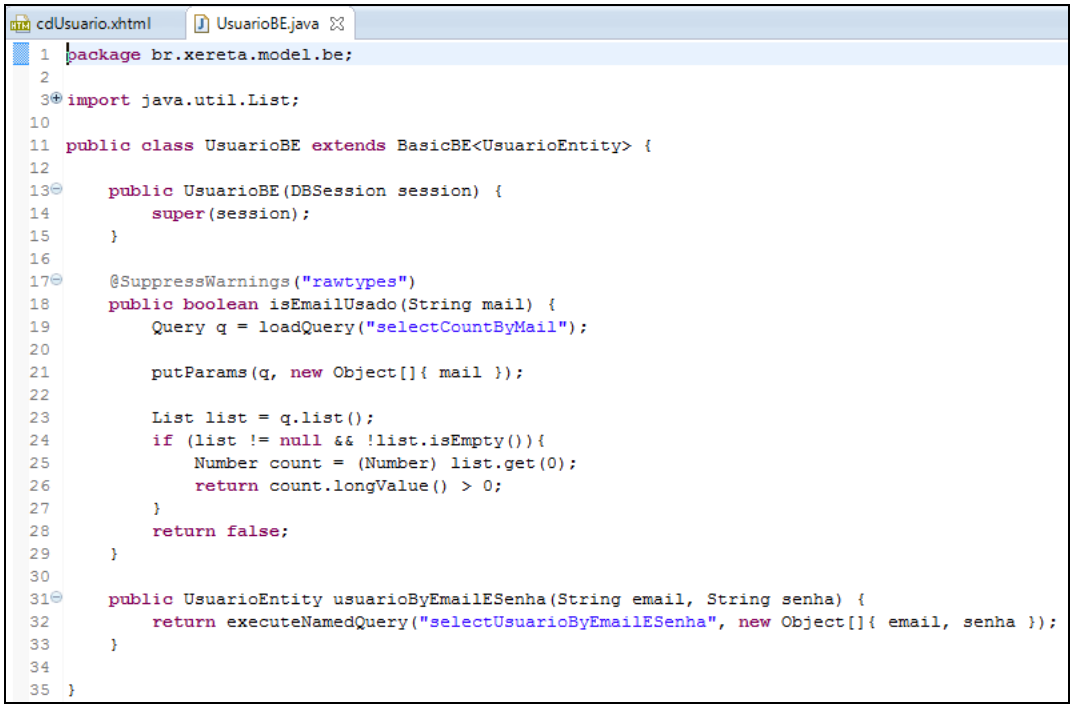
Em JSF, a tela é no formato de *xhtml* e para que ela consiga recuperar informações é necessário utilizar o *managed bean* que é uma classe responsável por intermediar a comunicação entre a tela e as classes do modelo. Na Figura 12 apresenta-se, à direita, a tela do cadastro de usuário e à esquerda encontra-se a classe CadastroUsuarioBean.

Figura 12 - Tela do Cadastro de Usuário em JSF2



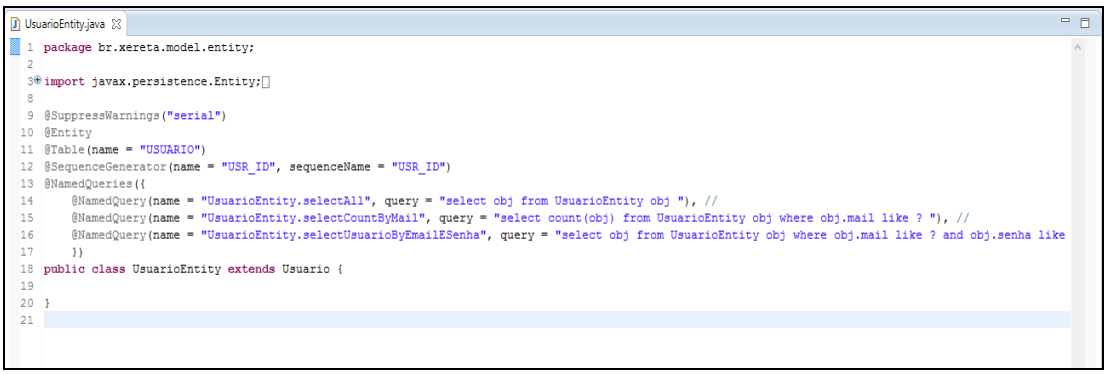
A classe do *managed bean* acessa a classe UsuarioBE que é a classe responsável por manipular a entidade "usuario", como mostra a Figura 13. No método "isEmailUsado" da classe UsuarioBE, verifica-se se o *e-mail* informado já foi cadastrado por outro usuário. Essa validação é utilizada no momento que o usuário está fazendo seu cadastro pessoal na aplicação. Já o método "usuarioByEmailESenha" recupera o objeto "usuario" segundo o *e-mail* e senha informados.

Figura 13 - Classe UsuarioBE



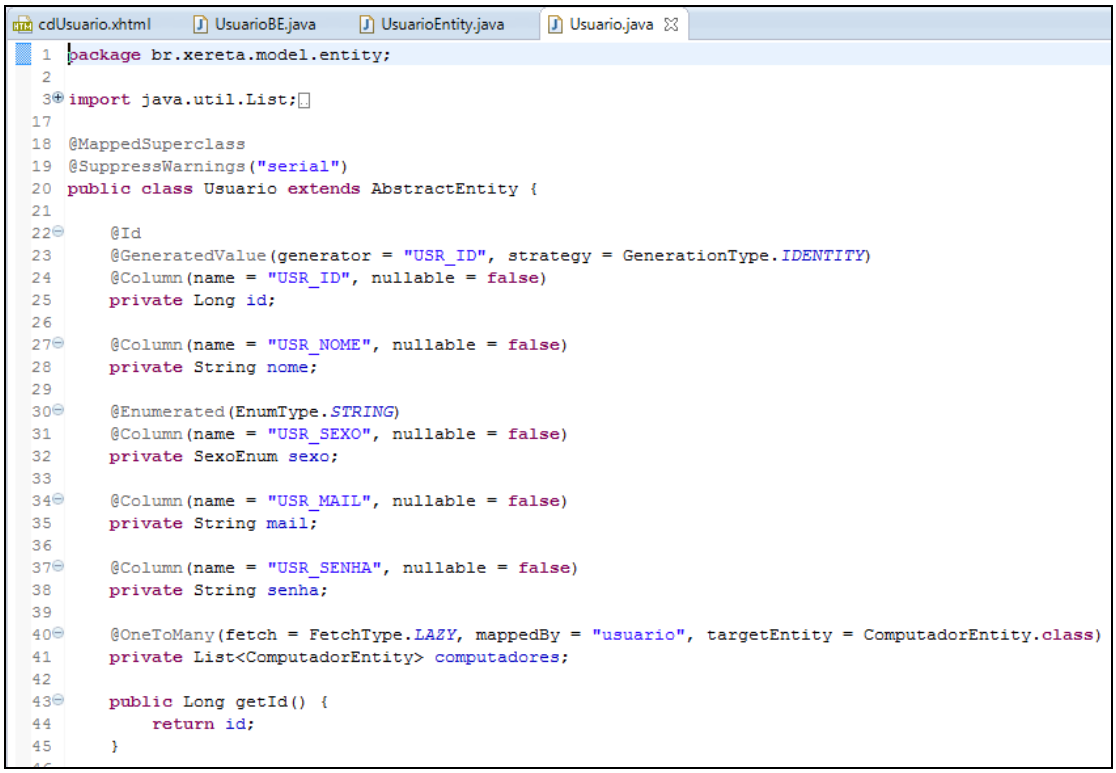
Para obter resultados, a classe UsuarioBE executa *queries* que encontram-se na classe UsuarioEntity, responsável por persistir os campos da entidade "usuario", como apresenta a Figura 14. Nas primeiras linhas da classe UsuarioEntity está a definição da entidade. A partir da linha 13 estão sendo descritas as *queries*, onde a *query* denominada “selectAll”, na linha 14, é responsável por selecionar o objeto "usuario". A *query* “selectCountByMail”, na linha 15, retorna a quantidade de usuários cadastrados com determinado *e-mail* informado e a *query* “selectUsuarioByEmailSenha”, linha 16, valida se o usuário existe através de *e-mail* e senha.

Figura 14 - Classe UsuarioEntity



Na Figura 15 apresenta-se a classe UsuarioEntity que utiliza a classe Usuario que nada mais é do que o mapeamento da tabela Usuario da base de dados.

Figura 15 - Classe Usuario



3.3.1.4 *Framework* Hibernate

Souza (2012) descreve o Hibernate como um *framework* para realizar o mapeamento objeto-relacional escrito na linguagem de programação Java. Seu principal objetivo é diminuir a complexidade envolvida no desenvolvimento de aplicações que precisam trabalhar com banco de dados relacional, onde ele realiza a intermediação entre o banco de dados e sua aplicação, poupando o desenvolvedor da preocupação com instruções SQL para recuperar ou persistir os dados de seu software.

A principal característica deste *framework* é a de transformar as classes Java em tabelas de dados. O Hibernate gera as chamadas SQL e torna o desenvolvedor livre do trabalho manual de converter os dados resultantes, mantendo o programa portável para quaisquer bancos de dados SQL, porém aumentando um pouco o tempo de execução (SILVA, O., 2013).

Souza (2012) ainda cita algumas razões de porque utilizar o Hibernate ao invés de instruções SQL diretamente no banco de dados:

a) possível migrar para a maioria dos SGDBs disponíveis no mercado apenas modificando o arquivo de configuração do *framework*, sem necessidade de alterar

uma linha de código da aplicação;

b) totalmente orientado a objetos;

c) executa as instruções SQL para recuperar apenas as informações necessárias, evitando executar consultas complexas apenas para obter um determinado dado;

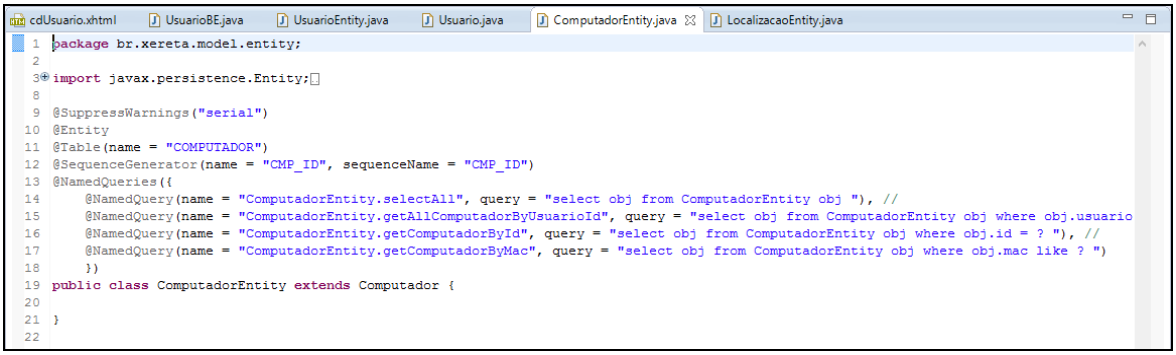
d) detecta automaticamente qualquer alteração ou inclusão das classes mapeadas e executa as alterações devidas no banco de dados;

e) realiza *cache* das operações, aumentando a performance do software;

f) é um software livre.

A Figura 16 mostra a classe ComputadorEntity que possui as *queries* e faz o mapeamento da tabela computador. Primeiramente está sendo definida a entidade da classe, que é "computador". A partir da linha 13 estão sendo criadas as *queries*, sendo que a *query* "selectAll", da linha 14, é responsável por selecionar o objeto "computador"; a *query* "getAllComputadorByUsuarioId" seleciona todos os computadores de determinado usuário; a *query* "getComputadorById" seleciona os computadores de acordo com seu *id* e a *query* "getComputadorByMac" seleciona os computadores de acordo com o seu endereço MAC, que assim como o *id* também é um campo de valor único para cada computador.

Figura 16 - Classe ComputadorEntity



3.3.1.5 Servidor de aplicação Jboss

O JBoss é um servidor de aplicação de código aberto baseado na plataforma *Java Platform, Enterprise Edition* (J2EE), implementada completamente na linguagem de programação Java. JBoss pode ser usado em qualquer Sistema Operacional que suporte Java, pois é baseado nessa linguagem (NEVES, 2008).

Um servidor de aplicações é um software que prepara um ambiente completo para que outras aplicações sejam executadas dentro dele usando uma gama de serviços provida pelo

servidor de aplicações. A maior vantagem de um servidor de aplicações é que os desenvolvedores não precisam se preocupar com aspectos como conexões a bancos de dados, autenticação e gerenciamento de recursos, pois estes são gerenciados pelo servidor de aplicações (4LINUX, 2013).

A arquitetura do Jboss foi projetada para que ele tivesse um bom desempenho. O servidor inicia somente um "contêiner", e depois carrega suas informações através de um arquivo passado pelo servidor por linha de comando. O servidor possui um *microkernel* onde todos os módulos são ligados providenciando assim diferentes funcionalidades do servidor. Através dessa arquitetura, o Jboss permite que o desenvolvedor personalize o ambiente de acordo com suas necessidades reduzindo assim custos com recursos de hardware, por exemplo (SACRAMENTO; MESQUITA, 2011).

3.3.1.6 Banco de Dados MySQL

Segundo Pisa (2012), o MySQL é um Sistema Gerenciador de Banco de Dados (SGBD) relacional de código aberto utilizado em grande parte das aplicações gratuitas para coordenar suas bases de dados. O serviço utiliza a Linguagem de Consulta Estruturada (*Structure Query Language* – SQL), que é a linguagem mais popular para inserir, acessar e gerenciar o conteúdo armazenado em um banco de dados.

O MySQL se tornou o banco de dados de código aberto mais popular do mundo, pois possui consistência, confiabilidade, alta performance e usabilidade. Atualmente, é usado em mais de 6 milhões de instalações em todos os continentes. Estas instalações vão desde instalações em grandes corporações a específicas aplicações embarcadas. Além disso, o MySQL funciona em mais de 20 plataformas, incluindo Linux, Windows, HP-UX, Netware, entre outros (OFICINA DA NET, 2007).

Ainda de acordo com Oficina da Net (2007), o MySQL apresenta algumas características, como:

a) é um banco de dados multiprocessado, ou seja, pode utilizar vários processadores ao mesmo tempo;
b) foi desenvolvido para várias plataformas incluindo ambientes Unix, OS/2 e Windows;
c) permite a seleção de diferentes tabelas de diferentes bases de dados em uma

mesma *query*;

d) suas características de privilégio de *password* são bastante flexíveis, permitindo também a validação por *host*;
e) possui algoritmos de criptografia de *password*, fornecendo assim segurança aos dados gravados nas tabelas;
f) permite a utilização de até 16 índices por tabela;
g) capacidade para manipular bancos com até 50 milhões de registros;
h) permite conexões via TCP/IP;
i) permite acesso via *Open Database Connectivity* (ODBC);
j) possui instruções para extração de informações relativas a tabelas, bancos, índices.

3.3.2 Operacionalidade da implementação

A aplicação *web* foi denominada “Xereta”, pois tem por objetivo permitir que o usuário investigue a localização de seu computador, ou seja, popularmente falando, o usuário pode “xeretar” a localização do seu computador.

Inicialmente é apresentada a tela inicial do protótipo ao usuário que está hospedado apenas em uma máquina local. Se o usuário já estiver cadastrado poderá fazer *login* utilizando a opção “Login”, que encontra-se no menu, à esquerda. Porém, se este é seu primeiro acesso, deverá ser realizado um cadastro através da opção “Cadastre-se” que também está no menu, à esquerda na página.

A Figura 17 mostra a tela inicial do protótipo.

Figura 17 - Tela inicial

Partindo do princípio que este é o primeiro acesso do usuário, a Figura 18 mostra a tela onde devem ser cadastrados os dados necessários para ter acesso ao protótipo. Todos os campos são obrigatórios e possuem esta validação.

Figura 18 - Tela de Cadastro do Usuário

Conforme pode ser visualizado na Figura 19, após concluir o cadastro com sucesso, o usuário poderá fazer *login* no protótipo informando seu *e-mail* e senha previamente

registrados no cadastro de seus dados pessoais.

Figura 19 - Tela de *Login*

Depois de logar-se no protótipo, o usuário é redirecionado para a página principal, como pode ser visualizado na Figura 20. É na página principal que o usuário terá acesso a todas as funcionalidades que a aplicação *web* oferece.

Figura 20 - Tela principal

A Figura 21 mostra a tela onde o usuário pode visualizar os dados que cadastrou e também alterar a senha que utiliza para acessar a aplicação.

Figura 21 - Tela de alteração de dados

.

Na Figura 22 é mostrado o cadastro de computadores, onde o usuário deve cadastrar seu computador para poder visualizar as localizações recebidas.

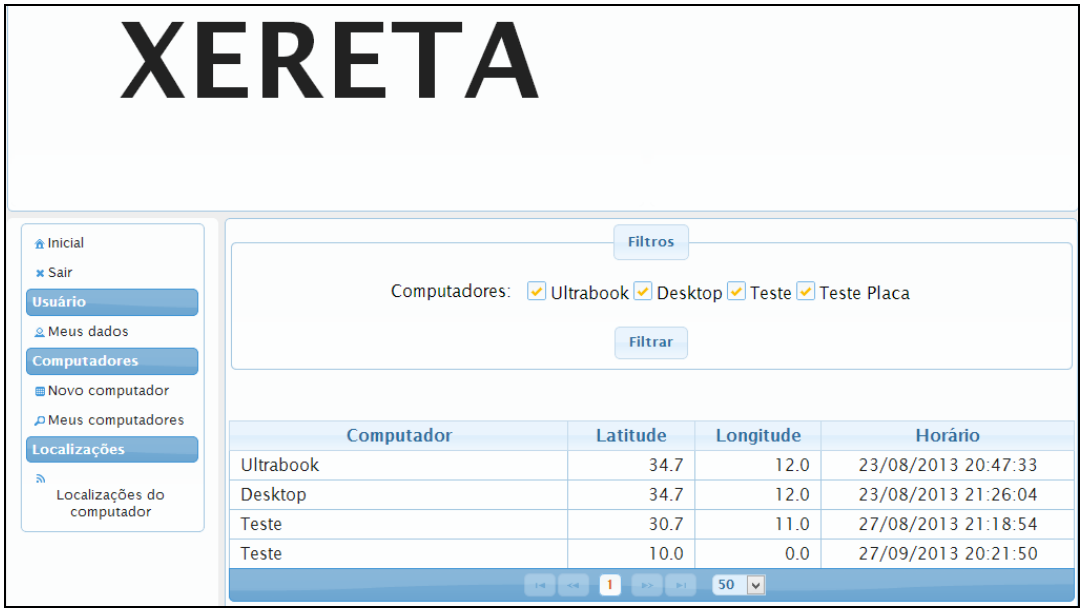
Figura 22 - Tela de Cadastro de Computador

Além de cadastrar computadores, o usuário também pode consultá-los, como mostra a Figura 23. Nessa tela, o usuário também pode desabilitar ou habilitar o envio de notificações de novas localizações que são enviadas para o *e-mail* cadastrado.

Figura 23 - Tela de Consulta de Computadores

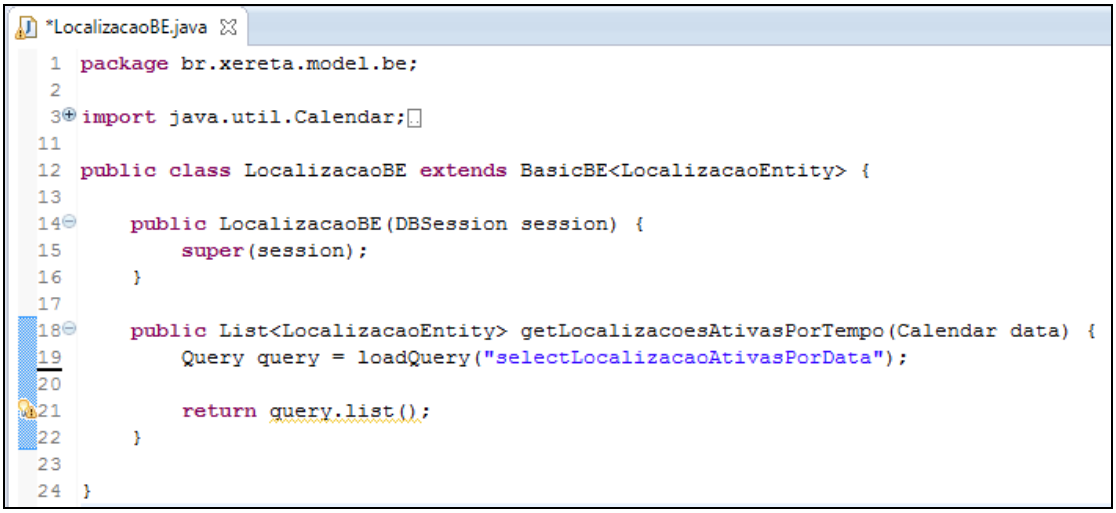
A Figura 24 apresenta a tela de localizações, onde o usuário pode visualizar as localizações recebidas e a data e horário do recebimento das mesmas.

Figura 24 - Telas de Localizações



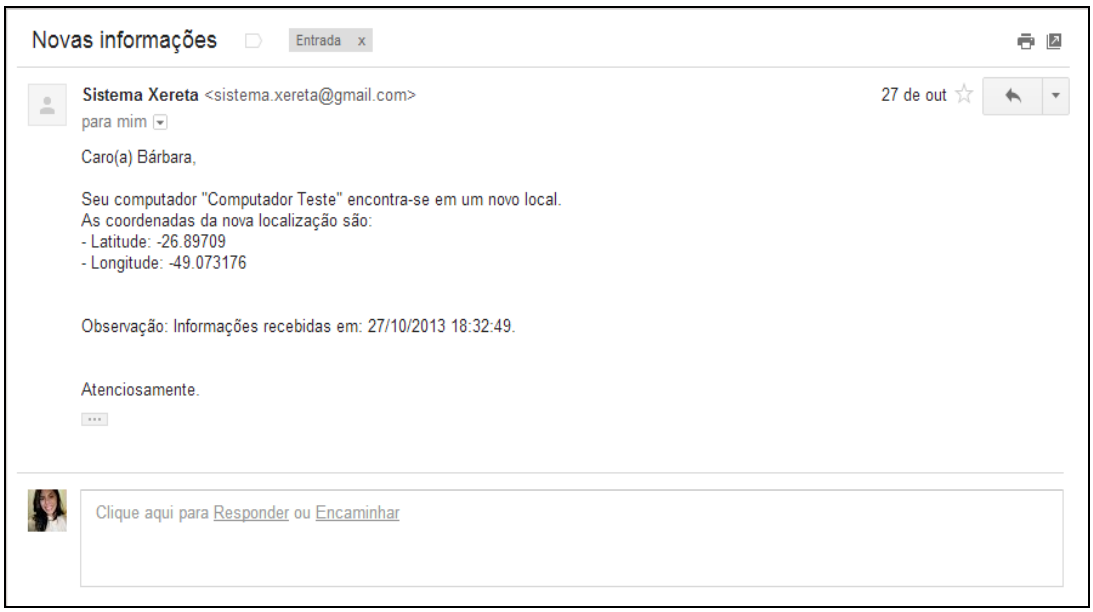
Na Figura 25 é possível visualizar o código executado para buscar as localizações. Onde na linha 19 executa-se uma query que retorna para uma lista as localizações recebidas na data passada de parâmetro e que estão ativas, ou seja, devem ser enviadas por *e-mail*.

Figura 25 - Código para consulta de localizações



Se o usuário ativou a opção de ser notificado por *e-mail*, quando forem registradas novas localizações para determinado computador, o usuário receberá um *e-mail* como o que mostra a Figura 26.

Figura 26 - *E-mail* de notificação de nova localização



Para implementação da rotina de envio de *e-mail* foram utilizadas três classes, são elas: AppServletContextListener, MailCronJob e MailUtil.

A Figura 27 exibe a classe AppServletContextListener que funciona como um *listener*, ou seja, ela fica escutando o servidor. Em seu método principal, que inicia na linha 25, primeiramente ela instancia um *scheduler* (agendador), que é responsável por agendar as

atividades. Na linha 29, a classe instancia uma tarefa identificando-a como uma tarefa de *e-mail* da classe MailCronJob. Na sequência, é criada uma *trigger* e logo após é definido que ela será disparada a cada trinta minutos e será executada enquanto o servidor estiver ativo para o envio de *e-mails* – a partir da linha 31.

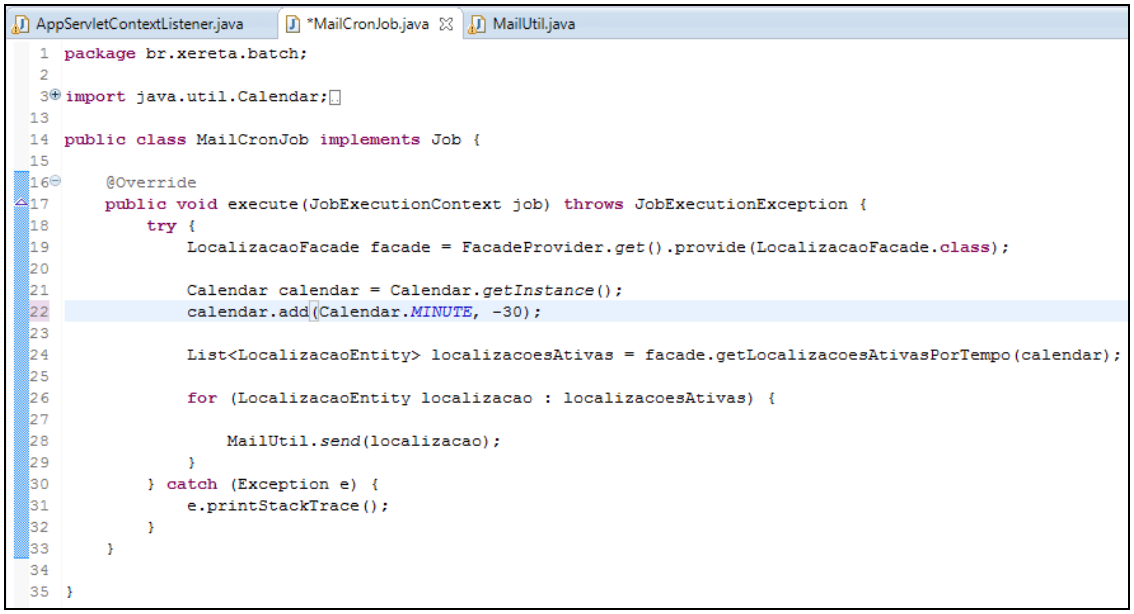
Figura 27 - Classe AppServletContextListener

AppServletContextListener.java | MailCronJob.java | MailUtil.java

```
1  package br.xereta.batch;
2
3  import static org.quartz.JobBuilder.newJob;
18 public class AppServletContextListener implements ServletContextListener {
19
20     @Override
21     public void contextDestroyed(ServletContextEvent arg0) {
22     }
23
24     @Override
25     public void contextInitialized(ServletContextEvent arg0) {
26         SchedulerFactory sf = new StdSchedulerFactory();
27         try {
28             Scheduler sche = sf.getScheduler();
29             JobDetail job = newJob(MailCronJob.class).withIdentity("MailJob", "Mail").build();
30
31             TriggerBuilder<Trigger> tbuilder = newTrigger().withIdentity("MailTrigger", "Mail");
32             tbuilder.startNow();
33             tbuilder.withSchedule(simpleSchedule().withIntervalInMinutes(30).repeatForever());
34
35             Trigger trigger = tbuilder.build();
36
37             sche.scheduleJob(job, trigger);
38             sche.start();
39         } catch (SchedulerException e) {
40             e.printStackTrace();
41             throw new RuntimeException(e);
42         }
43     }
44
45 }
```

A classe MailCronJob, mostrada na Figura 28, é responsável por enviar os *e-mails* referentes as novas localizações. Inicialmente ela busca o objeto localizacao – linha 19, em seguida utiliza a classe *Calendar* e passa para a mesma a data e hora atual decrementando trinta minutos. O próximo passo é buscar uma lista de localizações ativas e que tenham sido recebidas nos últimos trinta minutos. Por fim, percorre-se a lista de localizações enviando os *e-mails* – a partir da linha 26.

Figura 28 - Classe MailCronJob

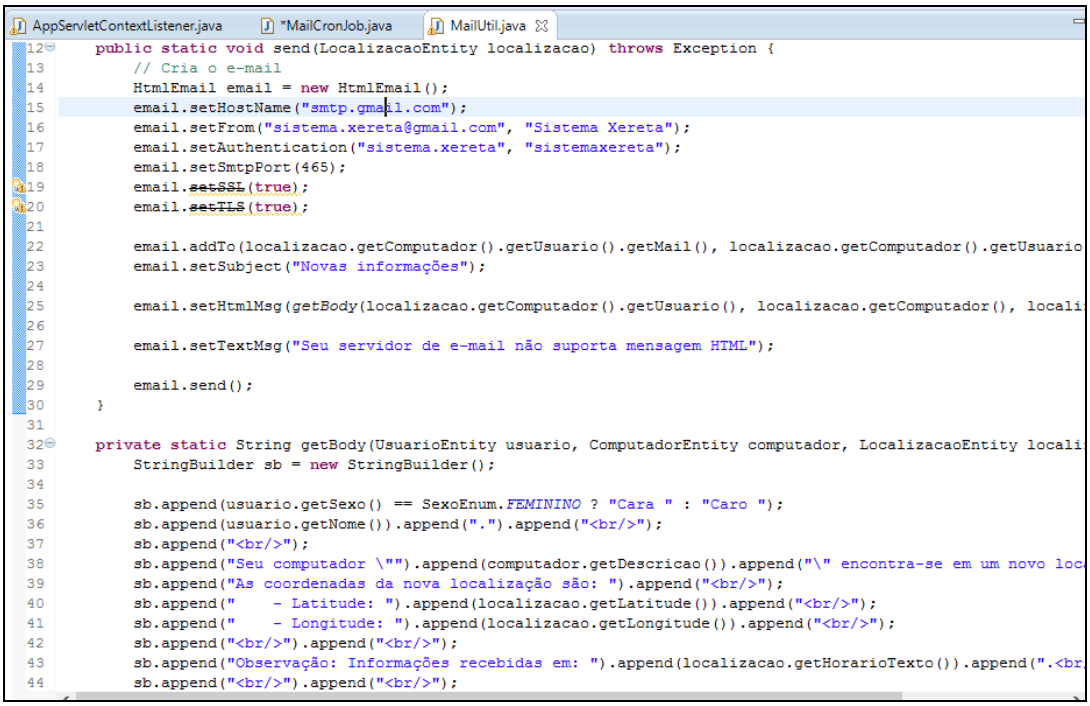


```
AppServletContextListener.java | *MailCronJob.java | MailUtil.java

1  package br.xereta.batch;
2
3  import java.util.Calendar;
13
14 public class MailCronJob implements Job {
15
16     @Override
17     public void execute(JobExecutionContext job) throws JobExecutionException {
18         try {
19             LocalizacaoFacade facade = FacadeProvider.get().provide(LocalizacaoFacade.class);
20
21             Calendar calendar = Calendar.getInstance();
22             calendar.add(Calendar.MINUTE, -30);
23
24             List<LocalizacaoEntity> localizacoesAtivas = facade.getLocalizacoesAtivasPorTempo(calendar);
25
26             for (LocalizacaoEntity localizacao : localizacoesAtivas) {
27
28                 MailUtil.send(localizacao);
29             }
30         } catch (Exception e) {
31             e.printStackTrace();
32         }
33     }
34
35 }
```

A Figura 29 apresenta a classe MailUtil, responsável pela mensagem de *e-mail*. O método *getBody* monta a mensagem que será enviada – a partir da linha 32, enquanto o método *send* é responsável por configurar a conta de *e-mail* de onde são remetidas as mensagens, buscar os dados do endereço que deve receber a mensagem e então enviar a mensagem para a conta de *e-mail* do usuário.

Figura 29 - Classe MailUtil



```
AppServletContextListener.java | *MailCronJob.java | MailUtil.java

12     public static void send(LocalizacaoEntity localizacao) throws Exception {
13         // Cria o e-mail
14         HtmlEmail email = new HtmlEmail();
15         email.setHostName("smtp.gmail.com");
16         email.setFrom("sistema.xereta@gmail.com", "Sistema Xereta");
17         email.setAuthentication("sistema.xereta", "sistemaxereta");
18         email.setSmtpPort(465);
19         email.setSSL(true);
20         email.setTLS(true);
21
22         email.addTo(localizacao.getComputador().getUsuario().getMail(), localizacao.getComputador().getUsuario
23         email.setSubject("Novas informações");
24
25         email.setHtmlMsg(getBody(localizacao.getComputador().getUsuario(), localizacao.getComputador(), locali
26
27         email.setTextMsg("Seu servidor de e-mail não suporta mensagem HTML");
28
29         email.send();
30     }
31
32     private static String getBody(UsuarioEntity usuario, ComputadorEntity computador, LocalizacaoEntity locali
33         StringBuilder sb = new StringBuilder();
34
35         sb.append(usuario.getSexo() == SexoEnum.FEMININO ? "Cara " : "Caro ");
36         sb.append(usuario.getNome()).append(".").append("<br/>");
37         sb.append("<br/>");
38         sb.append("Seu computador \"").append(computador.getDescricao()).append("\" encontra-se em um novo loc
39         sb.append("As coordenadas da nova localização são: ").append("<br/>");
40         sb.append("    - Latitude: ").append(localizacao.getLatitude()).append("<br/>");
41         sb.append("    - Longitude: ").append(localizacao.getLongitude()).append("<br/>");
42         sb.append("<br/>").append("<br/>");
43         sb.append("Observação: Informações recebidas em: ").append(localizacao.getHorarioTexto()).append(".<br
44         sb.append("<br/>").append("<br/>");
```

3.4. RESULTADOS E DISCUSSÃO

Este trabalho teve por finalidade desenvolver um protótipo de um sistema de rastreamento para computadores que visa proporcionar mais segurança aos proprietários de computadores, bem como auxiliá-los na recuperação destes em caso de perda e/ou roubo. No que diz respeito ao funcionamento do protótipo, de maneira geral, todos os objetivos foram alcançados. Contudo, por esse rastreamento ser direcionado a computadores, o intuito inicial era o de colocar o rastreador dentro de um computador *desktop* e alimentá-lo com a energia da fonte do próprio computador. Porém, o módulo GPS utilizado está limitado ao recebimento de sinal apenas enquanto encontrar-se a céu aberto, ou seja, não é possível colocar o rastreador dentro de um computador.

Quanto aos trabalhos correlatos, constatam-se semelhanças com o protótipo de Beszczynski (2008), porém este desenvolveu um protótipo de um sistema de rastreamento para veículos. Em relação ao software Prey Project Prass (2011), desenvolvido pela empresa Fork Ltd., tanto este quanto o protótipo desenvolvido neste trabalho tem por finalidade o rastreamento de computadores roubados ou perdidos. Os mesmos necessitam de um pré-cadastro do usuário e seu computador e disponibilizam as localizações encontradas em uma página *web*. O que difere o protótipo do Prey Project, é que o Prey Project é apenas um software instalado no computador, enquanto o protótipo em questão não necessita de nenhum aplicativo instalado para funcionar e conta com a fundamental contribuição de um rastreador.

4 CONCLUSÕES

Segundo Silva, E. (2013), o medo da violência tem levado as pessoas a investirem mais em segurança privada. As empresas têm registrado elevação na procura por esse tipo de serviço. Câmeras de vigilância, alarme, cerca elétrica e até mesmo vigilantes noturnos se tornaram itens de primeira necessidade para garantir a segurança da população. Observando esta questão e analisando que no que diz respeito a computador, as opções para segurança de dados presentes no mercado são inúmeras, mas para integridade física dos mesmos não estão disponíveis tantas opções. Propôs-se neste trabalho o desenvolvimento de um protótipo de sistema de rastreamento para computadores com o propósito de fornecer mais segurança física a estes equipamentos e auxiliar na recuperação dos mesmos em caso de perda e/ou roubo.

Os objetivos específicos propostos neste trabalho foram alcançados. Mostrou-se que é possível fazer o rastreamento de computadores com as tecnologias, hardwares e ferramentas utilizadas, bem como registrar e disponibilizar as informações relevantes. No rastreador, que na verdade é uma placa com Linux embarcado, com módulo GPS integrado e um modem 3G plugado, é executado um *script* desenvolvido na linguagem de programação Python, que gerencia desde a detecção de coordenadas válidas pelo GPS até o envio destas para o servidor da aplicação *web*. Em relação a interface de interação com o usuário, que é a aplicação *web*, foram desenvolvidas telas que além de serem muito simples e intuitivas, cumprem perfeitamente o objetivo de mostrar ao usuário as localizações recebidas.

Para o desenvolvimento da aplicação embarcada fez-se uso de hardware e software. O hardware é uma placa Terra G25 a qual foi integrado um módulo GPS e conectado um modem 3G. Dentro da placa, é executado um *script* que foi desenvolvido na linguagem de programação Python. Escolheu-se a placa Terra G25 por esta possuir a estrutura necessária para execução deste trabalho, bem como a linguagem de programação Python por ser clara, de fácil entendimento e produzir código curto e legível devido as automações disponíveis em suas bibliotecas. A aplicação *web* foi desenvolvida utilizando os *frameworks* JSF2 e Hibernate juntamente com o servidor de aplicação JBoss. Para armazenamento das informações, utilizou-se o banco de dados MySQL. Todas as ferramentas utilizadas atenderam as necessidades e auxiliaram satisfatoriamente na construção deste protótipo.

Ao término deste trabalho concluiu-se que tecnologia associada a ideias coerentes e inovadoras e a ferramentas e equipamentos corretos podem resultar em grandes projetos que proporcionam uma série de vantagens para a vida das pessoas. Com a construção deste, foram

encontrados muitos desafios, mas também foram adquiridos muitos conhecimentos, principalmente em ferramentas, linguagens, conceitos e técnicas que eram parcialmente desconhecidos pela autora. Além disso, também criou-se uma nova ideia de produto de rastreamento, que pode ser aprimorada e futuramente colocada a disposição dos usuários.

4.1. EXTENSÕES

Para continuidade do presente trabalho, sugere-se:

a) substituir o módulo GPS do rastreador por um módulo que seja capaz de receber sinal tanto em lugares abertos, como em lugares fechados. Para que esse GPS receba sinal, é necessário que o mesmo tenha uma antena discreta que fique em contato com o ambiente externo, isto é, fora do computador em questão para evitar problemas de mau contato com os componentes eletrônicos do computador e facilitar a captura de coordenadas;
b) automatizar a detecção do endereço MAC do computador em questão, ou seja, fazer com que o rastreador comunique-se com o computador ao qual ele pertence e descubra o endereço MAC da placa de rede deste computador. Esta funcionalidade deve ser implementada no rastreador desenvolvido;
c) permitir que o usuário informe pontos neutros referentes a localizações que o servidor não deve armazenar, pois tais lugares são de estadia frequente do usuário, como, por exemplo, sua própria casa e trabalho;
d) reduzir o tamanho do rastreador para que o mesmo possa ser colocado também em computadores portáteis, como *laptops*, *notebooks*, *ultrabooks* e *netbooks*.

REFERÊNCIAS

4LINUX, FREE SOFTWARE SOLUTIONS. **O que é Jboss?**. São Paulo, [2013?]. Disponível em: <http://www.4linux.com.br/o-que-e-jboss.html>. Acesso em: 23 set. 2013.

ACME Systems. **Terra - Linux embedded single board computer designed with Aria G25 SoM**. Ladispoli, Itália, [2013?]. Disponível em: <http://www.acmesystems.it/brochures/TerraBoardDescription.pdf>. Acesso em: 1 nov. 2013.

ALVES, Rudson. **Python 1 – Uma introdução ao Python**. [S.l], 2009. Disponível em: <http://rra.etc.br/MyWorks/2009/09/07/python-1-uma-introducao-ao-python/#comments>. Acesso em: 17 set. 2013.

BESZCZYNSKI, Leandro. **Protótipo de um sistema de rastreamento veicular baseado no Módulo Telit**. 2008. Monografia (Curso Bacharel em Ciência da Computação) – Departamento de Sistemas de Computação da Universidade Regional de Blumenau, Blumenau, 2011.

CÂMARA, Marlon. **O que é internet 3G?**. [S.l], 2012. Disponível em: <http://www.techtudo.com.br/artigos/noticia/2012/01/o-que-e-internet-3g.html>. Acesso em: 18 set. 2013.

COMPUTERWORLD. **Quase metade dos brasileiros diz não ter controle sobre dados na redes**. [S.l], 2012. Disponível em: <http://computerworld.uol.com.br/seguranca/2012/08/16/quase-metade-dos-brasileiros-diz-nao-ter-controle-sobre-dados-na-redes/>. Acesso em: 6 jun. 2013.

CONVERGÊNCIA DIGITAL. **Brasil:** jovens são 'loucos' por Tecnologia. [S.l], 2013. Disponível em: <http://convergenciadigital.uol.com.br/cgi/cgilua.exe/sys/start.htm?infoid=33973&sid=46#.UbJBPqBID2i>. Acesso em: 3 jun. 2013.

CORRÊA, William. **Linux + Segurança:** segurança – o ponto mais forte do Linux. [S.l], 2004. Disponível em: <http://imasters.com.br/artigo/2079/seguranca/seguranca-o-ponto-mais-forte-do-linux/>. Acesso em: 14 abr. 2013.

DECICINO, Ronaldo. **GPS:** Sistema de Posicionamento Global tem diferentes utilidades. [S.l], 2009. Disponível em: <http://educacao.uol.com.br/disciplinas/geografia/gps-sistema-de-posicionamento-global-tem-diferentes-utilidades.htm>. Acesso em: 15 abr. 2013.

DILÃO, Rui. **GPS**. [Lisboa, Portugal], [2013?]. Disponível em: <http://www.cienciaviva.pt/latlong/anterior/gps.asp>. Acesso em: 15 abr. 2013.

DRUMOND, Douglas. **A História do Python**. [S.l], 2009. Disponível em: <http://python-history-pt-br.blogspot.com.br/>. Acesso em: 6 jun. 2013.

EMBEDDED ARCHITECTS. **O que é um sistema embarcado**. Brasília, [2013?]. Disponível em: <http://www.embarc.com.br/p1600.aspx>. Acesso em: 11 set. 2013.

FOROUZAN, Behrouz A. **Comunicação de Dados e Redes de Computadores**. 3. ed. Tradução Glayson Eduardo de Figueiredo, Pollyanna Miranda de Abreu. Porto Alegre: Bookman, 2004.

FOROUZAN, Behrouz A. **Protocolo TCP/IP**. 3. ed. Tradução João Eduardo Nóbrega Tortello. Porto Alegre: AMGH, 2010.

GERMANO, Alan. **Você sabe o que são Sistemas Embarcados?**. Campos do Jordão, 2013. Disponível em: <http://www.gruponetcampos.com.br/2011/05/voce-sabe-o-que-sao-sistemas-embarcados/>. Acesso em: 13 set. 2013.

GODOY, Fernando. **O que é JSF?**. [S.l], 2011. Disponível em: <http://fernandogodoy.wordpress.com/category/javaserver-faces/>. Acesso em: 23 set. 2013.

LUTZ, Mark; ASCHER, David. **Aprendendo Python**. 2. ed. Porto Alegre: Bookman, 2007.

MATOS, Francisco Jarbas Teixeira. **Entendendo os recursos do Linux**. São Paulo: Digerati Books, 2007.

MORIMOTO, Carlos E. **3G: UMTS, WCDMA, HSDPA e a questão das freqüências**. [S.l], 2011. Disponível em: <http://www.hardware.com.br/dicas/umts.html>. Acesso em: 10 nov. 2013.

NEGUS, Christopher. **Linux**: a Bíblia. Tradução Daniela Botelho. Rio de Janeiro: Alta Books, 2007.

NEVES, Bruno. **O que é Jboss?**. [S.l], 2008. Disponível em: <http://www.dimensaotech.com/2008/03/o-que-e-jboss/>. Acesso em: 23 set. 2013.

OFICINA DA NET. **Por que usar MySQL?**. [S.l], 2007. Disponível em: <http://www.oficinadanet.com.br/artigo/484/por_que_usar_mysql>. Acesso em: 23 set. 2013.

OLHAR DIGITAL. **Conheça as diferenças entre 1G, 2G, 3G e 4G**. [S.l], 2013. Disponível em: <http://olhardigital.uol.com.br/noticia/conheca-as-diferencas-entre-1g,-2g,-3g-e-4g/34225>. Acesso em: 22 set. 2013.

OLIVEIRA, Wilson José de. **Segurança da Informação**: técnicas e soluções. Florianópolis: Visual Books, 2001.

PAGANINI, Silvio. JSF 2.0. **Java Magazine**, Rio de Janeiro, n. 78, abr. 2010. Disponível em: < http://www.devmedia.com.br/artigo-java-magazine-78-jsf-2-0/16559>. Acesso em: 23 set. 2013.

PEREIRA, Ana Paula. **A história do Linux**. [S.l], 2010. Disponível em: <http://www.tecmundo.com.br/sistema-operacional/4228-a-historia-do-linux.htm>. Acesso em: 14 abr. 2013.

PÉRICAS, Francisco Adell. **Redes de Computadores**: conceitos e a arquitetura Internet. 3 ed. Blumenau: Ed. do Autor, 2012.

PISA, Pedro. **O que é e como usar o MySQL?**. [S.l], 2012. Disponível em: <http://www.techtudo.com.br/artigos/noticia/2012/04/o-que-e-e-como-usar-o-mysql.html>. Acesso em: 23 set. 2013.

PORTOGENTE. **GPS - Global Positioning System**. In: PORTOPÉDIA, a enciclopédia livre que todos podem editar. São Paulo, [2013?]. Disponível em: <http://portogente.com.br/portopedia/gps-global-positioning-system-73983/73983>. Acesso em: 04 dez. 2013.

PRASS, Ronaldo. **Conheça programa para rastrear computador perdido ou roubado**. [S.l], 2011. Disponível em: <http://g1.globo.com/tecnologia/noticia/2011/06/conheca-programa-para-rastrear-computador-perdido-ou-roubado.html>. Acesso em: 4 jun. 2013.

REZENDE, Sidney. **Governo brasileiro publica diretrizes para segurança da informação**. [S.l], 2013. Disponível em: <http://www.sidneyrezende.com/noticia/208027+governo+brasileiro+publica+diretrizes+para+seguranca+da+informacao>. Acesso em: 3 jun. 2013.

SACRAMENTO, Ivana; MESQUITA, Tâmara. **Jboss Application Server**. [S.l], 2011. Disponível em: <http://pesquompile.wikidot.com/jboss-application-server>. Acesso em: 23 set. 2013.

SCRIMGER, Rob et al. **TCP/IP**: a Bíblia. Tradução Edson Furmankiewicz. Rio de Janeiro: Elsevier, 2002.

SILVA, Ednéia. **Cresce a procura por segurança privada**. Rio Claro, 2013. Disponível em: <http://www.jornalcidade.net/rioclaro/seguranca/seguranca/107525--Cresce-a-procura-por-seguranca-privada->. Acesso em: 10 nov. 2013.

SILVA, Osmar J. **Hibernate:** O que é, como baixar, instalar, configurar e testar. [S.l], [2013?]. Disponível em: <http://www.arquivodecodigos.net/principal/diretorios/hibernate/artigos_tutoriais/hibernate_o_que_e_como_baixar_instalar_configurar_e_testar.php>. Acesso em: 23 set. 2013.

SOUZA, Naison. **O que é hibernate?**. [S.l], 2012. Disponível em: <http://blog.naison.com.br/java/o-que-e-hibernate>. Acesso em: 23 set. 2013.

SPORTACK, Mark A. **TCP/IP, primeiros passos**. Tradução Flávio Morgado. Rio de Janeiro: Ciência Moderna, 2007.

STUMM JUNIOR., Valdir. **Acessando recursos na web com Python**. [S.l], 2013. Disponível em: <http://pythonhelp.wordpress.com/2013/03/12/acessando-recursos-na-web-com-python/>. Acesso em: 31 out. 2013.

TAURION, Cezar. **Software embarcado**: oportunidades e potencial de mercado. Rio de Janeiro: Brasport, 2005.

TURRM, Renan. **Dicas de como utilizar a internet pelo celular durante sua viagem**. São Paulo, 2011. Disponível em: <http://www.guanabara.info/2011/12/dicas-de-como-utilizar-a-internet-pelo-celular-durante-sua-viagem/>. Acesso em: 04 dez. 2013.

VIEIRA, Nando. **Entendendo um pouco mais sobre o protocolo HTTP**. [S.l], 2007. Disponível em: <http://simplesideias.com.br/entendendo-um-pouco-mais-sobre-o-protocolo-http>. Acesso em: 22 set. 2013.

APÊNDICE A – Descrição dos Casos de Uso

Este Apêndice apresenta a descrição dos principais casos de uso descritos na seção de especificação deste trabalho. No Quadro 6 tem-se a descrição do caso de uso Consultar lugares em que o computador esteve, da aplicação *web*.

Quadro 6 - Descrição do UC02.01 Consultar lugares em que o computador esteve

UC02.01 – Consultar lugares em que o computador esteve
Permite que o usuário visualize as últimas localizações em que o computador esteve através da página *web*.

Ator: Usuário
Pré-condição: O usuário deve estar cadastrado.
Pré-condição: O usuário deve estar logado na aplicação.
Pós-condição: Usuário visualiza todas as localizações que estão registradas para determinado computador.

Cenário Principal

1. O usuário informa *e-mail* e senha;
2. O sistema valida os dados informados;
3. O sistema direciona o usuário para a página inicial;
4. O usuário seleciona no menu a opção para visualizar as localizações do computador cadastrado;
5. O sistema exibe todas as localizações registradas até o momento.

Fluxo Alternativo 1

1. O usuário informa o *e-mail* e/ou senha inválidos;
2. Alerta com mensagem “Dados Inválidos” é mostrado.

No Quadro 7 visualiza-se o detalhamento do caso de uso Ativar/Desativar notificações.

Quadro 7 - Descrição do UC03.01 Ativar/desativar notificações

UC03.01 – Ativar/desativar notificações
Permite que o usuário ative e/ou desative notificações enviadas ao *e-mail* do mesmo previamente cadastrado. Estas notificações são em relação as localizações do computador, ou seja, a cada nova localização identificada, a mesma é enviada por *e-mail* ao usuário.

Ator: Usuário
Pré-condição: O usuário deve estar cadastrado.

Pré-condição: O usuário deve estar logado na aplicação.
Pré-condição: O usuário deve possuir pelo menos um computador cadastrado.
Pós-condição: Usuário recebe ou deixa de receber notificações em seu *e-mail*.

Cenário Principal

1. O usuário escolhe o computador que deseja ativar/desativar as notificações;
2. O usuário clica no *checkbox* e ativa/desativa as notificações.

Fluxo Alternativo 1

1. O usuário não possui computadores para realizar esta operação.

No Quadro 8 apresenta-se o detalhamento do caso de uso Enviar notificações por *e-mail*.

Quadro 8 - Descrição do UC04.01 Enviar notificações por *e-mail*

UC04.01 – Enviar notificações por *e-mail*
Permite que o servidor da aplicação *web* envie notificações sobre a localização do computador por *e-mail*.

Ator: Servidor
Pré-condição: O servidor deve estar com seus serviços iniciados, bem como o mesmo deve estar acessível;
Pré-condição: O usuário deve ter um *e-mail* cadastrado;
Pré-condição: O usuário deve estar com a opção "Enviar notificações" ativa;
Pós-condição: Servidor envia notificações para o *e-mail* do usuário cadastrado.

Cenário Principal

1. O usuário ativa a opção "Enviar notificações";
2. O servidor recebe a informação de que deve enviar as próximas localizações do computador para o *e-mail* do usuário;
3. A cada nova localização enviada pelo rastreador e recebida pelo servidor, o mesmo envia para o *e-mail* do usuário cadastrado.

Fluxo Alternativo 1

1. O usuário não possui computadores para realizar esta operação.

No Quadro 9 demonstra-se o detalhamento do caso de uso Enviar as coordenadas para o servidor.

Quadro 9 - Descrição do UC02.02 Enviar as coordenadas para o servidor

UC02.02 – Enviar as coordenadas para o servidor.
Permite que o rastreador através do GPS integrado no mesmo capture a localização do computador e envie a mesma para o servidor através de um *script* de gerenciamento da aplicação embarcada.

Ator: Software embarcado
Pré-condição: O rastreador deve estar conectado a rede 3G;
Pré-condição: O rastreador deve possuir coordenadas válidas para enviar.
Pós-condição: O servidor recebe as coordenadas enviadas pelo rastreador.

Cenário Principal

1. O software embarcado verifica se as coordenadas que o GPS está retornando são válidas;
2. O software embarcado envia as coordenadas encontradas para o servidor da aplicação *web*.

Fluxo Alternativo 1

1. O GPS não consegue capturar coordenadas e, portanto, o software embarcado não envia nenhuma informação para o servidor.

No Quadro 10 apresenta-se o detalhamento do caso de uso Armazenar as coordenadas recebidas.

Quadro 10 - Descrição do UC06.01 Armazenar as coordenadas recebidas

UC06.01 – Armazenar as coordenadas recebidas.
Permite que servidor armazene as coordenadas recebidas de acordo com as validações necessárias.

Ator: Servidor
Pré-condição: O servidor deve estar com seus serviços iniciados, bem como o mesmo deve estar acessível;
Pré-condição: O servidor deve ter recebido pelo menos uma requisição com as coordenadas a serem registradas;
Pós-condição: O servidor registra as coordenadas em seu banco de dados.

Cenário Principal

1. O servidor recebe a requisição com as coordenadas;
2. A aplicação *web* valida se as coordenadas recebidas estão fora da rota permitida informada anteriormente pelo usuário, no cadastro do computador;

Fluxo Alternativo 1

1. O servidor não recebeu nenhuma requisição.

Fluxo Alternativo 2

1. No passo 2 do fluxo principal, a aplicação *web* verifica que as coordenadas recebidas estão dentro da rota permitida pelo usuário e, portanto, esta localização não é registrada.

APÊNDICE B – Descrição do Dicionário de Dados

Este Apêndice apresenta a descrição das tabelas do banco de dados mencionadas na seção de especificação deste trabalho. Os tipos de dados utilizados para os campos são:

a) *bigint*: armazena valores inteiros. Usada quando existe a possibilidade de os valores inteiros excederem a faixa suportada pelo tipo de dados;
b) *varchar*: armazena caracteres alfanuméricos;
c) *tinyint*: armazena valores inteiros que se encaixem na faixa de -128 a 127;
d) *datetime*: armazena valores de data e hora;
e) *double*: armazena valores decimais de até 8 bytes.

Os Quadros de 11 a 13 apresentam o dicionário de dados das tabelas Usuario, Computador e Localizacao, respectivamente.

Quadro 11 – Tabela Usuário

Usuário – Entidade responsável por armazenar os usuários do protótipo.				
Campo	**Descrição**	**Tipo**	**Tamanho**	**Chave Primária**
usr_id	Código do usuário	Bigint	20	Sim
usr_mail	E-mail do usuário	Varchar	255	Não
usr_nome	Nome do usuário	Varchar	255	Não
usr_senha	Senha do usuário	Varchar	255	Não
usr_sexo	Sexo do usuário	Varchar	255	Não

Quadro 12 – Tabela Computador

Computador – Entidade responsável por armazenar os computadores cadastrados através da aplicação *web*.				
Campo	**Descrição**	**Tipo**	**Tamanho**	**Chave Primária**
cmp_id	Código do computador	Bigint	20	Sim
cmp_descricao	Descrição do computador	Varchar	255	Não
cmp_mac	Endereço MAC do computador	Varchar	255	Não
cmp_notificar_mail	Notifica usuário	Tinyint	1	Não

Quadro 13 – Tabela Localizacao

Localizacao – Entidade responsável por armazenar as localizações de determinado computador enviadas pelo rastreador.				
Campo	**Descrição**	**Tipo**	**Tamanho**	**Chave Primária**
loc_id	Código da localização	Bigint	20	Sim
loc_horario	Data e hora da localização	DateTime	-	Não
loc_latitude	Latitude da localização	Double	-	Não
loc_longitude	Longitude da localização	Double	-	Não