UNIVERSIDADE FEDERAL DO CEARÁ
DEPARTAMENTO DE ENGENHARIA DE TELEINFORMÁTICA
GRADUAÇÃO EM ENGENHARIA DE TELEINFORMÁTICA

SISTEMA DE MONITORAMENTO COM CÂMERAS DE VÍDEO DE BAIXO CUSTO EM TEMPO REAL

FORTALEZA - CEARÁ
2011

MISSIAS UCHOA CAVALCANTE

SISTEMA DE MONITORAMENTO COM CÂMERAS DE VÍDEO DE BAIXO CUSTO EM TEMPO REAL

Monografia de Conclusão de Curso apresentada à Coordenação do Curso de Graduação em Engenharia de Teleinformática da Universidade Federal do Ceará como parte dos requisitos para a obtenção do grau de Engenheiro de Teleinformática.

Orientador: Professor José Marques Soares

FORTALEZA - CEARÁ

2011

RESUMO

Diante do cenário atual, em que a violência vem aumentando a cada dia no mundo, um tipo de sistema de vigilância vem ganhando espaço no campo da segurança: são os sistemas de monitoramento com câmeras de vídeo. Porém algumas versões desses sistemas ainda são pouco acessíveis à maioria das pessoas. Neste trabalho propomos um sistema de monitoramento com o uso de câmeras de vídeo de baixo custo, as câmeras *web* (ou *webcams*). Estes dispositivos podem ser encontrados a preços bem acessíveis e estão freqüentemente disponíveis nos computadores usados em ambientes doméstico e profissional. A disponibilidade, o baixo custo do equipamento e o acesso à internet banda larga, cada vez mais presente nos lares e empresas, garante o baixo custo da solução proposta. Acessível à maioria das pessoas, este sistema oferece recursos de visualização em tempo real das suas imagens em qualquer local do mundo utilizando navegadores *Web*.

Palavras-chaves: Sistema de Vigilância, Monitoramento com Câmeras de Vídeo, Câmeras *Web*, Tempo Real, *Internet*.

ABSTRACT

Front the current scenario, where violence is increasing every day around the world, a type of surveillance system has been gaining ground in the field of safety: they are monitoring systems with video cameras. But some versions of these systems are still inaccessible to most people. In this paper we propose a monitoring system using video cameras at low cost, web cameras (or webcams). These devices can be found at very affordable prices and are often available on the computers used in home and professional environments. The availability, the low cost of equipment and access to broadband Internet, increasingly present in homes and businesses, ensures the low cost of the proposed solution. Accessible to most people, this system offers real-time viewing features of your images anywhere in the world using web browsers.

Keywords: Surveillance System, Monitoring with Video Cameras, Webcams, Real Time, Internet.

Dedico este trabalho aos meus pais, Raimundo e Helena,
Por sempre incentivarem e apoiarem meus estudos.

AGRADECIMENTOS

Primeiramente a Deus, por estar comigo em todos os momentos de minha vida,

À minha família, por estar sempre unida e pronta para me ajudar quando preciso,

Aos colegas do curso de graduação em Engenharia de Teleinformática, pelo companheirismo e amizade,

Ao professor José Marques Soares, pelos ensinamentos e por sempre estar disponível para ajudar durante a orientação acadêmica,

Ao professor Alexandre de Moreira Moraes, pelos ensinamentos e apoio de grande valia para minha formação profissional,

Aos demais professores do Departamento de Engenharia de Teleinformática que fizeram parte de minha graduação, pelos conhecimentos repassados,

À Universidade Federal do Ceará, pela formação de nível superior,

Ao Colégio Espaço Aberto, pela bolsa integral de ensino pré-vestibular que me ajudou bastante a estar aqui hoje escrevendo esta monografia.

O que queres que os homens façam por ti, faça igualmente por eles.

Jesus Cristo

Sumário

LISTA DE FIGURAS

LISTA DE SIGLAS

AAC - *Advanced Audio Coding* (Codificação de Áudio Avançada)
AJAX - *Asynchronous Javascript And XML* (Javascript e XML Assíncronos)
ASP - *Active Server Pages*
CEP - Código de Endereçamento Postal
CSS - *Cascading Style Sheets*
DVD - *Digital Video Disc*
FLV - *Flash Video*
FTP - *File Transfer Protocol* (Protocolo de Transferência de Arquivo)
GPL - *General Public Licence*
HTML - *HyperText Markup Language*
IDE - *Integrated Development Environment* (Ambiente Integrado de Desenvolvimento)
IP - *Internet Protocol* (Protocolo de Internet)
JDBC - *Java Database Connectivity*
LAN - *Local Area Network*
MPEG - *Moving Picture Experts Group*
ODBC - *Open Data Base Connectivity*
RTCP – *Real-time Transport Control Protocol* (Protocolo de Controle de Transporte em Tempo Real)
RTP - *Real Time Protocol* (Protocolo de Tempo Real)
SQL - *Structured Query Language*
TCP – *Transport Control Protocol* (Protocolo de Controle de Transporte)
UDP - *User Datagram protocol*
VOD - *Video On Demand* (Vídeo Sob Demanda)
VCD - *Video Compact Disc*
VLC - *Video LAN Client*
WMA - *Windows Media Audio*
WMV - *Windows Media Video*
XML - *Extensible Markup Language*

1 INTRODUÇÃO

Iniciamos a apresentação deste trabalho com o conceito segurança. Segundo o dicionário Aurélio, no contexto deste trabalho, segurança é o estado, a qualidade ou a condição de estar seguro, isto é, estar livre de perigo ou risco.

Diante da escalada da violência no mundo, uma tecnologia vem ganhando muito espaço em seu combate e prevenção: são os sistemas de vigilância utilizando câmeras de vídeo. É comum hoje em dia abrirmos um noticiário e vermos notícias sobre imagens de câmeras de segurança que ajudaram a impedir que delitos ocorressem ou, após a sua ocorrência, ajudaram no reconhecimento e conseqüente prisão dos infratores. Atualmente encontramos sistemas de monitoramento com câmeras de vídeo em uma grande parte dos lugares em que freqüentamos. Empresas, shoppings, comércios, repartições públicas, condomínios, residências, nas ruas mais movimentadas das cidades, e em muitos outros lugares esses tipos de sistemas são instalados com o intuito de proporcionar maior segurança.

Com o avanço da tecnologia, os sistemas de monitoramento com câmeras de vídeo tornaram-se cada vez mais sofisticados e representam atualmente verdadeiros aliados da segurança pública e privada. No início tínhamos apenas sistemas de vigilância utilizando câmeras de vídeo com imagens de baixa qualidade monitoradas através de Circuitos Fechados de TV ou CFTV[1]. Estes sistemas eram restritos a pequenas áreas, como os espaços de uma empresa, por exemplo. O armazenamento das imagens capturadas era feito em fitas cassetes somente no próprio local em que o sistema de vigilância estava instalado, o que proporcionava algumas falhas como a subtração das imagens por intrusos para não serem reconhecidos. Com o contínuo avanço dos computadores, da internet e dos equipamentos utilizados nestes sistemas, como câmeras de vídeo, temos sistemas de vigilância muito mais eficientes que permitem que as imagens sejam monitoradas em tempo real de qualquer local do mundo, com qualidade, em geral, satisfatória de

[1] http://pt.wikipedia.org/wiki/Circuito_fechado_de_televisão

imagens. Além disso, o armazenamento dos vídeos produzidos pelo sistema de vigilância pode ser feito em um servidor remoto distante do local onde o sistema está instalado dificultando deste modo, o roubo ou furto desses arquivos de vídeo.

Os avanços da tecnologia de equipamentos eletrônicos e a diminuição dos custos de produção vêm barateando estes sistemas que, por isso, estão ficando cada vez mais comuns no nosso quotidiano. Entretanto, o custo de instalação de um sistema de monitoramento com câmeras de vídeo pode ainda apresentar alto custo, não sendo acessível a todos os usuários potenciais.

Neste trabalho desenvolvemos um sistema de monitoramento com câmeras de vídeo de baixo custo que tem por objetivo ser acessível a qualquer pessoa que possua computador ligado à *internet*. Graças aos grandes avanços no setor de informática e à popularização da rede mundial, percebe-se uma quantidade cada vez maior de usuários de banda larga que possuem ou podem adquirir a preço baixo câmeras de vídeo que são ligadas diretamente aos seus computadores: as câmeras web ou *webcams*.

Além de ser utilizado como sistema de segurança no monitoramento de áreas como interiores e exteriores de residências ou empresas, este sistema pode ter outras aplicações como, por exemplo, uma mãe observar seus filhos pequenos que brincam em um ambiente da casa enquanto ela trabalha ou faz outra coisa em outro lugar da casa. A aplicação pode variar de acordo com a necessidade do usuário e evidentemente respeitando as limitações que uma câmera *web* pode oferecer em relação às câmeras especificas de sistemas de vigilância.

1.1 OBJETIVOS

O objetivo geral desta monografia, assim como seus objetivos específicos, são apresentados nesta seção.

1.1.1 Objetivo Geral

O objetivo principal deste trabalho é o desenvolvimento de um sistema *web* de vigilância utilizando câmeras de vídeo que tenha um custo de implantação acessível à maioria das pessoas.

A principal contribuição deste trabalho é oferecer às pessoas uma solução alternativa aos atuais sistemas de vigilância com câmeras de vídeo disponíveis no mercado que são pouco acessíveis em decorrência de seus altos custos.

Além disso, toda a tecnologia utilizada para o desenvolvimento do *software* é livre.

1.1.2 Objetivos específicos

Os objetivos específicos deste trabalho estão listados a seguir:

1. O sistema desenvolvido deve ser capaz de capturar, transmitir e exibir remotamente as imagens de uma câmera conectada ao computador do usuário;
2. O sistema deve ser capaz de salvar os vídeos capturados localmente no computador do usuário;
3. O sistema deve gerenciar todos os usuários e suas respectivas câmeras de vigilância;
4. O sistema deve prover um site *web* onde o usuário possa ter acesso às suas câmeras de vigilância e aos outros recursos do sistema;
5. O sistema deve permitir a visualização das imagens em tempo real através de acesso ao site do sistema via navegador *web*;
6. O sistema deve prover segurança aos dados dos usuários, impedindo que pessoas não autorizadas tenham acesso às imagens e dados pessoais dos mesmos.

1.2 ORGANIZAÇÃO DA MONOGRAFIA

A apresentação deste trabalho está organizada da seguinte forma. No capítulo 2 temos a fundamentação teórica, onde são apresentados conhecimentos que deram suporte ao desenvolvimento do sistema que apresentamos nesta monografia. No capítulo 3 temos a apresentação do sistema desenvolvido neste trabalho, o Sistema Safe House. O capítulo 3 mostra o desenvolvimento do sistema e sua arquitetura, bem como também a avaliação do mesmo. Enfim, no capítulo 4 temos as conclusões a respeito deste trabalho, assim como as perspectivas para trabalhos futuros baseados neste.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo apresentamos conhecimentos que foram fundamentais no desenvolvimento da solução apresentada nesta monografia.

2.1 Streaming

Streaming é uma tecnologia que permite reproduzir conteúdos multimídia no momento em que se inicia o recebimento dos dados. Este conceito é diferente do *download*, pois neste é preciso esperar o arquivo ser transferido integralmente para iniciar a sua reprodução. A tecnologia *streaming* (fluxo ou fluxo de mídia) é freqüentemente utilizada para distribuir conteúdo multimídia através da *internet*. Meios de entretenimento e informação utilizam uma forma de *streaming* como televisão e rádio (TOPIC, 2002).

Durante um *streaming,* o cliente solicita ao servidor o início da transferência dos dados multimídia que pode ser, por exemplo, um arquivo de vídeo. O servidor, por sua vez, começa a lhe mandar os dados multimídia em um fluxo contínuo, enviando pacotes periodicamente, de acordo com a taxa de amostragem usada para a sua construção. O cliente usa um *buffer* para salvar parte da informação, visando garantir a continuidade da reprodução considerando a variação do retardo na recepção dos pacotes. Após um retardo estipulado a reprodução inicia ao mesmo tempo em que continua o *download*. A figura 2.1 mostra o funcionamento de *streaming*:

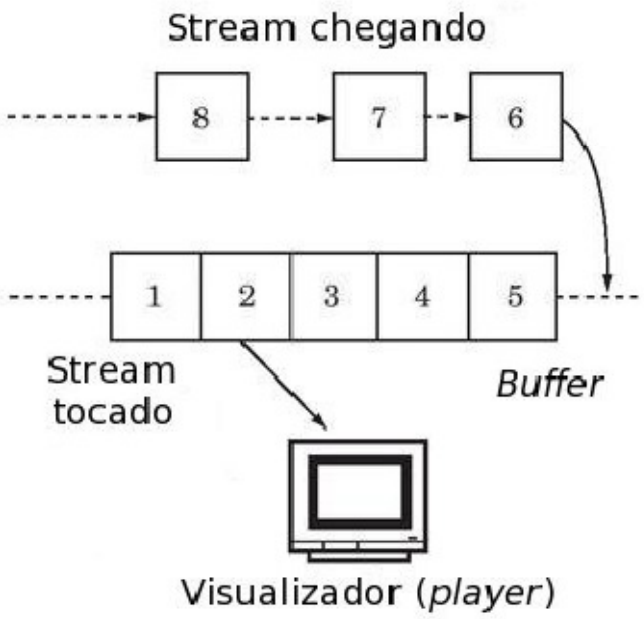

Figura 2.1 – Fluxo de *Stream* (TOPIC, 2002)

Se em algum momento a conexão sofrer decréscimos de velocidade, o que pode causar maior atraso no recebimento do *stream*, nesse caso se utiliza os dados que estão no *buffer*, de modo que um pouco dos efeitos causados por esse decréscimo podem ser contornados. Podemos citar alguns fatores que causam atraso no recebimento do *stream*: tráfego na rede, a largura de banda disponível, o número de roteadores no caminho, a qualidade e o tamanho do vídeo. Se a comunicação for perdida durante muito tempo, o *buffer* se esvazia e a execução do arquivo é interrompida (inanição) até que se restabeleça a comunicação.

A tecnologia de *streaming* mostra-se sob duas formas: sob demanda (*on demand*) e ao vivo (*live*). No *streaming* sob demanda os dados são armazenados em um servidor aguardando alguma requisição por parte do usuário para começar a transmissão dos dados. Em *streaming* ao vivo pacotes multimídia são enviados por uma fonte de maneira contínua, independente de terem sido capturados ou não em tempo real. Neste último tipo de comunicação, o usuário se conecta à fonte, ao invés de requerer uma mídia específica previamente armazenada. Este é o caso de um sistema de vigilância com câmeras de vídeo em tempo real.

O usuário, no *streaming* sob demanda, possui a liberdade de controlar individualmente o que está visualizando, ao contrário do que ocorre no *streaming* ao vivo, onde o controle é limitado por causa do compartilhamento do streaming. A Figura 2.2 ilustra os tipos de *streaming* possíveis, o modo de transmissão (*unicast*, *multicast*, unidirecional e bidirecional) e a experiência do usuário no recebimento dos dados (individual ou compartilhada).

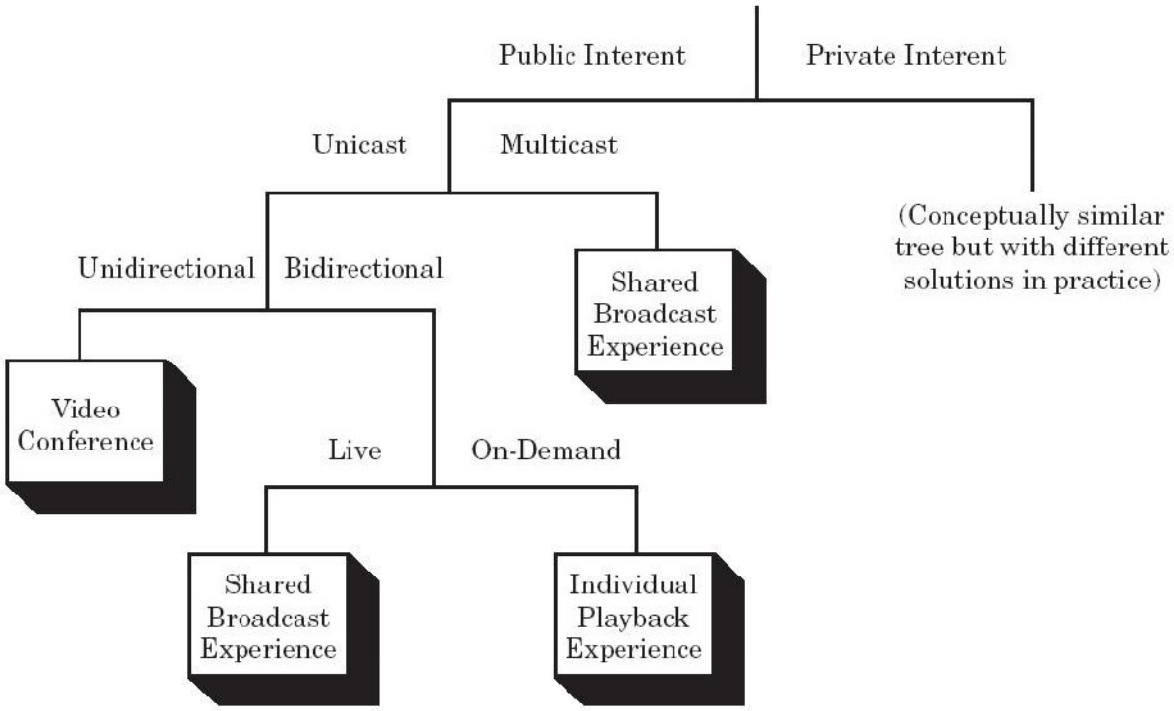

Figura 2.2 – Árvore dos tipos e formas de *streaming* (TOPIC, 2002)

2.1.1 Vídeo sob demanda

Segundo Pisa em seu artigo “Vídeo par a par”, no processo de Vídeo sob Demanda (*Video on Demand - VoD*), os vídeos ficam armazenados em um disco no formato de arquivos. Neste modelo, devido aos vídeos ficarem disponíveis para o usuário, é possível avançar ou retroceder no vídeo, recomeçar ou parar a exibição, pois os vídeos estão prontos e o usuário pode visualizá-los no momento em que preferir. O processo de exibição é similar a assistir um DVD. A Figura 2.3 mostra o processo de transmissão de vídeo sob demanda, onde pode ser visualizado o acúmulo de dados (eixo Y) em relação ao tempo decorrido (eixo X). Os itens (1) e (2) da mesma figura, demonstram os passos dados pelo servidor para capturar um vídeo e posteriormente enviá-lo para a rede. A barra vertical assinalada na Figura 2.3 mostra o momento em que o vídeo é reproduzido em relação à recepção do vídeo transmitido.

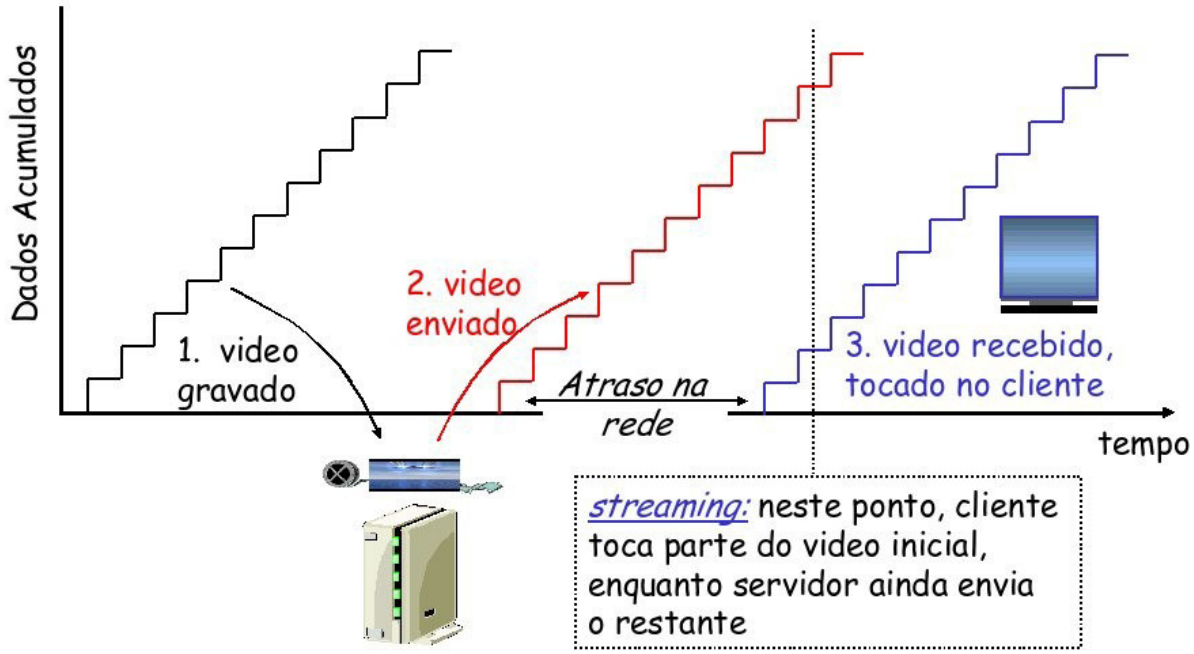

Figura 2.3 – *Streaming* de *Video On Demand* (KUROSE, 2004)

O controle da exibição pode ser feito através de *downloads* ou de um protocolo de controle como o RTCP. No caso de sistemas que realizam o *download* do arquivo, o usuário executa as funções de controle no arquivo que está no seu computador. Já os *softwares* que utilizam protocolos de controle, podem requisitar na *Internet* as mudanças realizadas pelo usuário na exibição do vídeo. Por exemplo, se um usuário deseja avançar no vídeo, o protocolo de controle requisita apenas a parte mais a frente do vídeo, dispensando a transmissão de um trecho do vídeo.

2.1.2 Streaming ao vivo (Live Streaming)

No *streaming* ao vivo de vídeo, também conhecido como difusão de vídeo, o usuário deve assistir aos vídeos enquanto eles são transmitidos, ou seja, independente de quando o usuário começar a acessar o servidor de vídeos, poderá assistir somente ao que estiver passando no momento. No *Live Streaming* o usuário deve se conectar a algum canal de transmissão de vídeo e receber os vídeos em exibição naquele momento. Este tipo de *streaming* é semelhante à transmissão de filmes pela televisão, o usuário não consegue assistir uma programação desde o início que começa às 17 horas se liga a televisão e sintoniza o canal às 18 horas.

Pela configuração da transmissão, os vídeos não podem ser avançados, retrocedidos ou pausados. Contudo, alguns tocadores possuem mecanismos que simulam as funções de retroceder e pausar, através da gravação no disco local do usuário. Mas jamais é possível avançar, pois o vídeo ainda não foi transmitido. Assim, o processo é semelhante à exibição de televisão.

O *Live streaming* permite que transmissões ao vivo sejam enviadas pela *Internet*, utilizando equipamentos como *webcam*, placas de captura e filmadoras digitais para capturar os vídeos. Além de possibilitar a transmissão de vídeo ou áudio em tempo real, neste modelo também é possível enviar conteúdos que estejam armazenados no computador ou em outra forma de mídia, porém na forma de “ao vivo”, sem a possibilidade de retrocesso.

2.2 Soluções semelhantes

Aqui apresentaremos alguns sistemas semelhantes ao que desenvolvemos e que será apresentado em breve.

2.2.1 Sistema GotoCamera

O sistema *GotoCamera*[2] é um serviço de monitoramento utilizando *webcams* que possui algumas semelhanças com o sistema apresentado nesta monografia. Este sistema é uma aplicação *web* composto por um programa cliente e por um servidor disponibilizado pela empresa em um site *Web*. O *software* é responsável por funções como a geração e processamento das imagens e fica instalado no computador do usuário. O site da Empresa, por sua vez, disponibiliza aos usuários recursos que o *software* permite acessar remotamente.

Apesar das semelhanças apresentadas por esta solução, ela tem objetivos diferentes da que é proposta neste trabalho. A seguir estão relacionadas as principais características e recursos que este sistema promete:

- Funciona com a maioria das câmeras *web* atualmente disponível no mercado;
- Suporta detecção de movimento;
- As câmeras podem ser monitoradas pelos usuários através do site do sistema a partir de um computador ou outros dispositivos que possuam navegador *web*;
- Possui armazenamento *online*, isto é, os vídeos podem ser armazenados no servidor do sistema e visualizados pelo usuário através do site do sistema.
- As câmeras podem ser compartilhadas com outros usuários do sistema;
- Um usuário pode ter até quatro câmeras nesse sistema.

Fizemos um teste com o *GotoCamera* para verificar algumas de suas

[2] http://www.gotocamera.com

características e obtivemos alguns resultados a seu respeito. As figuras 2.4 e 2.5 e mostram o sistema GotoCamera em funcionamento:

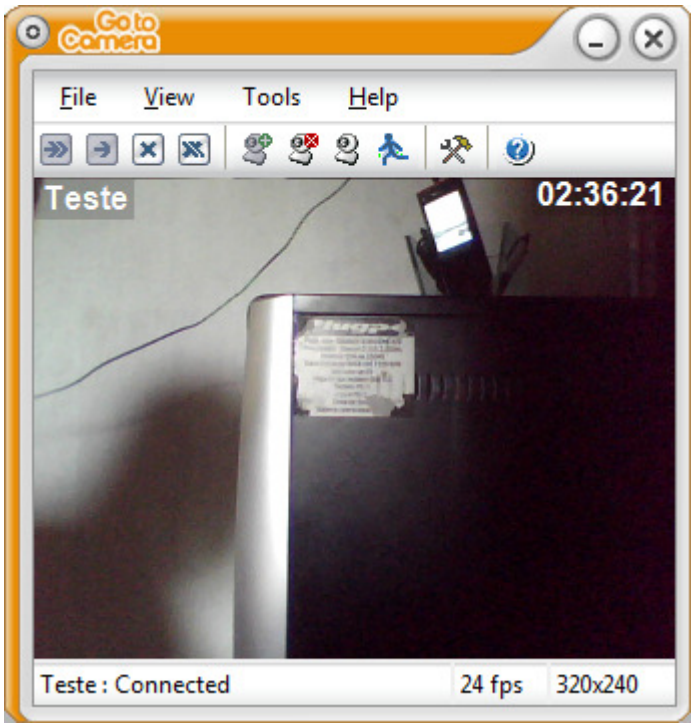

Figura 2.4 – Software *GotoCamera* em funcionamento.

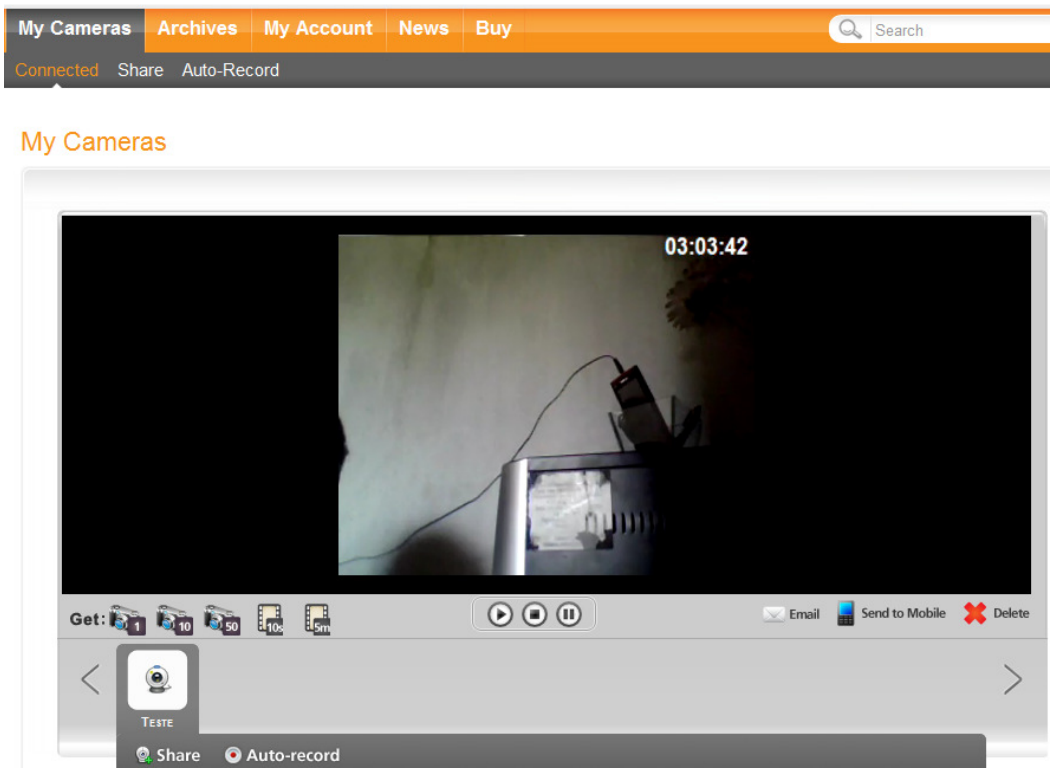

Figura 2.5 – Site do sistema GotoCamera em funcionamento.

Na figura 2.4 temos a imagem do *software* instalado no computador do usuário realizando a tarefa de captura e transmissão das imagens. Enquanto isso, na segunda figura, pode-se ver a imagem do site do sistema em que o usuário tem acesso às imagens geradas no *software* e aos recursos disponibilizados pelo sistema remotamente.

A partir do teste realizado no sistema GotoCamera podemos obter algumas informações mais detalhadas a respeito do mesmo. Como pontos positivos, temos a facilidade de manuseio do *software* e do sistema em geral, o recurso de detecção de movimento e o armazenamento *online*. Apesar de possuir características interessantes, este software não é gratuito, ou seja, é cobrado um valor de aproximadamente 40 dólares por ano para que o usuário possa desfrutar de seus recursos. O sistema é oferecido gratuitamente apenas por um curto período de tempo para que novos usuários possam testá-lo. Outra informação relevante sobre este sistema é o atraso da transmissão do vídeo em tempo real que é exibido

através do site. No nosso teste o vídeo teve um atraso de cerca de 50 segundos, isto é, o frame capturado demorou aproximadamente 50 segundos para ser exibido no visualizador do site do sistema. Mais informações sobre este sistema podem ser encontradas em http://www.gotocamera.com. Vale ressaltar que o acesso e o teste do sistema foram realizados em 12/04/2011.

2.2.2 Sistema WebCam Monitor

O sistema WebCam Monitor[3] é um *software* de vigilância que, assim como o sistema apresentado nesta monografia, utiliza *webcams* para capturar vídeos de segurança. Diferentemente do GotoCamera este sistema não possui um site onde recursos podem ser acessados remotamente. Nele todos os recursos são disponibilizados pelo *software* WebCam Monitor, que fica instalado no computador do usuário.

Este sistema tem objetivos semelhantes ao GotoCamera e algumas semelhanças com a solução apresentada nesta monografia. A seguir temos as principais características e recursos deste *software*:

- Além de poder utilizar *webcams*, este *software* também suporta o uso de câmeras IP, que são câmeras próprias para vigilância;
- Dá suporte a microfones tornando possível a produção de vídeos com áudio;
- Possuí suporte á detecção de movimento e também detecção de ruído, podendo gerar alertas se configurado para tal;
- É possível ampliar o vídeo em até 200% através de *zoom* digital;
- As câmeras podem ser monitoradas remotamente através de um servidor de *stream* local, isto é, a visualização em tempo real pode ser feita apenas na rede interna ou pela *internet*. Porém, o monitoramento pela *internet* só é

[3] http://www.deskshare.com/lang/po/wcm.aspx

possível se o computador em que o *software* está executando possuir endereço IP válido;

- Os vídeos podem ser armazenados localmente ou, ainda, em um endereço remoto utilizando um serviço de FTP para fazer o *upload* dos vídeos;

- O usuário pode ter uma ou mais câmeras nesse sistema (não foi estabelecida uma quantidade máxima de câmeras).

Fizemos um teste com o *WebCam Monitor* com o objetivo de verificar algumas de suas características e obtivemos alguns resultados sobre o mesmo. As figuras 2.6 e 2.7 apresentam imagens do *software* WebCam Monitor em funcionamento durante o teste realizado:

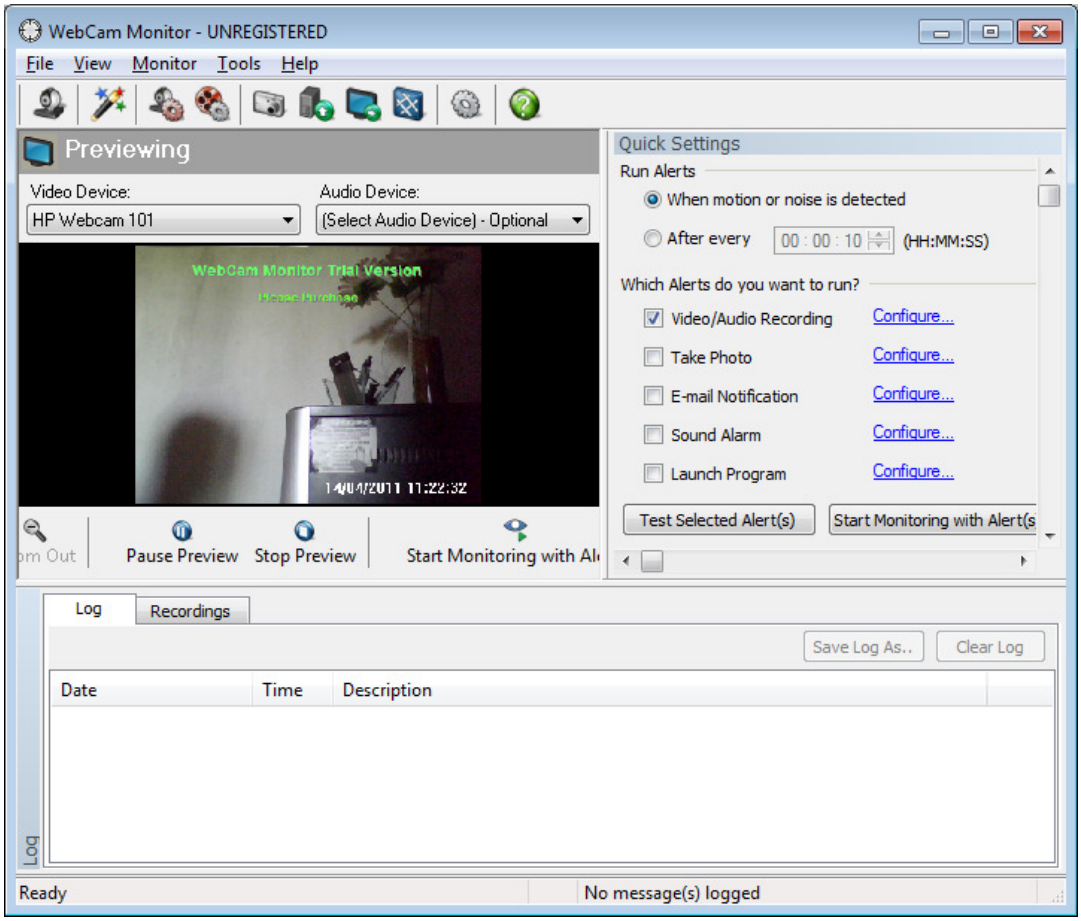

Figura 2.6 – *Software* WebCam Monitor em funcionamento.

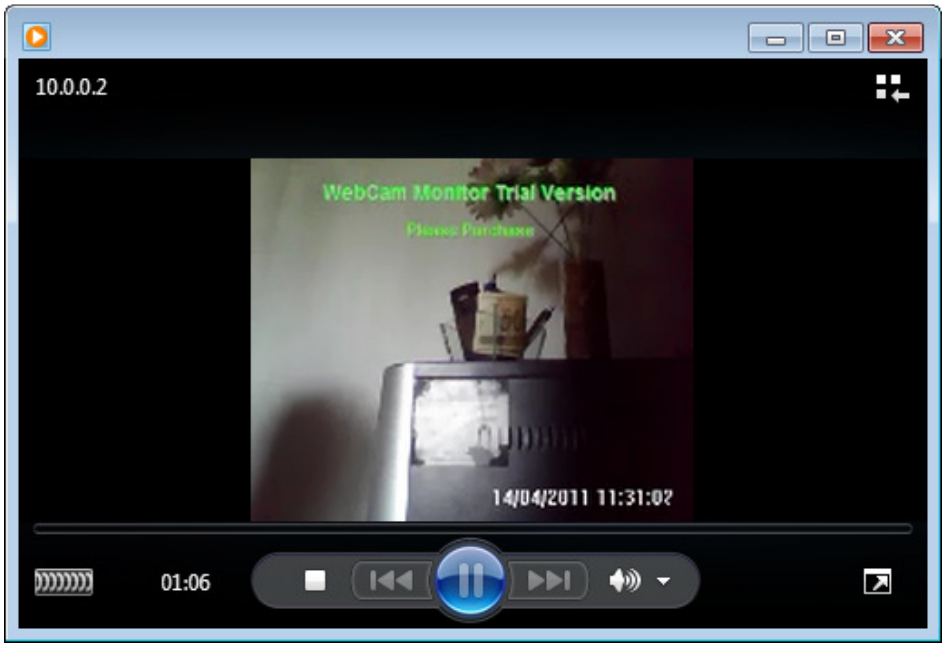

Figura 2.7 – *Streaming* de vídeo com o *software* WebCam Monitor.

Na primeira figura podemos ver a imagem do *software* WebCam Monitor que está instalado no computador do usuário realizando a tarefa de captura e exibição das imagens, assim como disponibilizando alguns recursos para o usuário. Na segunda figura, podemos ver as imagens geradas e transmitidas pelo software WebCam Monitor sendo exibidas remotamente utilizando o *software* Windows Media Player (WMP)[4]. É interessante observar que a geração e *streaming* das imagens foram feito em uma mesma rede, pois seria necessário ter um computador com o IP válido para a realização deste teste através da *internet*.

Analisando os testes realizados com o *software* WebCam Monitor podemos ver que este *software* possui alguns recursos muito bons como o suporte à câmeras IP, por exemplo. Podemos citar também como ponto positivo os recursos de detecção de movimento e detecção de ruído (áudio), assim como o próprio suporte ao vídeo com áudio. Porém, assim como o GotoCamera, este *software* também não é gratuito, isto é, é cobrado um valor de aproximadamente 70 dólares para que o usuário possa utilizar os recursos do mesmo. O custo deste *software* não é tão alto se compararmos aos custos para se instalar um sistema de vigilância através de uma empresa de segurança, mas ele apresenta algumas características que o

[4] http://windows.microsoft.com/pt-BR/windows/products/windows-media-player

tornam muito limitado. Uma dessas características é o fato dele não oferecer suporte para a visualização das imagens em tempo real remotamente. Ele oferece apenas um serviço de *stream* de vídeo que pode ser usado em redes locais ou até pela *internet* (desde que o computador em que está executando possua um endereço IP fixo), mas sem nenhuma segurança dos dados, ou seja, qualquer pessoa que possua o endereço do serviço de *stream* de vídeo poderia ter acesso às imagens. O armazenamento remoto utilizando FTP também não é uma solução muito inteligente, pois requer que o usuário do *software* tenha acesso a um *host* com serviço de FTP, isso geraria mais gastos. Mais informações sobre este *software* podem ser encontradas em http://www.deskshare.com/lang/po/wcm.aspx, que foi acessado e testado em 14/04/2011.

3 SISTEMA SAFE HOUSE

Neste capítulo apresentamos o Sistema Safe House, sistema desenvolvido neste trabalho de conclusão de curso, e mostramos o seu desenvolvimento.

3.1 APRESENTAÇÃO DO SISTEMA

O Sistema Safe House, expressão inglesa que significa casa segura, é um sistema de vigilância com câmeras de vídeo que utiliza *webcams* na captura dos vídeos. Este sistema foi desenvolvido com o objetivo de ter um baixo custo, portanto uma alternativa aos sistemas atuais que geralmente apresentam um alto custo de implantação.

Um usuário do sistema Safe House necessita apenas de um computador pessoal com acesso à *internet* banda larga e de uma *webcam* conectada a este computador para que possa implantar seu sistema de vigilância. Hoje em dia computadores pessoais com acesso à *internet* banda larga são muito comuns nas residências da maioria das pessoas, e uma *webcam* pode ser comprada por preços bem acessíveis. Então, este sistema de vigilância pode ser facilmente montado por um custo muito baixo.

Por ser um sistema desenvolvido em um trabalho acadêmico e por uma única pessoa, ele apresenta suas limitações e evidentemente não possuí todos os recursos que um sistema de vigilância de uma empresa de segurança possui. Porém este sistema apresenta algumas funcionalidades e recursos razoáveis e que podem ser aproveitados satisfatoriamente em um sistema de vigilância que tem por objetivo um baixo custo.

A seguir apresentamos os recursos e as principais características do Sistema Safe House:

- Visualização de qualquer lugar do mundo em tempo real dos vídeos através de um navegador *web* desde que se tenha um computador com *internet* banda larga;
- Gravação dos vídeos de segurança no computador pessoal do usuário;
- Baixo custo de implantação do sistema, necessitando apenas de um computador com *internet* banda larga e uma *webcam*;
- Possibilidade de utilização de várias câmeras;
- Site *Web* com um sistema de controle de acesso que proporciona segurança aos dados do usuário;
- Gerenciamento das câmeras de vídeo e dos dados do usuário pelo site *Web* do sistema;
- Sistema de segurança que impede que pessoas não autorizadas tenham acesso às imagens das câmeras de vídeo dos usuários;

3.2 METODOLOGIA DE DESENVOLVIMENTO

No desenvolvimento do Sistema Safe House foi utilizada a seguinte metodologia:

- Revisão bibliográfica acerca do tema deste trabalho, assim como também sobre o desenvolvimento de aplicações para redes de computadores e aplicações *web*;
- Análise e planejamento do sistema em desenvolvimento, inclusive utilizando técnicas de engenharia de *software*;
- Escolha das linguagens de programação mais adequadas para o desenvolvimento dos *softwares* que compõem este sistema;

- Escolha dos recursos e ferramentas computacionais necessárias para o desenvolvimento e testes deste sistema;
- Desenvolvimento dos aplicativos integrantes do sistema e do site *Web* que compõe o sistema;
- Realização de testes para a validação do sistema desenvolvido.

3.3 RECURSOS UTILIZADOS NO DESENVOLVIMENTO

No desenvolvimento deste trabalho foi necessária a utilização dos seguintes recursos:

- Livros e outros meios de pesquisa relacionados às diversas áreas envolvidas no desenvolvimento deste sistema, como redes de computadores, aplicações multimídia, linguagens de programação, engenharia de *software*, dentre outros;
- Microcomputador pessoal onde foram instaladas todas as ferramentas necessárias para o desenvolvimento e testes do sistema;
- Microcomputador com *internet* banda larga de alta velocidade e com endereço IP válido (IP fixo) utilizado como servidor do sistema. Componentes do sistema como o aplicativo Safe House Server, o site e o banco de dados integrado necessitam de um computador como esse para testes;
- *Webcams* para serem utilizadas como dispositivos de captura de vídeo nos testes do sistema;
- O *software* NetBeans IDE[5] para o desenvolvimento de alguns aplicativos do sistema. O NetBeans IDE é um ambiente de desenvolvimento integrado (IDE) gratuito e de código aberto para desenvolvedores de *software* nas

[5] http://www.netbeans.org

linguagens Java, C, C++, PHP, Groovy, Ruby, entre outras;

- O *software* Aptana Studio[6] para o desenvolvimento do site do sistema. O Aptana Studio é um *software* de código aberto e gratuito para ambiente de desenvolvimento integrado (IDE) que suporta as linguagens CSS, HTML, JavaScript, ScriptDoc, XML, dentre outras;

- A linguagem de programação Java para o desenvolvimento do aplicativo cliente Safe House e do aplicativo servidor Safe House Server;

- As linguagens de programação HTML, PHP, Javascript e CSS no desenvolvimento do site do sistema;

- A linguagem SQL e o sistema de gerenciamento de banco de dados MySQL[7], utilizados no banco de dados do sistema;

- O *software* de código aberto VLC[8] Media Player para integrar o sistema realizando algumas tarefas de suma importância para o sistema;

- O aplicativo JW Player[9] para a exibição dos vídeos no site do sistema. O JW Player é um tocador de mídia de código aberto utilizado incorporado a sites *Web* para reproduzir áudio e/ou vídeo;

- O software Apache[10] nos testes do sistema. O Apache é um servidor web livre e foi utilizado na hospedagem do site do sistema.

[6] http://www.aptana.com
[7] http://www.mysql.com
[8] http://www.videolan.org/vlc
[9] http://www.longtailvideo.com/players
[10] http://www.apache.org

3.4 ANÁLISE ECONÔMICA

A análise econômica visa a estabelecer se o projeto de *software* gerará lucro, e se a receita gerada será o suficiente para cobrir os custos. Porém neste caso, como se trata de um trabalho para conclusão de curso que tem por objetivo o desenvolvimento de um sistema de baixo custo, foram analisados apenas os custos para o desenvolvimento e implantação do sistema.

No caso dos custos para o desenvolvimento deste sistema garantimos gastos muito pequenos e dentro dos padrões para este tipo de projeto. Isso é devido à escolha de ferramentas gratuitas, utilização de *softwares* de códigos abertos, e o fato de os equipamentos necessários serem populares e de baixo custo.

Em se tratando dos custos para implantação do *software*, segue-se a mesma tendência dos custos de desenvolvimento. Os recursos requeridos para a instalação e uso do sistema são bem acessíveis do ponto de vista econômico. Isso garante que o objetivo principal do sistema deve ser atendido, isto é, o desenvolvimento de um sistema de vigilância com câmeras de vídeo de baixo custo.

3.5 ANÁLISE DOS REQUISITOS

Na análise dos requisitos para a construção de um *software* geralmente se extrai os requisitos a partir de informações passadas pelo cliente que deseja o produto. Neste caso, como estamos desenvolvendo um sistema em que ainda não existem clientes ou usuários, desenvolvemos os requisitos do sistema a partir dos objetivos deste trabalho.

Nesta extração e análise dos requisitos do sistema tomamos uma abordagem de organizá-los de acordo com seu grau de importância. Assim, requisitos considerados mais importantes para este projeto tiveram maior prioridade no desenvolvimento.

Os requisitos deste *software* foram divididos em requisitos funcionais e requisitos não-funcionais. Os requisitos funcionais descrevem as funcionalidades

que se espera que o sistema disponibilize, de uma forma completa e consistente. Enquanto que, os requisitos não-funcionais referem-se a aspectos não-funcionais do sistema, como restrições nas quais o sistema deve operar ou propriedades emergentes do sistema. A seguir estão listados os requisitos elaborados para esse sistema:

- Requisitos funcionais:
  - Capturar imagens de uma câmera (*webcam*) conectada ao computador do usuário;
  - Transmitir as imagens via *internet* no momento em que são capturadas;
  - Exibir as imagens transmitidas em tempo real para o usuário do sistema;
  - Armazenar os vídeos localmente no computador do usuário do sistema;
  - Gerenciar os usuários do sistema;
  - Gerenciar as câmeras de vigilância dos usuários conectadas ao sistema;
  - Possuir um site *web* para fazer a interface do sistema com os usuários;
  - Permitir a visualização das imagens em tempo real através do site do sistema;
  - Prover um cadastro dos usuários através do site do sistema;
  - Permitir ao usuário alterar seus dados cadastrais através do site *web* do sistema;
  - Permitir ao usuário gerenciar suas câmeras através do site *web* do sistema;
  - Prover a segurança dos dados pessoais dos usuários;
  - Prover a segurança dos vídeos transmitidos em tempo real impedindo que pessoas não autorizadas tenham acesso ao mesmo.

- Requisitos não-funcionais:

  - Atraso máximo na transmissão das imagens em tempo real deve ser compatível com sistemas atuais que tenham a mesma finalidade;

  - Prover *softwares*, que o usuário tenha contato direto, com interfaces amigáveis;

  - Prover um site com interface amigável com os usuários.

3.6 ESPECIFICAÇÃO

O sistema foi especificado de acordo com os requisitos extraídos apresentados. Após a análise dos requisitos, chegamos à conclusão de que usaríamos uma abordagem cliente – servidor, porém usaríamos um modelo dessa abordagem modificado de acordo com os nossos propósitos.

A utilização do modelo cliente – servidor neste caso se justifica pelo fato de necessitarmos de um *software* gerenciador centralizado (servidor), e ainda um software que realiza várias funções no computador do cliente e deve interagir com o *software* gerenciador.

Analisando os requisitos especificamos que o sistema deveria ter os seguintes módulos: um aplicativo servidor, um aplicativo cliente, um site *web* para sistema e um banco de dados. O *software* cliente deve ser instalado no computador do usuário e se comunicará com o servidor via *internet*. O *software* servidor deve ser instalado em um computador *host* e será responsável pelo gerenciamento dos *softwares* clientes e pelo tratamento das requisições partidas do site. O banco de dados será comum ao *software* servidor e ao site do sistema, sendo responsável por armazenar as informações dos usuários.

Para facilitar a codificação especificamos as tarefas de todos os componentes do sistema. A seguir temos todos os componentes do sistema com suas devidas atribuições.

3.6.1 Software cliente

O *software* cliente será denominado Safe House e deverá realizar as seguintes tarefas:

- Capturar imagens de uma câmera do tipo *webcam*;
- Armazenar as imagens capturadas localmente no computador;
- Gerar a o *streaming* ao vivo do vídeo a ser transmitido;
- Se conectar ao servidor passando dados de autenticação do usuário e registro da câmera;
- Transmitir as imagens capturadas em tempo real para o servidor quando for solicitado pelo mesmo;
- Apresentar uma interface de acesso com os campos destinados a usuário e senha e botão de conexão;
- Apresentar uma interface para configuração de dados. São dados configuráveis a escolha do dispositivo de captura de imagens, o nome ou identificador da câmera de vídeo, e outros dados que seja necessária a configuração que apareçam durante a implementação;

Na revisão bibliográfica feita antes deste projeto, constatamos que desenvolver um *software* com essas características seria um tanto quanto complexo e despenderia muito tempo. Então, optamos por utilizar um *software* de código aberto para realizar algumas dessas funções (captura, armazenamento e geração do *streaming* do vídeo) e desenvolveríamos um *software* para realizar o restante das funções (conectar usuário e sua câmera com o servidor, transmitir o *streaming* de vídeo em tempo real, fazer interface com o usuário e gerenciar o *software* livre utilizado).

Após algumas pesquisas, descobrimos o *software* livre VLC que se encaixava perfeitamente no nosso propósito. O *software* VLC então será

responsável pela captura, armazenamento e geração do *streaming* dos vídeos e funcionará como um serviço para o *software* cliente a ser desenvolvido.

3.6.2 Software servidor

O *software* servidor será denominado Safe House Server e terá as seguintes atribuições:

- Iniciar o serviço de conexões com *softwares* clientes. Este serviço será realizado na porta 1991;
- Iniciar o serviço de requisições de vídeo. Este serviço será realizado na porta 1990;
- Autenticar um usuário e registrar sua câmera junto ao banco de dados quando ocorrer uma conexão com um *software* cliente. A autenticação do usuário deverá ser feita utilizando um *login* e uma senha escolhidos pelo usuário no cadastro do mesmo;
- Gerar um número inteiro de nove dígitos aleatório e único para identificar a câmera do usuário. Este identificador será obrigatório nas requisições de vídeo devendo ser alterado toda vez que houver o registro da câmera. Este identificador da câmera deverá ser guardado no banco de dados do sistema de forma que somente o usuário dono da câmera tenha acesso ao mesmo. O objetivo deste identificador é proteger os vídeos do usuário impedindo que pessoas não autorizadas tenham acesso aos vídeos;
- Este servidor deverá ter a capacidade de atender múltiplas requisições ao mesmo tempo. Também deverá ter a capacidade de gerenciar múltiplas conexões com os aplicativos clientes;

3.6.3 Banco de dados

O sistema deverá utilizar o sistema de gerenciamento de banco de dados MySQL que utiliza a linguagem SQL como interface. O banco de dados do sistema deverá possuir uma tabela para armazenar as informações do usuário que deverá ter os seguintes campos:

- Nome: campo para armazenar o nome do usuário;
- Email: campo para armazenar o email do usuário;
- Senha: campo destinado ao armazenamento da senha do usuário;
- Endereço: campo para armazenar o endereço do usuário;
- Complemento: campo para armazenar complementos sobre o endereço do usuário, como um ponto de referência, por exemplo;
- CEP: campo destinado ao código de endereçamento postal do usuário;
- Telefone: campo destinado ao telefone do usuário;
- Câmeras: campo destinado ao armazenamento dos nomes das câmeras do usuário. Os nomes serão separados por um caractere especial escolhido na etapa de codificação;
- Deverá existir mais um campo para cada câmera do usuário. Estes campos serão destinados ao armazenamento do identificador único das câmeras que será utilizado nas requisições de vídeo ao vivo.

Alguns campos adicionais poderão ser criados nessa tabela de usuário do banco de dados do sistema de acordo com a necessidade que venha a surgir durante o processo de implementação do *software*;

3.6.4 Site web

O site web do sistema é o meio principal de interface do usuário com o sistema, visto que o *software* cliente apresentará apenas algumas opções de

configuração e a interface de conexão com o servidor. Este site deve possuir os seguintes atributos:

- Página inicial com interface amigável com o usuário e com fácil acessibilidade a todas as informações e recursos do sistema;
- O site deverá possuir as seguintes páginas: cadastro de usuários, acesso ao sistema, visualização de informações da conta do usuário, configuração da conta do usuário, monitoramento das câmeras do usuário, informações sobre o sistema e, outras mais que venham a ser necessárias;
- *Menu* de fácil utilização e com os seguintes *links*: acesso á página inicial, acesso as dados do usuário, acesso à página de configuração do sistema, acesso às câmeras do usuário e, acesso às informações sobre o sistema;
- O site deverá ter *links* que facilitem o acesso do usuário à página de *login* do sistema. Também deve haver facilidade no acesso à página de cadastro de usuários do sistema;
- A página de acesso deverá ter um campo para *login*, um campo para senha e um botão para efetuar o acesso;
- A página de cadastro deverá ter campos para todas as informações essenciais ao cadastro de um novo usuário;
- A página de visualização das informações da conta deverá apresentar todos os dados pessoais e câmeras do usuário, exceto a senha do mesmo;
- Na página de configuração o usuário poderá alterar seus dados pessoais, exceto *login*, e adicionar ou remover câmeras de sua conta;
- Na página de monitoramento das câmeras o usuário deverá ter a possibilidade de visualizar as imagens de todas as suas câmeras de vídeo que estejam conectadas no momento;
- Todas as informações de cadastro deverão ser verificadas e em seguida

armazenadas no banco de dados do sistema em seus campos reservados;

3.6.5 Comunicação entre os módulos do Sistema

No sistema a ser desenvolvido precisamos especificar a forma de comunicação entre seus componentes. Neste sistema temos dois tipos de comunicação: a troca de informações comuns, como dados de *login*, e a transferência de vídeo em tempo real ou *live streaming*.

Na comunicação com dados comuns não temos muitas exigências, apenas que os dados cheguem com segurança ao destino. Já na comunicação de mídia, é necessário que os dados cheguem ao destinatário com segurança e com um atraso tolerável para exibição em tempo real.

Especificamos então que toda a comunicação entre os componentes será feita utilizando o protocolo de nível de aplicação HTTP ou diretamente o protocolo de transporte TCP. Nas requisições de *streaming* de vídeo ao vivo do site para o servidor teremos uma comunicação utilizando o protocolo HTTP. Na comunicação entre os *softwares* cliente e servidor teremos uma comunicação utilizando *sockets* TCP.

A melhor forma de fazer transferência de vídeos em tempo real não é utilizando diretamente os protocolos de transporte da pilha TCP/IP. Existem protocolos adequados para a transferência em tempo real, como os protocolos RTP e RTCP, por exemplo. Porém, estes protocolos utilizam o protocolo UDP para transporte, o que necessita que o computador remetente conheça o endereço IP do destinatário para envio direto de datagramas, visto que tal protocolo opera sem conexão. A maioria das pessoas que possuem internet banda larga usa serviços que fornecem somente um IP dinâmico, cobrando um custo adicional por um IP fixo. Neste caso, o uso dos protocolos RTP/RTCP requer soluções mais complexas. O tempo curto para o desenvolvimento deste projeto inviabiliza, portanto, a utilização desses protocolos.

3.7 CODIFICAÇÃO DO SISTEMA

Nesta etapa do projeto temos que implementar os módulos do sistema. Na figura 3.1 é mostrada a arquitetura que foi projetada para o Sistema.

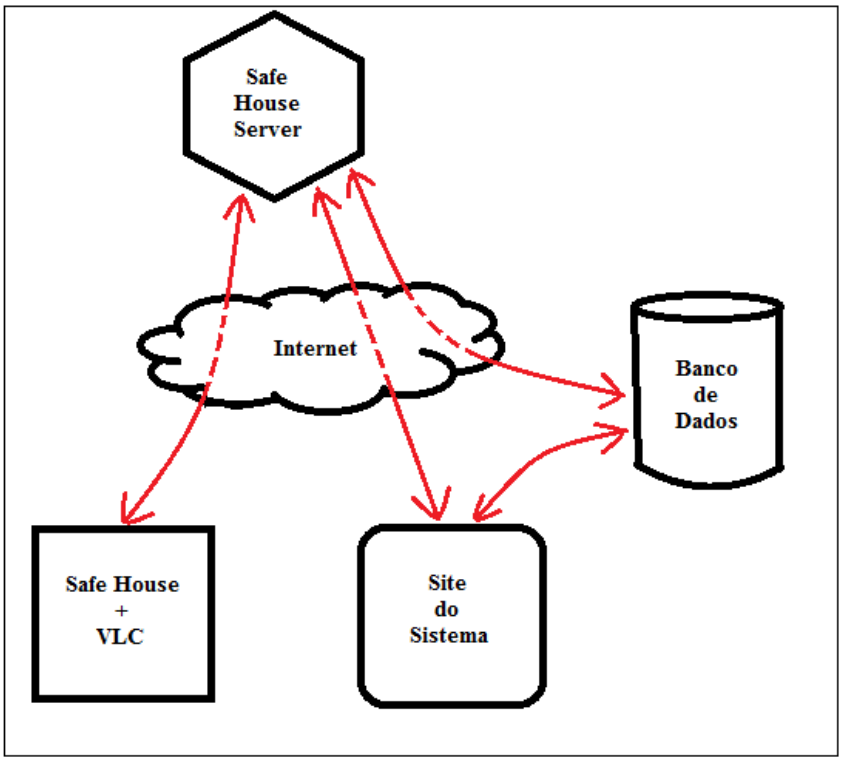

Figura 3.1 – Arquitetura projetada para o sistema.

De acordo com a etapa de especificação, precisamos codificar dois softwares e um site web. Um dos softwares é o aplicativo Safe House, que pertence ao módulo cliente. O outro software é o aplicativo Safe House Server, pertencente ao módulo servidor.

Com o intuito de facilitar o trabalho, dividimos a codificação em duas partes: na primeira parte implementamos os *softwares* dos módulos cliente e servidor enquanto que na segunda parte implementamos o site *web* do sistema.

3.7.1 Implementação dos módulos cliente e servidor

Antes de iniciar a codificação dos *softwares* é necessário definirmos as linguagens de programação que serão utilizadas. Neste caso, escolhemos trabalhar com a linguagem Java. O Java foi escolhido por ser uma linguagem com boas ferramentas e por já ter uma boa experiência no desenvolvimento de aplicações com essa linguagem de programação. Também foi preciso escolher a IDE em que iríamos trabalhar. Pelo mesmo motivo que escolhemos a linguagem Java, escolhemos a IDE NetBeans com ferramenta de programação.

Após definidas as ferramentas e linguagens a serem utilizadas, foi iniciada a codificação. A codificação dividiu-se em várias etapas. Cada etapa de codificação era dividida em três partes: implementação, testes e integração. Na implementação eram feitos os códigos fonte que em seguida, nos testes, eram testados e, por fim, integrados ao sistema. Assim o sistema ficava mais robusto a cada etapa. A seguir listamos as etapas de codificação destes *softwares*:

- 1ª etapa: Implementar no aplicativo Safe House a captura, armazenamento e geração de *streaming* de vídeo usando o *software* VLC;

    - Nesta etapa, o Safe House inicia o VLC através de um comando. Neste comando são passados para o VLC parâmetros de configuração. Estes parâmetros são: formato do vídeo, qualidade do vídeo, endereço local de armazenamento e configurações para a transmissão do *stream* de vídeo. O formato do vídeo escolhido foi o FLV (*flash*), pois é um formato bastante usado na *internet* e isso facilitaria a exibição. A qualidade foi ajustada de maneira que não houvesse prejuízos na qualidade do serviço.

- 2ª etapa: Implementar no servidor o serviço de gerenciamento de clientes e o transporte do *streaming* de vídeo;

    - Nesta etapa, realizamos a comunicação do aplicativo cliente com o servidor através de *socket* TCP. Usando o protocolo TCP/IP conseguimos realizar a transferência do *streaming* de vídeo do computador do usuário para a internet.

- 3ª etapa: implementar no servidor a autenticação dos usuários, e o registro das câmeras junto ao banco de dados, assim como gerar um identificador único para cada câmera;

    - Nesta etapa, integramos o servidor Safe House Server com o banco de dados do sistema. Isso foi possível utilização a API MySQL Connector/J que é um *driver* que possibilita que um aplicativo Java se comunique com um banco de dados SQL. Também, nesta etapa, implementamos o identificador único para a câmera e o serviço de autenticação de um usuário que se conecta ao servidor a partir do aplicativo Safe House.

- 4ª etapa: implementar o tratamento de requisições de vídeo por parte de um usuário no site;

    - Nesta etapa implementamos o sistema de tratamento de requisições de vídeo. Feito isso o *software* servidor já estava pronto para receber solicitações de vídeo e repassá-las a *software* cliente que, por sua vez envia o *stream* de vídeo ao servidor que repassa ao site para ser reproduzido.

- 5ª etapa: finalizar a interface gráfica do *software* Safe House e integrar os dois *softwares* ao site do sistema.

    - Esta etapa foi realizada em conjunto com a construção da parte do site de visualização de vídeo. A parte de visualização do site também foi implementada e integrada ao resto do sistema já desenvolvido.

3.7.2 Implementação do site web

Depois de terminada a primeira parte da codificação, falta agora a segunda parte que é a construção do site do sistema. Para a construção do site do sistema utilizamos várias linguagens de programação web. As linguagens utilizadas são: HTML, PHP, Javastript, CSS e XML.

A primeira etapa da construção do site foi a implementação da página de monitoramento das câmeras. Ela foi construída primeira devido a necessidade de integração com o restante do sistema que precisava ser testado. Em seguida foi construído o restante do site do sistema.

Na página de monitoramento foi utilizado o JW Player para exibir o *stream* de vídeo em tempo real. O JW Player é um reprodutor de mídia de código aberto utilizado incorporado a sites web para tocar áudio ou vídeo.

Na construção do site do sistema foi necessário o desenvolvimento de vários *scripts* nas linguagens PHP e *Javascript*. Também foi utilizado o modelo de desenvolvimento web AJAX. AJAX é o uso metodológico de tecnologias como *Javascript* e XML que utiliza solicitações assíncronas, isso torna as páginas *web* mais interativas e ágeis.

3.8 ARQUITETURA DO SISTEMA

O Sistema Safe House é composto por vários módulos aplicativos que funcionam integrados. Fazem parte do sistema: um *web site*, um aplicativo servidor (denominado Safe House Server), um aplicativo cliente (chamado apenas de Safe House) e um aplicativo de gerenciamento da câmera, este último é um *software* livre chamado VLC. Além disso, o sistema MySQL, que é um gerenciador de banco de dados (SGBD), também integra a arquitetura do sistema. Na figura 3.2 podemos ver a arquitetura do Sistema Safe House com todos os seus componentes.

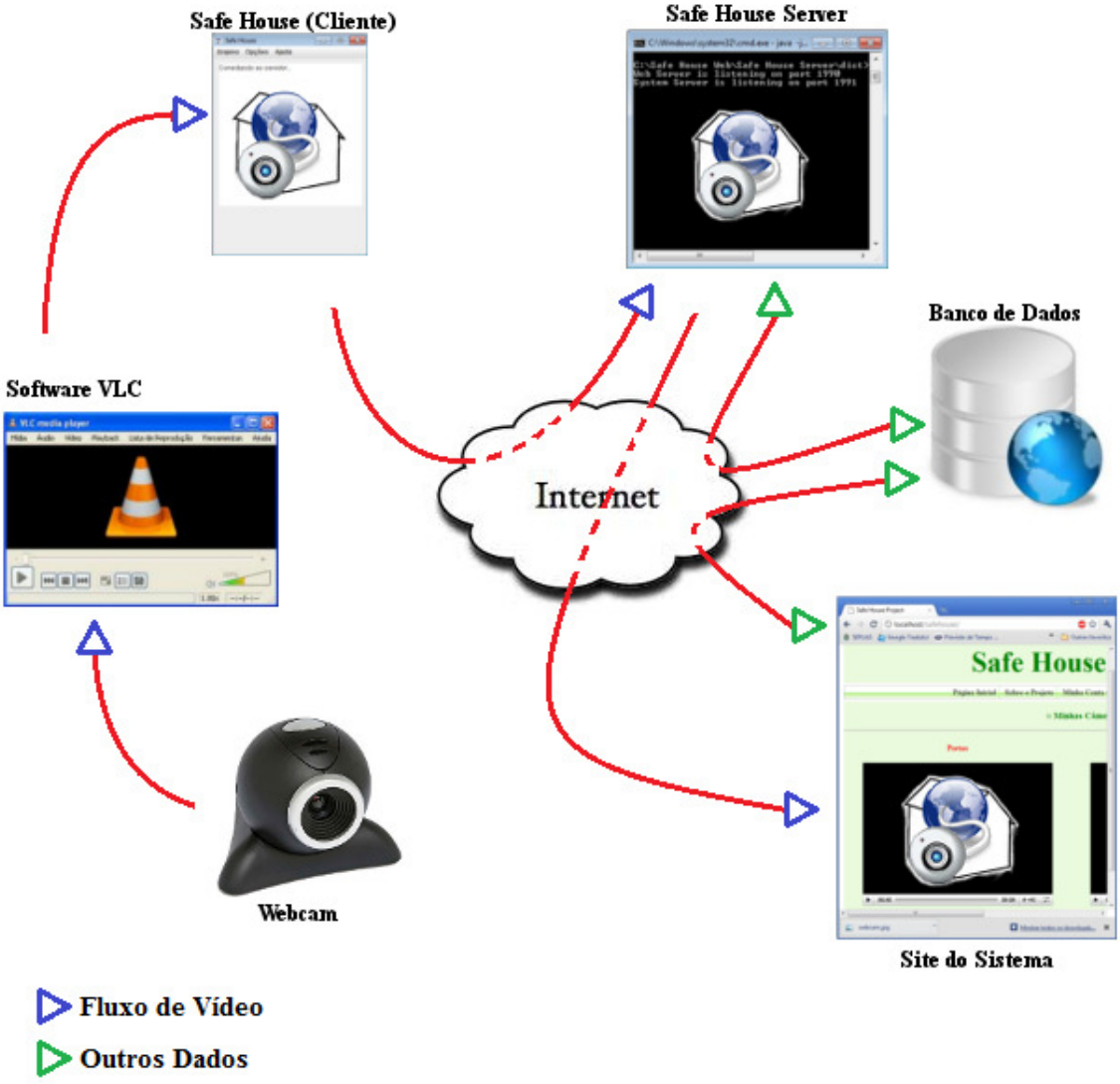

Figura 3.2 – Arquitetura do Sistema Safe House

Como podemos ver na figura 3.1, todos os componentes do sistema estão integrados para proporcionar os serviços para os quais ele foi concebido. O *software* VLC está conectado à câmera de vídeo e é controlado pelo *software* cliente Safe House. Este está conectado ao Safe House Server através da *internet* e, por sua vez, comunica-se com o SGBD do sistema e com o *web site* do sistema.

Nas subseções a seguir será apresentado individualmente cada um dos componentes do Sistema Safe House. Nesta apresentação os componentes serão descritos detalhadamente e também será mostrado seu funcionamento.

3.8.1 Safe House

Safe House é o *software* que é instalado no computador pessoal do usuário em que as câmeras serão conectadas. Este *software* é chamado de aplicativo cliente, pois ele deve se conectar ao aplicativo servidor do sistema, o Safe House Server. O Safe House é responsável por capturar, salvar localmente e enviar o vídeo quando solicitado pelo usuário. Além disso, é nele que são realizadas as configurações relacionadas à captura, armazenamento e envio do vídeo via *streaming*.

O serviço de captura e armazenamento dos vídeos não é feito propriamente pelo aplicativo Safe House. Para isso é utilizado o VLC, ficando o Safe House com as tarefas de gerenciamento, configuração e envio do vídeo. O *software* VLC, como foi mostrado anteriormente, também é um componente do Sistema Safe House e será apresentado mais adiante.

Ao iniciar este aplicativo, o usuário primeiramente visualiza a sua interface inicial. A interface inicial contém uma tela de acesso ao sistema e também um *menu* do tipo barra. Na tela de acesso temos os campos destinados à digitação de usuário e senha e, ainda, uma *checkbox* para marcação de acesso automático e um botão para ativar o início da conexão com o servidor. No *menu* em barra temos algumas opções para o usuário, dentre elas destacamos o botão para a janela de configuração do aplicativo. Na Figura 3.3 é mostrada a tela inicial do aplicativo Safe House.

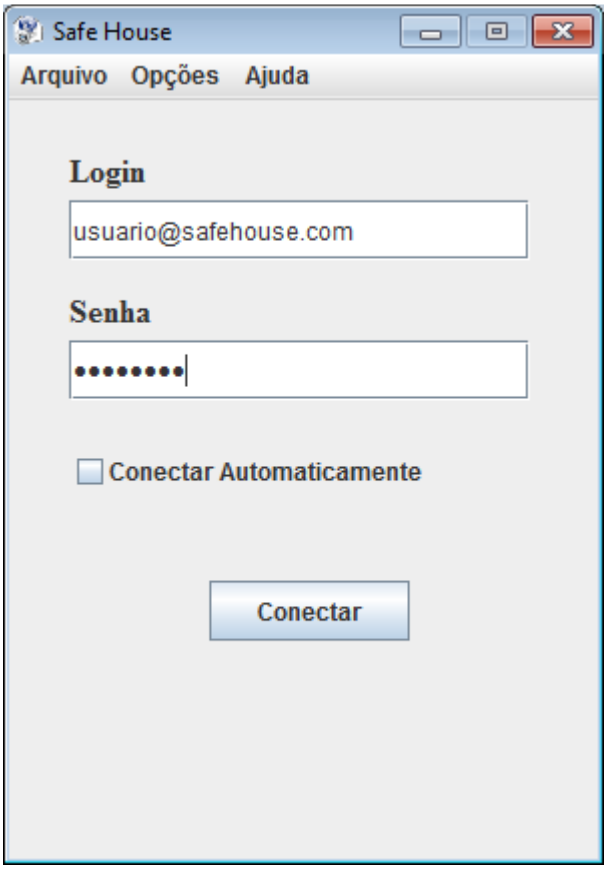

Figura 3.3 – Tela inicial do aplicativo Safe House

Quando o aplicativo é iniciado ele ainda não está conectado ao servidor do sistema, isto é, ao aplicativo Safe House Server. Para se conectar ao servidor é necessário que o usuário insira seu *login* e sua senha que são obtidos através do cadastro do usuário no site do sistema. O usuário tem a opção de se conectar automaticamente nas próximas vezes em que iniciar o programa marcando a *checkbox* "Conectar Automaticamente".

Além do *login* e da senha, para que o usuário possa se conectar ao sistema é necessário que o aplicativo esteja configurado. Então, caso seja a primeira vez que o usuário irá se conectar ao sistema, ele deverá antes configurar o aplicativo. Caso o *software* não esteja configurado ele não conseguirá se conectar ao Sistema Safe House. Na Figura 3.4 é mostrada a tela de configuração do Safe House.

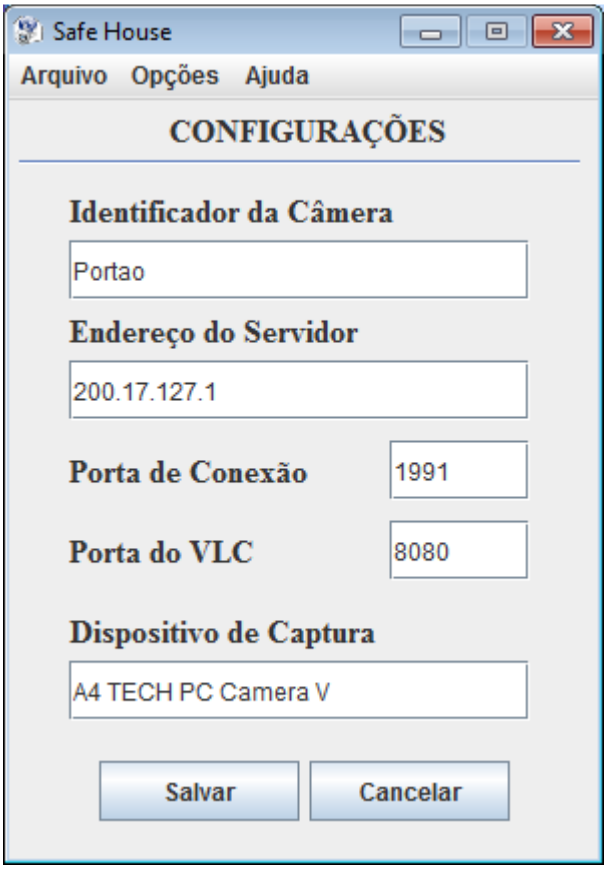

Figura 3.4 – Tela de configuração do aplicativo Safe House

Nesta janela de configurações temos o identificador da câmera, endereço do servidor do sistema, portas utilizadas na comunicação e o dispositivo de captura.

A configuração do aplicativo é bem simples e intuitiva, pois são poucos dados a serem inseridos e os nomes dos campos são bem sugestivos. O identificador da câmera serve para que o usuário possa identificar o local em que a câmera está instalada. Este campo deve ser preenchido de acordo com a configuração feita no site do sistema, pois as câmeras só podem ser adicionadas ou removidas através do *site*. No campo “Endereço do Servidor” temos o endereço do servidor, que se refere ao endereço remoto em que o *software* servidor Safe House Server está executando. O campo “Porta de Conexão” refere-se à porta de conexão com o servidor do sistema. O campo “Porta do VLC” refere-se à porta de comunicação com o aplicativo VLC. E por fim, temos o campo “Dispositivo de

Captura", que se refere ao dispositivo de captura de vídeo, isto é, a *webcam*.

Com todas as devidas configurações feitas, o usuário pode agora inserir seu *login* e sua senha e clicar no botão "Conectar". Ao fazer isso, o aplicativo Safe House inicia o processo de conexão com o aplicativo Safe House Server que, por sua vez, verifica os dados do usuário junto ao banco de dados do sistema. Feita a autenticação dos dados, o aplicativo Safe House Server gera um identificador único para a câmera, salva as devidas informações e envia uma mensagem ao aplicativo Safe House informando que o usuário está conectado.

Após estar conectado ao servidor, o aplicativo Safe House precisa agora iniciar o serviço de captura e armazenamento do vídeo da câmera configurada para ele, assim como também estar pronto para atender a uma requisição de *streaming* de vídeo em tempo real. Então, logo após conectar-se ao servidor, o aplicativo Safe House executa o *software* VLC com todos os parâmetros e configurações necessárias.

Agora com o Safe House executando e conectado ao servidor e, ainda, com todos os serviços funcionando, ele está pronto para atender uma requisição do usuário através do *site* e enviar o *streaming* do vídeo em tempo real.

3.8.2 Software VLC

O *Video LAN Client* (VLC) é um *player* multimídia de código aberto multi-paltaforma, isto é, suportado por vários sistemas operacionais. O VLC inicialmente era o *software* cliente para o projeto VideoLAN que tinha por objetivo criar um programa para transmitir vídeo (*streaming*) por meio de uma rede. Foi criado na *Ecole Centrale* Paris e licenciado na GNU *General Public License* em 1º de fevereiro de 2001, passando a ser desenvolvido por voluntários do mundo todo.

O VLC dá suporte para vários formatos de áudio e vídeo. Podemos citar os formatos de vídeo WMV, MPEG-1, MPEG-2, MPEG-4, DivX, DVD, VCDs como os

mais conhecidos suportados pelo VLC. Também listamos vários formatos de áudio que o VLC dá suporte: OGG, MP3, AAC, WMA, Real áudio, WAV, dentre outros. Além disso, oferece suporte a vários protocolos de *streaming* e transmissão em *unicast* ou *multicast*.

Aproveitando as boas referências do VLC, suas características na transmissão de *streaming* e o fato dele ser um *software* de código aberto, o mesmo foi escolhido para ser incorporado ao Sistema Safe House.

O VLC tem um importante papel no sistema desenvolvido neste trabalho, pois ele quem realiza todo o trabalho de captura e geração do *streaming* de vídeo. O VLC funciona como um serviço para o aplicativo cliente Safe House. O Safe House inicializa o VLC passando todas as configurações necessárias. Feito isso, o VLC inicia o processo de captura e armazenamento do vídeo, assim como também disponibiliza o *stream* do vídeo para o Safe House.

3.8.3 Safe House Server

O Safe House Server é um aplicativo servidor componente do Sistema Safe House que é responsável por gerenciar toda a comunicação do sistema. Ele fica hospedado em um *host* e, como é característico dos *softwares* servidores, deve ficar funcionando ininterruptamente. Outra característica do Safe House Server é que ele não possui interface gráfica, isso porque ele funciona como um serviço não havendo, portanto, a necessidade da mesma. O Safe House Server é um servidor que acumula múltiplas tarefas importantes ao sistema. As principais tarefas deste *software* estão listadas a seguir:

- Fazer o registro das câmeras dos usuários no sistema. Nesta tarefa o Safe House Server gera um identificador único para a câmera do usuário e guarda essa informação no banco de dados no campo destinado, ficando o sistema pronto para realizar o monitoramento em tempo real desta câmera;

- Realizar o serviço de tratamento de requisições de vídeo. Este serviço é feito

quando um usuário solicita, através do *site* do sistema, a visualização em tempo real das imagens de uma câmera. Nesse caso o Safe House Server trata a requisição semelhantemente a um servidor *web* e, se possível, inicia a transmissão do vídeo requisitado;

- Intermediar a transmissão do *stream* de vídeo. Nesta tarefa o Safe House Server funciona como um servidor *proxy*. Quando um vídeo é requisitado pelo usuário, o servidor repassa a requisição para o aplicativo cliente (Safe House) que, por sua vez, trata a requisição e inicia *streaming* do vídeo, sendo o servidor novamente responsável por repassar o *stream* para o usuário.

Ao iniciar, o Safe House Server precisa inicializar alguns serviços para estar pronto. Primeiramente ele inicializa o serviço de comunicação, aguardando futuras conexões de aplicativos clientes. Os aplicativos clientes se comunicam com o servidor através da porta 1991. Em seguida, é inicializado o serviço de “servidor web”. Esse serviço é responsável pelo atendimento das requisições de vídeo que serão feitas pelos usuários através do site do sistema. Este serviço é realizado através da porta 1990, e analogamente a um servidor *web*, que por padrão funciona na porta 80, o Safe House Server fica aguardando as requisições dos usuários via protocolo HTTP. Terminado esse processo de inicialização, o Safe House Server está pronto para realizar todo o trabalho a que foi destinado.

3.8.4 Site Web do Sistema

O Site do Sistema Safe House é a principal interface com o usuário do sistema. É através dele que o usuário pode fazer seu cadastro no sistema, editar seus dados, visualizar suas câmeras e fazer a maioria das configurações de sua conta. Na Figura 3.5 podemos visualizar o site do sistema:

Figura 3.5 – Visualização do site do sistema.

A seguir são listados os principais recursos deste componente do Sistema Safe House:

- Possui um sistema seguro de cadastro de usuários onde as informações são guardadas em um banco de dados e só podem ser acessadas ou modificadas pelas pessoas autorizadas. A Figura 3.6 mostra a página onde o usuário realiza seu cadastro no sistema;

Figura 3.6 – Página de cadastro de usuários do site do sistema.

- Possui um sistema de acesso em que é exigido ao usuário informar seu *login* e senha. Apenas usuários cadastrados podem ter acesso aos recursos do sistema. Os usuários têm acesso apenas aos seus dados e câmeras, não podendo visualizar informações ou câmeras de outros usuários. Na Figura 3.7 é mostrada a página de acesso do sistema;

Figura 3.7 – Página de acesso dos usuários ao sistema.

- Permite ao usuário a visualização de todos os seus dados. Através do *menu* “minha conta”, acessando o item “visualizar” é possível ao usuário ver as informações detalhadas de sua conta. A Figura 3.8 mostra um exemplo desta página;

Figura 3.8 – Página de visualização dos dados do usuário.

- Permite ao usuário a configuração de sua conta. Na página de configuração o usuário pode alterar seus dados cadastrais, adicionar câmeras e, ainda, remover câmeras de sua conta. A Figura 3.9 mostra esta página de configuração;

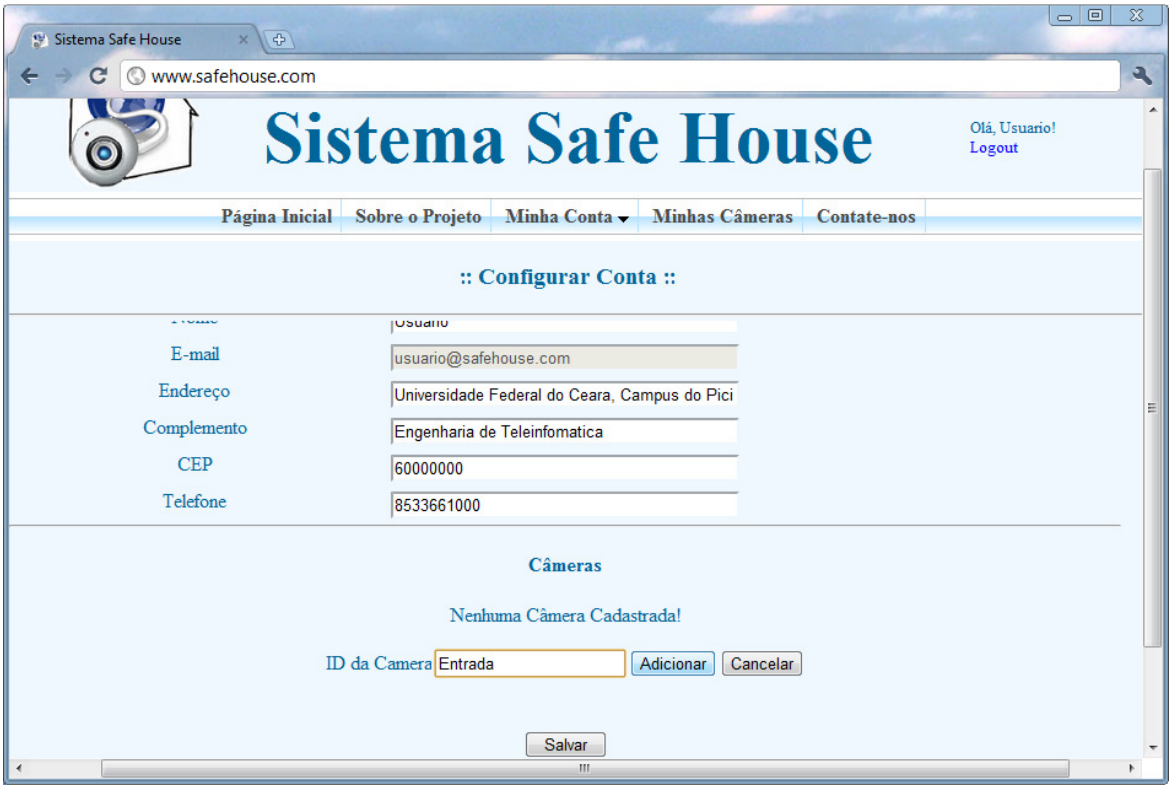

Figura 3.9 – Página de configuração da conta do usuário.

- Possui uma página em que o usuário pode visualizar as imagens de suas câmeras em tempo real. Esta é a parte mais importante do site, pois é nela que o usuário faz o monitoramento de suas câmeras. Esta página tem um *player* de vídeo embutido para cada câmera do usuário. O usuário pode visualizar uma ou mais câmeras por vez e tem, ainda, a opção de visualizar as imagens em tela cheia, isto é, expandir a imagem para toda a tela do computador. Na Figura 3.10 é mostrada a página de monitoramento das câmeras;

Figura 3.10 – Página de monitoramento em tempo real das câmeras do usuário.

O *site* mantém os dados dos usuários no banco de dados do sistema. Isso proporciona segurança às informações e facilita o compartilhamento das mesmas com outros componentes do sistema.

Além de se comunicar com o banco de dados do sistema, o site também se comunica diretamente com o Safe House Server. No momento em que o usuário faz uma requisição de vídeo, através da página de monitoramento das câmeras, o site envia a requisição ao Safe House Server. Este, por sua vez, trata a requisição e inicia a transmissão do *streaming* em tempo real, que será exibido no *player* da página de monitoramento.

3.8.5 Banco de Dados do Sistema

O banco de dados do sistema é o local onde são armazenados todos dados dos usuários do Sistema Safe House. Neste sistema é utilizado o banco de dados MySQL. O MySQL é um sistema de gerenciamento de banco de dados que utiliza a linguagem SQL como interface. Dentre outras razões, ele foi escolhido porque é um *software* livre e atualmente é um dos bancos de dados mais populares do mundo.

A seguir estão listadas as principais características do banco de dados MySQL:

- Portabilidade (suporta praticamente qualquer plataforma atual);
- Compatibilidade (existem *drivers* ODBC, JDBC e .NET e módulos de interface para diversas linguagens de programação, como Delphi, Java, C/C++, C#,Visual Basic, Python, Perl, PHP, ASP e Ruby);
- Excelente desempenho e estabilidade com requerimentos a poucos recursos de *hardware*;
- Facilidade de uso;
- É um *software* Livre com base na GPL;
- Suporta controle transacional;
- Suporta *Triggers*;
- Suporta *Cursors* (*Non-Scrollable* e *Non-Updatable*);
- Suporta *Stored Procedures* e *Functions*;
- Replicação facilmente configurável;
- Interfaces gráficas (MySQL *Toolkit*) de fácil utilização cedidos pela MySQL Inc.

3.9 AVALIAÇÃO DO SISTEMA

Após todas as etapas de projeto e desenvolvimento finalmente temos os resultados deste trabalho, o Sistema Safe House. O Sistema Safe House consegue atender praticamente todos os requisitos que foram propostos no início. Porém apresenta algumas limitações. Podemos citar como um bom resultado do projeto, o sistema de monitoramento através do site que funcionou como desejado. Por outro lado, o monitoramento do vídeo em tempo real apresenta um atraso significativo para aplicações de tempo real, cerca de 30 segundos. Esses atrasos estão relacionados principalmente ao tempo de codificação do *stream* de vídeo, atrasos na rede e à forma de transmissão de vídeo utilizada, que não é a mais adequada para transmissão em tempo real.

No item a seguinte apresentamos a operação do sistema no monitoramento de câmeras de vídeo em tempo real.

3.9.1 Monitoramento em Tempo Real

O monitoramento em tempo real é o principal recurso do Sistema Safe House. Nesta seção mostraremos o funcionamento passo a passo do processo de monitoramento de uma câmera neste sistema.

Primeiramente, o usuário deve estar cadastrado no sistema. Como vimos anteriormente, o site possuí um sistema de cadastramento de usuários. Partindo do pressuposto de que o usuário está devidamente cadastrado e já se conectou ao sistema através da página de acesso, o primeiro passo é adicionar uma câmera em sua conta. Neste exemplo adicionamos uma câmera de nome “Frente da Casa”. Através do *menu* “Minha Conta” acionamos o item “Configurar” para ter acesso à página de configuração que é onde adicionaremos a câmera. A Figura 3.11 mostra esta página de configuração no momento da adição da nova câmera:

Figura 3.11 – Adição de uma nova câmera na página de configuração do site.

Após a adição da câmera com sucesso na conta, o próximo passo é conectar a câmera ao computador e obter o seu identificador. O identificador do dispositivo de captura pode ser obtido na lista de dispositivos do computador, como visto na Figura 3.12.

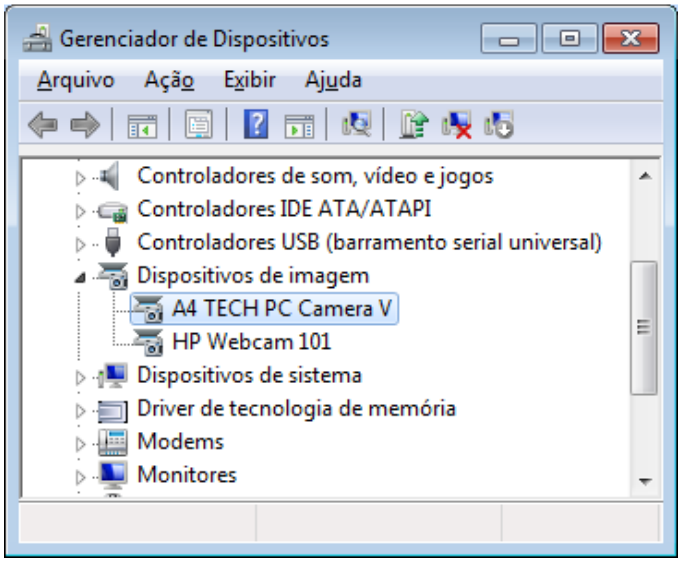

Figura 3.12 – Obtendo o identificador do dispositivo de imagem.

Agora, de posse do identificador do dispositivo de captura, a próxima etapa é executar o Safe House e configurar o aplicativo.

A configuração do Safe House é bem simples. O primeiro campo é destinado ao identificador da câmera e deve ser preenchido com o nome escolhido na primeira etapa. No caso deste exemplo o identificador da câmera é “Frente da Casa”. O segundo campo é destinado ao endereço do servidor. Neste exemplo o aplicativo Safe House Server está sendo executado na mesma máquina em que fazemos o teste. Então o campo foi preenchido com o endereço de *localhost*, isto é, “127.0.0.1”. O próximo campo é a porta de conexão com o servidor. Esta porta por padrão é a 1991. No campo seguinte temos a porta de conexão com o aplicativo VLC. Neste caso estamos utilizando a porta 8080, mas poderia ser qualquer outra, desde que estivesse livre e não for reservada a outra aplicação. No último campo temos agora que preencher com o identificador do dispositivo de captura de imagem. Neste exemplo utilizamos uma *webcam* cujo identificador foi obtido no passo anterior. A configuração final do aplicativo Safe House para este exemplo é mostrada na Figura 3.13:

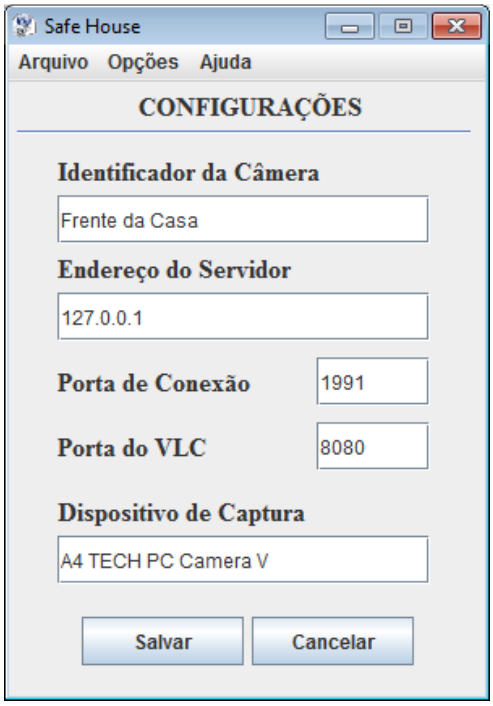

Figura 3.13 – Configuração do aplicativo Safe House.

Preenchido todos os campos corretamente, o próximo passo é salvar as configurações e conectar ao servidor. Para se conectar ao Safe House Server, basta inserirmos usuário e senha, na janela de acesso do aplicativo, e clicar no botão “Conectar”. Feito isso aparecerá uma janela de informações no aplicativo Safe House. Esta janela informa ao usuário o registro dos principais eventos na execução do *software*, como mostrado na Figura 3.14:

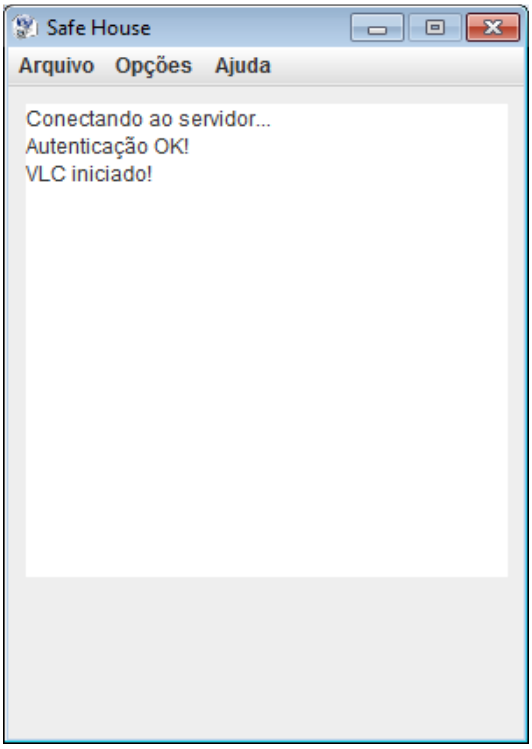

Figura 3.14 – Eventos na execução do Safe House.

Após a conexão com sucesso do Safe House com o servidor, voltamos novamente ao site do sistema para o último passo. Agora, após acessar a sua conta através do site, o usuário acessa a página de monitoramento clicando no *menu* “Minhas Câmeras”. Na página de monitoramento o usuário visualiza os *players* de vídeo destinados a cada uma de suas câmeras. No nosso exemplo adicionamos apenas uma, então o usuário encontrará um *player* de vídeo com o nome da câmera, ou seja, “Frente da Casa”. Finalmente, clicando no botão “play” do *player* de vídeo iniciaremos o processo de monitoramento das imagens da câmera em tempo real. A Figura 3.15 mostra o resultado final do nosso exemplo:

Figura 3.15 – Monitoramento de imagens em tempo real com o Sistema Safe House.

Como vimos no exemplo acima o monitoramento funciona muito bem. Vale ressaltar que a qualidade do vídeo depende não só do Sistema Safe House, mas também da câmera de vídeo que é utilizada. No exemplo acima a imagem não é muito boa justamente por causa da baixa qualidade da câmera utilizada.

4 CONCLUSÕES

Ao final deste trabalho podemos concluir que o Sistema Safe House obteve sucesso na realização de seus principais objetivos. O principal objetivo, a implementação de um sistema de vigilância com câmeras de vídeo que tivesse um baixo custo, foi alcançado. Mostrou-se que com apenas uma câmera *web* e um computador com *internet* banda larga é possível construir um sistema de monitoramento de câmeras em tempo real através da *internet*.

Apesar dos resultados positivos, registramos também que o sistema apresenta algumas limitações, como o atraso na transmissão e exibição do *stream* de vídeo ao vivo. Do ponto de vista da qualidade de serviço para um sistema de tempo real, isso é um aspecto negativo. Porém, considerando as limitações naturais do TCP/IP para o *streaming* multimídia, quando comparamos este atraso com o de outros sistemas semelhantes na *Internet*, concluímos que ele está dentro dos padrões dos *softwares* atuais do mesmo tipo.

O sistema desenvolvido neste trabalho apresentou bons resultados, porém concluímos que é possível melhorá-lo. Melhorias poderiam ser feitas no aplicativo Safe House, incorporando as funções realizadas pelo *software* VLC e tornando-o mais robusto. O transporte do *stream* de vídeo também poderia ser melhorado, alterando os protocolos de transferência para protocolos dedicados ao transporte em tempo real, o que necessitaria de uma reavaliação em sua arquitetura.

Tem-se em perspectiva, portanto, outras melhorias e a incorporação de outros recursos, tornando este sistema mais completo e atrativo. No que se refere aos sistemas de vigilância com câmeras de vídeo pela *internet*, a tendência atual é que esses evoluam e se popularizem cada vez mais.

5 REFERÊNCIAS BIBLIOGRÁFICAS

KUROSE, James F; ROSS, Keith W. **Computer Networking: A Top-Down Approach Featuring the Internet**. 3. ed. EUA: Addison Wesley, 2004

TOPIC, Michael **Streaming Media Demystified**. 1. ed. EUA: McGraw-Hill Professional, 2002

TANENBAUM, Andrew S. **Redes de Computadores**. 4. ed. Campus Elsevier, 2003

SOMMERVILLE, Ian **Engenharia de Software**. 8. ed. Pearson, 2007

DEITEL, Paul J; DEITEL, Harvey M. **AJAX, Rich Internet Applications e Desenvolvimento Web para Programadores**. Pearson, 2008

DEITEL, Paul J; DEITEL, Harvey M. **Java Como Programar**. 3. ed. Porto Alegre: Bookman, 2008

SCHULZRINNE, H. **RTP: A Transport Protocol for Real-Time Applications**. Internet Engineering Task Force (IETF), Julho 2003. (Request for Comments: 3550). url: ftp://ftp.rfc-editor.org/in-notes/rfc3550.txt

SCHULZRINNE, H. **Real Time Streaming Protocol (RTSP)**. Internet Engineering Task Force (IETF), Abril 1998. (Request for Comments: 2326). url: ftp://ftp.rfc-editor.org/in-notes/rfc2326.txt

POSTEL, J. **User Datagram Protocol**. Internet Engineering Task Force (IETF), Agosto 1980. (Request for Comments: 768). url: ftp://ftp.rfc-editor.org/in-notes/rfc768.txt

FIELDING, R. **Hypertext Transfer Protocol – HTTP/1.1**. Internet Engineering Task Force (IETF), Julho 1999. (Request for Comments: 2616). url: ftp://ftp.rfc-editor.org/in-notes/rfc2616.txt

DARPA. **Transmission Control Protocol**. Internet Engineering Task Force (IETF), Setembro 1981. (Request for Comments: 793). url: ftp://ftp.rfc-editor.org/in-notes/rfc793.txt

DARPA. **Internet Protocol**. Internet Engineering Task Force (IETF), Setembro 1981. (Request for Comments: 791). url: ftp://ftp.rfc-editor.org/in-notes/rfc791.txt

PISA, Pedro Silveira **Vídeo Par a Par**. [online]. Disponível na internet via www. url: http://www.gta.ufrj.br/grad/08_1/video-p2p/index.html. Arquivo capturado em 16 de janeiro de 2011

LONGTAIL. **Streaming Video with the JW Player**. [online]. Disponível na internet via www. url: http://www.longtailvideo.com/support/jw-player/46/streaming-video-the-jw-player. Arquivo capturado em 10 de novembro de 2010

ADOBE. **Cross-domain policy for Flash movies**. [online]. Disponível na internet via www. url: http://kb2.adobe.com/cps/142/tn_14213.html. Arquivo capturado em 11 de novembro de 2010

VIDEOLAN. **VideoLAN Streaming Howto**. [online]. Disponível na internet via www. url: http://www.videolan.org/doc/streaming-howto/en/. Arquivo capturado em 14 de outubro de 2010