UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

VISUALIZAÇÃO DE IMAGENS CAPTURADAS EM UM CIRCUITO FECHADO DE TELEVISÃO (CFTV) NO IPHONE

DIOGO CARLASSARA

BLUMENAU
2009

2009/2-05

DIOGO CARLASSARA

VISUALIZAÇÃO DE IMAGENS CAPTURADAS EM UM CIRCUITO FECHADO DE TELEVISÃO (CFTV) NO IPHONE

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

Prof. Dalton Solano dos Reis, M.Sc – Orientador

BLUMENAU
2009

2009/2-05

VISUALIZAÇÃO DE IMAGENS CAPTURADAS EM UM CIRCUITO FECHADO DE TELEVISÃO (CFTV) NO IPHONE

Por

DIOGO CARLASSARA

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

Presidente:	Prof. Dalton Solano dos Reis, M.Sc. – Orientador, FURB
Membro:	Prof. Paulo César Rodacki Gomes, Doutor – FURB
Membro:	Prof. Mauro Marcelo Mattos, Doutor – FURB

Blumenau, 15 de dezembro de 2009

Dedico este trabalho a todos os amigos e familiares, especialmente aqueles que estiveram comigo na realização deste, incentivando e apoiando.

AGRADECIMENTOS

A Deus, pelo seu imenso amor e graça.

À minha família, que esteve sempre presente, me apoiando e incentivando.

Aos meus amigos, pelos empurrões e cobranças.

Ao meu orientador, Dalton Solano dos Reis, por ter contribuído e acreditado na conclusão deste trabalho.

A empresa, Benner Sistemas S.A., por permitir o horário flexível quando fora necessário.

Todo o bem que eu puder fazer, toda a ternura que eu puder demonstrar a qualquer ser humano, que eu os faça agora, que não os adie ou esqueça, pois não passarei duas vezes pelo mesmo caminho.

James Greene

RESUMO

Este trabalho apresenta o desenvolvimento de uma ferramenta para visualização de vídeos gerados por *webcams* diretamente no iPhone, a fim de permitir mobilidade no acompanhamento de sistemas de Circuitos Fechados de TeleVisão (CFTV). O trabalho é dividido em duas fases, o desenvolvimento de uma aplicação servidora, chamada Gerador e uma cliente, chamada Cliente. O Gerador deve ser instalado no computador *desktop*, pessoal ou notebook, possuindo uma câmera instalada, que fará a captura e gravação dos vídeos. Os vídeos são gravados em um diretório local na máquina onde o Gerador está instalado, utilizando-se uma resolução de 300 *pixels* de altura e 240 *pixels* de largura, com uma profundidade de cores de 32 *bits*. O Gerador possui, além de recurso de iniciar e parar uma gravação de vídeo, a possibilidade de fotografar imagens em tempo real. O Cliente, que por sua vez é instalado no iPhone, permite receber e visualizar os vídeos e imagens gerados pelo Gerador. Esta aplicação permite ainda configurar vários locais de monitoramento e a recepção necessita de uma conexão com a internet. Ambas, aplicação Gerador e Cliente foram desenvolvidas em sistema operacional Mac OS X versão 10.5.8, utilizando-se ambiente XCode (versão 3.1.3), para codificação e Interface Builder (versão 3.1.2) para desenho das telas.

Palavras-chave: iPhone. Dispositivo móvel. Circuito fechado de televisão. Imagens digitais.

ABSTRACT

This paper presents the development of a tool for viewing video cameras directly generated by the iPhone, to enable mobile monitoring systems Closed Circuit Television (CCTV). The work is divided into two phases, the development of an application server and a client. The server must be installed on your computer desktop, notebook or personal, having a camera installed, which will capture and record videos. The videos are recorded in a local directory on the machine where the server is installed, using a resolution of 300 pixels high and 240 pixels wide, with a color depth of 32 bits. The server has, and feature start and stop a video recording, the ability to shoot images in real time. The Client, which in turn is installed on the iPhone, to receive and view the videos and images generated by the server. This application can also configure multiple monitoring locations received independent and to have access to a WiFi network and / or 3G. Both application server and client were developed in Mac OS X version 10.5.8, using XCode environment (version 3.1.3), for encoding and Interface Builder (version 3.1.2) to design the screens.

Key-words: iPhone. Mobile device. Closed circuit television. Digital images.

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

3G – 3 Geração (Rede de telefonia móvel de terceira geração)

ACID – Atomicidade, Consistência, Isolamento e Durabilidade

API – *Application Programming Interface*

CCD – *Charged Coupled Device*

CCTV – *Closed Circuit TeleVision*

CDMA – *Code Division Multiple Access*

CFTV – Circuitos Fechados de TeleVisão

CPU - *Central Processing Unit*

DHCP – *Dynamic Host Configuration Protocol*

EDGE - *Enhanced Data Rates for GSM Evolution*

FDMA – *Frequency Division Multiple Access*

GHz – *GigaHertz*

GPRS – *General Packet Radio Service*

GPS – *Global Positioning System*

GSM – *Global System for Mobile Communications*

HSDPA – *High-Speed Downlink Packet Access*

IEEE – Instituto de Engenheiros Eletricistas e Eletrônicos

IMT – *International Mobile Telecommunication*

KBPS – *KiloBitt Per Second*

PPC – *PowerPC*

RAM - *Random Access Memory*

SGBD – Sistema Gerenciador de Banco de Dados

SP - *Service Pack*

SQL – *Structured Query Language*

SQL92 – *Search Query Language* 92

TCL – *Tool Command Language*

TDMA – *Time Division Multiple Access*

UIT – União Internacional de Telecomunicações

UMTS – *Universal Mobile Telecommunications System*

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

WLAN – *Wireless Local Area Network*

SUMÁRIO

1 INTRODUÇÃO

Com o aumento da constante necessidade de segurança, seja em nossa sociedade, seja em nossa casa ou trabalho, uma das alternativas que se pode encontrar é a implantação de sistemas de Circuitos Fechados de TeleVisão (CFTV). Este tipo de sistema tem por finalidade, por exemplo, monitorar as atividades de um funcionário em um estabelecimento comercial, monitorar um caixa de banco, ou ainda monitorar um bebê que está dormindo no quarto ao lado de seus pais.

> O Sistema de Circuito Fechado de TV – CFTV, tem como objetivo possibilitar o monitoramento de vários locais em um único ponto, centralizando o gerenciamento e facilitando a tomada de decisões. Atualmente os Sistemas de CFTV são utilizados nos mais diversos locais, sempre tendo como função principal o auxílio ao gerenciamento (Controle). (PROCURADORIA REGIONAL DO TRABALHO 19ª REGIÃO, 2006, p. 4).

Atualmente, para o funcionamento de um CFTV, se faz necessária a aquisição de equipamentos como um computador ou televisão, que receberão as imagens geradas por *webcams*. Um problema quanto a isso seria a visualização, a qual fica restrita apenas a um destes equipamentos, sem nenhuma mobilidade.

Uma alternativa para visualização poderia ser através de um navegador, tal como Firefox ou Internet Explorer, via internet. O navegador pode trazer um pouco de flexibilidade, pois permite a verificação das imagens a partir de qualquer computador ou *notebook*, entretanto não possui a mobilidade que um dispositivo móvel pode trazer, uma vez que as consultas podem ser feitas em qualquer hora e local, independente de um computador.

O desafio proposto é possibilitar a visualização das imagens geradas por *webcams* em um único ponto central, sendo este ponto um dispositivo móvel, o iPhone. A transmissão das imagens para o iPhone será feita através da internet, utilizando arquivos de vídeo, com a finalidade de ser possível a consulta em qualquer hora ou lugar.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é exibir as imagens e vídeos previamente gravados em uma máquina remota no iPhone.

Os objetivos específicos do trabalho são:

a) capturar imagens através de *webcams*;
b) utilizar o acesso a internet para receber as imagens e vídeos.

1.2 ESTRUTURA DO TRABALHO

O presente trabalho está estruturado em quatro capítulos. Neste contexto, o segundo capítulo apresenta a fundamentação teórica necessária para o seu desenvolvimento. Nele são expostos detalhes sobre CFTV, sobre iPhone e suas particularidades, sobre a linguagem de programação utilizada para o desenvolvimento do trabalho, os meios de transmissão de dados sem fio, algumas aplicações disponíveis no mercado e também as ferramentas utilizadas no desenvolvimento. O capítulo também mostra características de alguns trabalhos correlatos.

No terceiro capítulo é apresentado o desenvolvimento das aplicações Gerador e Cliente, onde são apresentados os requisitos e a especificação das aplicações desenvolvidas. Esta especificação compreende os diagramas de casos de uso, de classes e de seqüência. O terceiro capítulo também demonstra a operacionalidade da aplicação e aborda aspectos relacionados à sua implementação, bem como os resultados obtidos.

Finalizando, no quarto capítulo são apresentadas as conclusões e a lista de tarefas para continuidade do estudo sobre o desenvolvimento para iPhone.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são abordados os assuntos e técnicas utilizadas para o desenvolvimento da ferramenta de visualização de imagens capturadas em um circuito fechado de televisão. Na seção 2.1 são apresentadas informações sobre o CFTV e sua estrutura básica de funcionamento. Em seguida, na seção 2.2 é apresentada a plataforma iPhone, suas particularidades, a linguagem de programação utilizada, as ferramentas de desenvolvimento e banco de dados utilizado. Após, na seção 2.3 são apresentados os meios de transmissão que podem ser utilizados para transferência dos dados capturados para o iPhone. Por fim, na seção 2.4 são descritos os trabalhos correlatos.

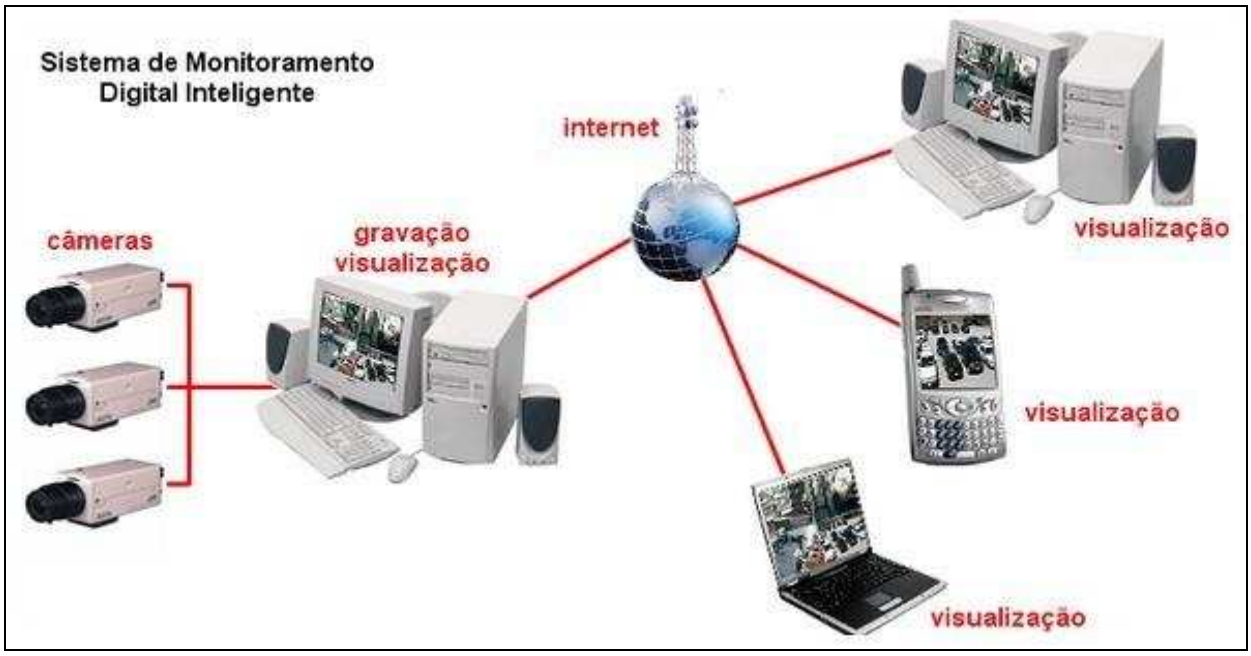
2.1 CFTV

Um CFTV é um sistema de televisão que distribui sinais oriundos de câmeras de vídeo localizadas em pontos específicos, para um ou mais pontos de visualização.

> Circuito Fechado de Televisão, (do termo inglês Closed Circuit TeleVision - CCTV), é um sistema de televisionamento que distribui sinais provenientes de câmeras localizadas em um local especifico, para um ponto de supervisão pré-determinado. Os sistemas de CFTV normalmente utilizam câmeras de vídeo CCD (para produzir o sinal de vídeo), cabos ou transmissores/receptores sem-fio ou redes (para transmitir o sinal), e monitores (para visualizar a imagem de vídeo captada). (Moraes, 2006, p. 6).

Para que um sistema de CFTV funcione completamente, os seguintes componentes são necessários: iluminação, lentes, câmera e seus componentes, tais como suporte, cabeamento ou transmissor sem fio e um dispositivo de carga acoplada (*Charged Coupled Device* - CCD), que é, segundo Oliveira Filho (1999) uma câmera com “[...] lente convergente, ou seja, que direciona os raios de luz em direção uns aos outros [...]”, processadores, monitores, alimentação e opcionalmente gravadores.

A estrutura básica para o funcionamento do sistema de CFTV é ilustrados na Figura 1.



Fonte: Horus (2009).

Figura 1 - Equipamentos necessários para o funcionamento do sistema de CFTV

A Figura 1 demonstra a conexão entre os equipamentos necessários para a criação de um sistema de CFTV. As câmeras são conectadas ao dispositivo de visualização e gravação, a fim de gravar as imagens capturadas em um disco rígido. Este é conectado, via internet, a outros equipamentos de visualização, tais como computadores, *notebooks* e dispositivos móveis.

2.2 IPHONE

O iPhone pertence ao grupo dos aparelhos chamados *smartphones* (telefones inteligentes, tradução nossa), que são aparelhos de telefonia móvel com funções avançadas, nas quais podem ser estendidas por meio de programas em seu sistema operacional. O iPhone é fabricado pela empresa Apple Inc., e entre os recursos pode-se citar: câmera digital, internet e rede sem fio, sistema de posicionamento global (*Global Positioning System* - GPS) e tela *touch-screen* (tela sensível ao toque, tradução nossa).

Uma particularidade do iPhone é a ausência de teclado. Quando o mesmo é necessário, um teclado virtual aparece na tela, permitindo a digitação através do toque dos dedos. Foi o primeiro telefone celular a implementar a tecnologia de *multi-touch* (múltiplos toques, tradução nossa), onde se pode tocar com dois dedos na tela, para se ter funcionalidades como

zoom de imagens, por exemplo. Outros *smartphones* também possuem a tecnologia *touch-screen*, mas o toque na tela deve ser feito com um dedo apenas. A Figura 2 mostra a interface atual do iPhone.

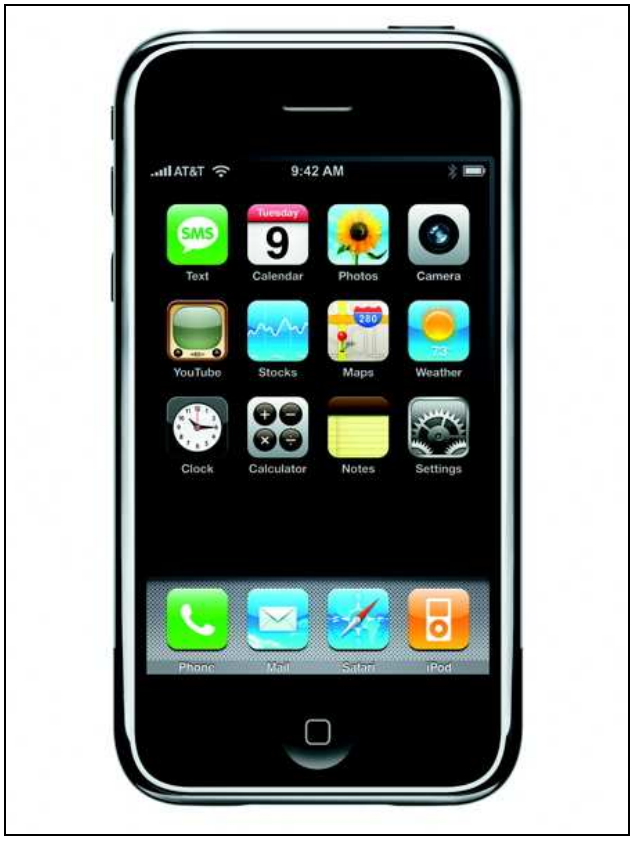


Figura 2 - Foto de um iPhone

2.2.1 LINGUAGEM DE PROGRAMAÇÃO

O desenvolvimento de aplicações para o iPhone é feito no sistema operacional Mac OS X, utilizando a linguagem Objective-C, que, segundo Webclaudio (2009), "[...] é a linguagem utilizada pela Apple para programação de seus sistemas [...]".

O Objective-C é uma linguagem que foi desenvolvida tendo como base outras duas linguagens, o C e o Smalltalk. Da linguagem C, o Objective-C trouxe a velocidade, e, do Smalltalk, a capacidade de ser programada com Orientação a Objetos (OO).

> A linguagem Objective-C foi criada por Brad Cox e sua empresa, a StepStone Corporation, no início da década de 80. Em 88 ela foi licenciada pela NeXT, tornando-se a linguagem de desenvolvimento do NeXTstep. O suporte do GNU/gcc foi acrescentado em 1992. Em 1994 as empresas NeXT Computer e Sun Microsystems criaram uma especificação do NeXTstep conhecida como OpenStep. A implementação da Free Software Foundation da especificação OpenStep é denominada GNUStep. (Martins, 2005).

Para a programação, é utilizado o *framework*[1] Cocoa, que é uma das cinco maiores APIs[2] disponíveis para o Mac OS X.

2.2.2 SQLITE

Segundo Devmedia (2009), "[...] SQLite é uma ferramenta que permite com que desenvolvedores possam armazenar os dados de suas aplicações em tabelas e manipular esses dados através de comandos SQL. A diferença é que tudo isso pode ser feito sem que seja preciso acessar um SGBD".

O SQLite foi desenvolvido em linguagem de programação C, e pode ser integrado com aplicações e ferramentas escritas em diferentes linguagens, com o intuito de possibilitar a manipulação de dados através de instruções *Structured Query Language* (SQL).

Segundo Sqlitebr (2009), algumas das características do SQLite pode ser:

a) transações são Atômicas, Consistentes, Isoladas e Duráveis (ACID) mesmo que o sistema trave ou a energia falhe;
b) configuração-zero - nenhuma instalação ou administração necessária;
c) implementação da maior parte do SQL92;
d) um banco de dados completo é armazenado em apenas um arquivo de sistema;
e) arquivos de banco de dados podem ser livremente compartilhados entre máquinas com diferentes ordens de byte;
f) suporta bases de dados de até 2 terabytes de tamanho;
g) tamanho de strings e BLOBs limitados apenas pela memória disponível;
h) mais rápido que populares bancos de dados cliente/servidor para a maioria das operações comuns;
i) API simples e fácil de usar;
j) TCL *bindings*[3] inclusas. *Bindings* para a maioria das linguagens disponíveis separadamente;

---

[1] *Framework*: termo para designar uma aplicação, um conjunto de aplicações, ou classes que servem de suporte ao desenvolvimento de software num determinado contexto.

[2] *Application Programming Interface* (API): conjunto de rotinas e padrões estabelecidos por um software para a utilização das suas funcionalidades por programas aplicativos. A API é composta por uma série de funções acessíveis somente por programação.

[3] *Bindings:* ligação entre o protocolo de comunicação e o adaptador de rede.

k) código fonte bem comentado, com mais de 95% coberto por testes;
l) auto-contido: sem dependências externas;
m) fontes estão em domínio público, podendo ser usado para qualquer propósito.

2.3 MEIOS DE TRANSMISSÃO DE DADOS

Segundo Scrócaro e Saturino (2005), “O meio de transmissão de dados serve para oferecer suporte ao fluxo de dados entre dois pontos. Computadores em rede ficam interligados por meio de fios elétricos, fibras ópticas, ondas de rádio ou raios de luz e nas redes com fio, pode-se utilizar o par trançado ou cabo coaxial”.

A seguir, na seção 2.3.1, é descrito as redes sem fio e na seção 2.3.2 as redes 3G, sendo estas as principais formas de transmissão de dados em dispositivos móveis.

2.3.1 REDES SEM FIO WIFI

Uma rede sem fio usa ondas de rádio, do mesmo modo que os telefones celulares, televisões e rádio. A comunicação ao longo da rede sem fio é muito parecida com a comunicação de rádio emissor-receptor (COMO TUDO FUNCIONA, 2009).

Segundo Péricas (2003, p. 53), “As especificações IEEE 802.11b e IEEE 802.11g, que podem transmitir por radiofreqüência ou infravermelho, oferecem transmissão sem fio sobre distâncias relativamente curtas, operando na freqüência de 2,4 GHz. A taxa de transmissão deste tipo de rede é de 11Mbps (IEEE 802.11b) ou 54Mbps (IEEE 802.11g)”.

Péricas (2003, p. 53) ainda diz que “O protocolo utilizado pelo IEEE 802.11 é o Ethernet e o método básico de acesso é o do tipo de coordenação distribuída, onde a decisão de transmitir é tomada individualmente pelas estações [...]”.

Resumidamente, a comunicação se dá em dois passos. Primeiro, o adaptador da rede sem fio (hardware) traduz os dados na forma de um sinal de rádio e os transmite usando uma antena. Então, o roteador sem fio recebe o sinal e o decodifica, enviando a informação para a internet usando uma conexão física, com fios.

2.3.2 REDES 3G

Segundo 3G, redes 3G dizem respeito “à terceira geração de padrões e tecnologias para a telefonia móvel”. Os padrões são baseados nas famílias de normas da União Internacional de Telecomunicações (UIT) e no programa Internacional de Telecomunicações Móveis (IMT).

Com a evolução das redes sem fio, mais serviços puderam ser oferecidos. Segundo Kataoka (2008) “[...] entre os serviços 3G estão o tráfego de voz, dados, vídeo, incluindo vídeo sob demanda e jogos multiplayer”.

A evolução das redes sem fio para telefonia móvel segundo Kataoka (2008) é:

a) 1ª geração: transmissão de voz analógica através de Acesso Múltiplo por Divisão de Frequência (*Frequency Division Multiple Access* – FDMA), que é uma técnica usada para permitir que mais de uma estação terrestre compartilhe a largura de banda de um repetidor de satélite;
b) 2ª geração: transmissão de voz digital através de Acesso Múltiplo por Divisão de Tempo (*Time Division Multiple Access* – TDMA), Sistema Global para Comunicação Móvel (*Global System for Mobile Communications* – GSM), Serviço de Rádio de Pacote Geral (*General Packet Radio Service* – GPRS) – que é uma evolução do GSM, *Enhanced Data Rates for GSM Evolution* (EDGE) – que é uma evolução do GPRS e Acesso Múltiplo por Divisão de Código (*Code Division Multiple Access* – CDMA);
c) 3ª geração: transmissão de voz e dados digitais em alta velocidade, utilizando Acesso Múltiplo por Divisão de Código de Banda Larga (*Wide-band Code-Division Multiple Access* – WCDMA) e a tecnologia CDMA evoluída, o CDMA2000.

2.4 APLICAÇÕES CCTV NO IPHONE

A seguir serão descritas algumas aplicações que estão disponíveis na App Store, que nada mais é de que um serviço disponibilizado pela Apple, no qual permite que usuários cadastrados naveguem e baixem softwares para iPhone.

2.4.1 YOICS

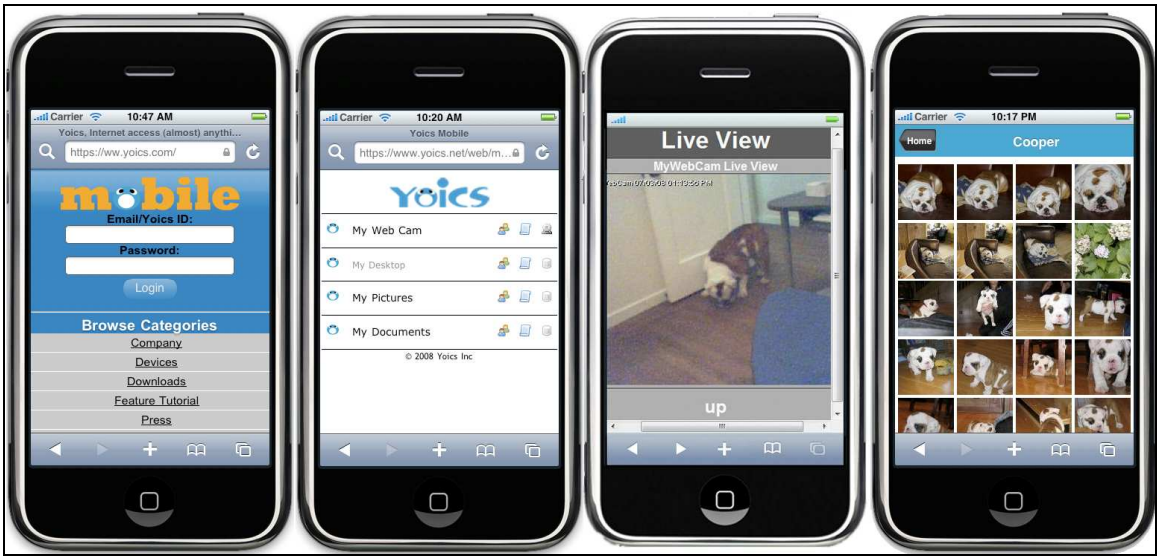
Yoics é uma ferramenta que oferece uma alternativa para acessar remotamente computadores pessoais, arquivos do computador, *webcams*, câmeras *Internet Protocol* (IP), dispositivos de armazenamento ou qualquer outro produto ou serviço acessível à rede.

O acesso aos dispositivos é realizado através do próprio navegador disponível no iPhone, uma vez que a ferramenta é chamada *webapp* (aplicação web, tradução nossa), ou seja, uma aplicação que é acessada via navegador, através da internet ou intranet.

A ferramenta roda sobre os sistemas operacionais Windows, Linux e Mac OS X, podendo transformar qualquer *webcam* em um sistema de vigilância, capturando e gravando tudo no disco rígido do computador onde está instalado. A visualização pode ser feita através do iPhone, mas pode ser feita a partir de qualquer navegador.

A ferramenta pode ser usada como uma aplicação de mensageiro instantâneo, mas ao invés de uma lista de contatos, ele exibe uma lista de dispositivos compartilhados por outras pessoas. Entre os compartilhamentos disponíveis, pode-se citar o compartilhamento de pastas, arquivos, serviços ou dispositivos externos, que podem ser acessados remotamente.

Um exemplo de funcionamento do Yoics no iPhone é demonstrado na Figura 3. Na primeira imagem, é demonstrada a tela inicial de *login*, na qual o usuário deverá inserir o nome de usuário e senha de acesso. A segunda imagem demonstra a tela de configurações disponíveis, entre elas estão destacam-se a biblioteca de imagens, de documentos e câmeras disponíveis. Na terceira imagem é demonstrado o monitoramento em tempo real de uma das câmeras cadastradas. Na quarta imagem, é demonstrada a biblioteca de imagens, com as fotos tiradas.



Fonte: Yoics (2009).

Figura 3 - Funcionamento do Yoics

Segundo Yoics (2009), para o seu funcionamento básico, alguns requisitos de hardware e software são exigidos, para o sistema operacional Windows (serviços adicionais do Yoics podem solicitar outros requisitos específicos):

a) sistema operacional Windows 2000 Service Pack (SP) 4, Windows XP SP 2 ou SP 3, Vista SP 1;
b) *framework* .NET 2.0 para Windows;
c) Unidade Central de Processamento (CPU) de 1 Gigahertz;
d) 1 Gigabyte de Memória de Acesso Randômico (*Random Access Memory* - RAM);
e) 100 Megabytes de espaço no disco rígido;
f) internet de banda larga, com velocidade igual ou superior a 384 Kbps.

Para o sistema operacional Mac OS X, os requisitos são os seguintes:

a) PPC G4, G5 ou processador Intel Core Duo;
b) Mac OS X version 10.4.5 ou superior;
c) 512 Gigabyte de Memória de Acesso Randômico (*Random Access Memory* - RAM);
d) 80 Megabytes de espaço no disco rígido;
e) internet de banda larga, com velocidade igual ou superior a 384 Kbps.

2.4.2 CAVU FREE VÍDEO SURVEILLANCE

Cavu (2009), é uma aplicação para iPhone que possibilita a visualização de câmeras de vigilância em tempo real, a partir de qualquer lugar. A ferramenta necessita conexão com internet, seja ela com fio, sem fio ou 3G.

Uma vez conectado à internet, o software disponibiliza todas as câmeras cadastradas em uma matriz onde as imagens podem ser visualizadas com o iPhone em posição vertical ou horizontal, e ainda podem ser organizadas em grupos.

Na versão gratuita, Cavu permite visualizar em tempo real, apenas as câmeras locais. A versão paga permite que qualquer câmera IP seja visualizada, e possui outras funcionalidades, tais como suporte à conexão com vários servidores, zoom, entre outros. A Figura 4 demonstra a tela de visualização das câmeras do Cavu.

Fonte: Cavu (2009).

Figura 4 - Tela de visualização das câmeras do Cavu

2.4.3 NEXTVIEW REMOTE VIDEO CAMERA SURVEILLANCE

Nextview (2009) é uma aplicação de vigilância para iPhone que utiliza conexão 3G ou sem fio. Para cada conta de usuário criada podem ser visualizadas até 16 câmeras em tempo real.

Uma particularidade dessa aplicação é a possibilidade de integração com o sistema de segurança doméstico (sistema de alarmes), acrescentando assim, a possibilidade de notificação com imagens de vídeo sempre que uma porta ou janela da casa for aberta ou fechada, através dos sensores instalados na residência. As câmeras são instaladas na rede utilizando o protocolo de configuração dinâmica de endereços de rede, também conhecido como *Dynamic Host Configuration Protocol* (DHCP).

A aplicação possui versões gratuitas e pagas, com funcionalidades diferentes. Na versão paga, por exemplo, as câmeras podem ser controladas remotamente (para cima, baixo, esquerda ou direita).

Um ponto fraco em relação à essa aplicação é que as câmeras utilizadas para a visualização devem ser as próprias disponibilizadas pela empresa que desenvolve o software, sendo elas *NextView Standart* ou *NextView Pro Wifi*. A Figura 5 demonstra a tela da aplicação Nextview.

Fonte: Nextview (2009).

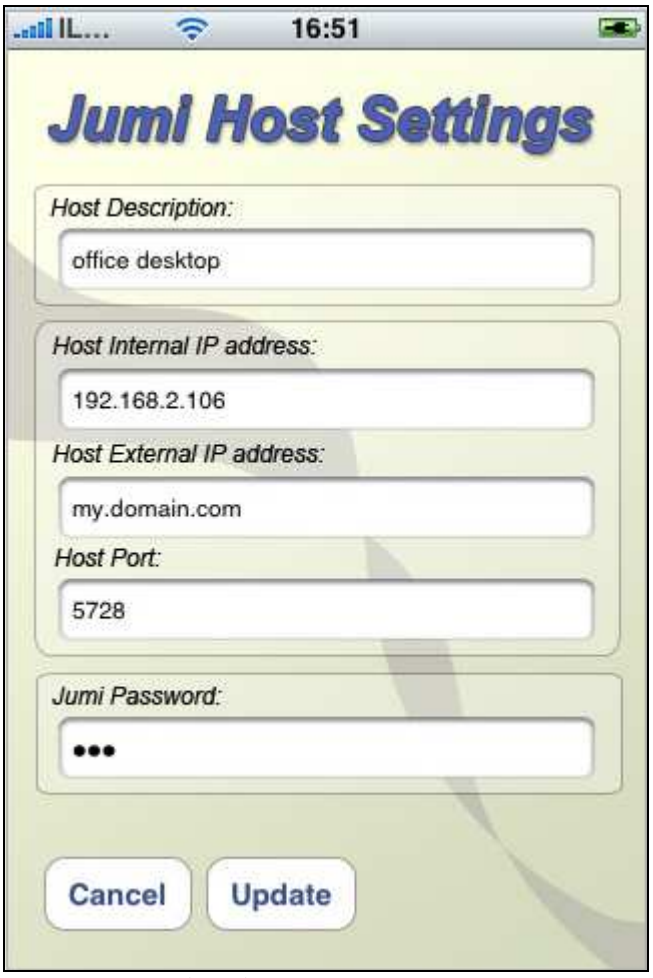
Figura 5 - Tela da aplicação Nextview

2.4.4 JUMICAM

Jumicam (2009) é uma aplicação para iPhone e/ou iPod que permite que sejam visualizadas webcams de qualquer lugar. Funciona em sistema operacional Windows XP ou Vista 32 bits, utilizando conexão com internet 3G ou sem fio.

Para funcionar, a aplicação cria o *streaming* a partir de um computador com webcam.

O software requer que sejam utilizados dois programas, o Jumi Controller no computador e o JumiCam no iPhone e/ou iPod. O Jumi Controller é que fará a geração do *streaming* e o disponibilizará para o JumiCam. O JumiCam por sua vez, receberá o *streaming* gerado e mostrará na tela do iPhone e/ou iPod. A Figura 6 demonstra a tela de configuração do servidor.



Fonte: Jumicam (2009).

Figura 6 - Tela de configuração do servidor

A Figura 7 demonstra a tela de visualização do JumiCam.

Fonte:Jumicam (2009).

Figura 7 - Tela de visualização do JumiCam

2.4.5 CAMCONTROL FOR IPHONE

O CamControl for iPhone (CAMCONTROL, 2009) é uma aplicação de vigilância desenvolvida para iPhone e/ou iPod que permite a visualização de câmeras em tempo real.

A transmissão das imagens se dá em torno de 12 quadros por segundo e se pode ter controle sobre as câmeras, tal como zoom. Possui várias distribuições (versões gratuitas e pagas, com funcionalidades diferentes para cada versão).

A comunicação com iPhone pode se dar em vários protocolos: *Universal Mobile Telecommunications System* (UMTS) em 3G, *High-Speed Downlink Packet Access* (HSDPA), *Wide-Band Code-Division Multiple Access* (W-CDMA), *Wireless Local Area Network* (WLAN). Com o iPod é disponibilizada apenas a conexão do tipo WLAN. A Figura 8 demonstra a tela de visualização do CamControl.

Fonte: Camcontrol (2009).

Figura 8 - Tela de visualização do CamControl

2.4.6 RESUMO E CARACTERÍSTICAS

Abaixo (Quadro 1) é demonstrado, resumidamente, um comparativo entre os principais softwares de vigilância disponibilizados na ITUNES STORE (2009).

Software	Yoics	CAVU	NextView	JumiCam	CamControl
Transmissão em tempo real	✔	✔ *	✔ **	✔ ***	✔
Funciona com rede sem fio	✔	✔	✔	✔	✔
Possui versão gratuita	✔	✔	✔	✔	
Visualização de câmeras públicas	✔	✔	✔	✔	✔
Zoom		✔	✔ **		✔
Câmera pode ser controlada remotamente			✔ **		✔
Visualizada no navegador	✔				

* Apenas na versão paga
** Devem ser adquiridas as câmeras do próprio fabricante
*** Apenas para câmeras USB conectadas no computador

Quadro 1- Comparativo entre os principais softwares de vigilância

2.4.7 INTERFACE BUILDER

Interface Builder (2009) é uma ferramenta para desenho e teste de interfaces na plataforma Mac OS X. Esta ferramenta permite aos desenvolvedores criar as interfaces apenas selecionando e arrastando componentes para dentro das mesmas. Possui várias configurações, dependendo do tipo de uso.

Entre estas configurações, destacam-se o desenvolvimento para iPhone e computador (*desktop*), chamada Cocoa. A Figura 9 demonstra os tipos de configurações disponíveis para desenvolvimento para iPhone e a Figura 10 demonstra as configurações disponíveis para desktop.

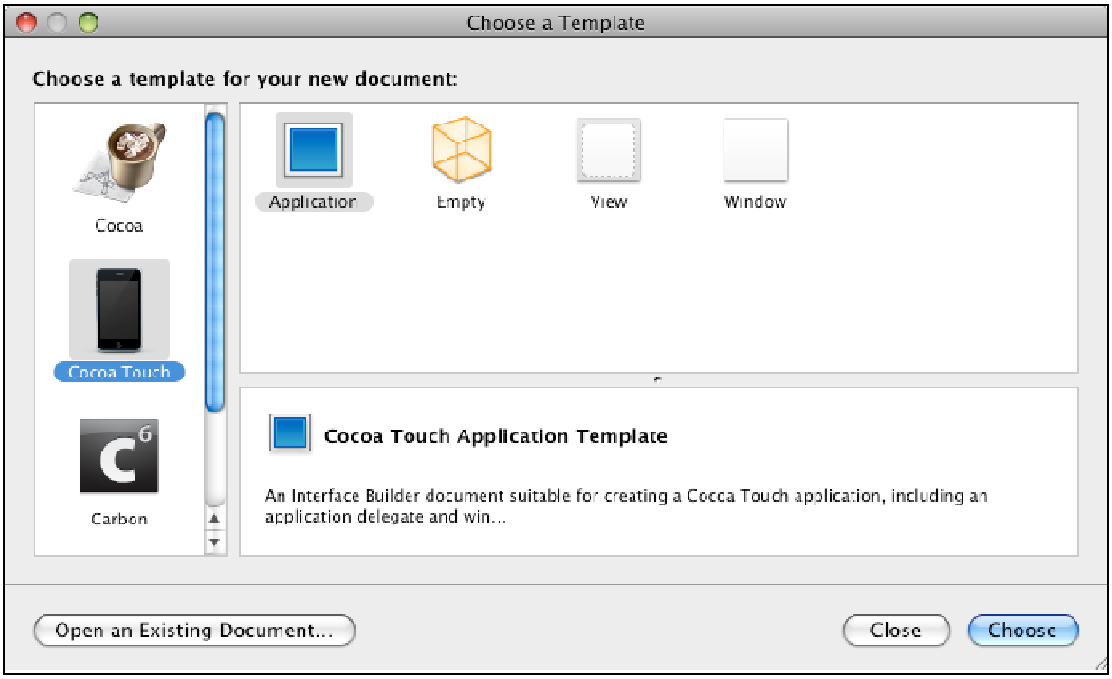


Figura 9 - Configurações disponíveis no desenvolvimento para iPhone

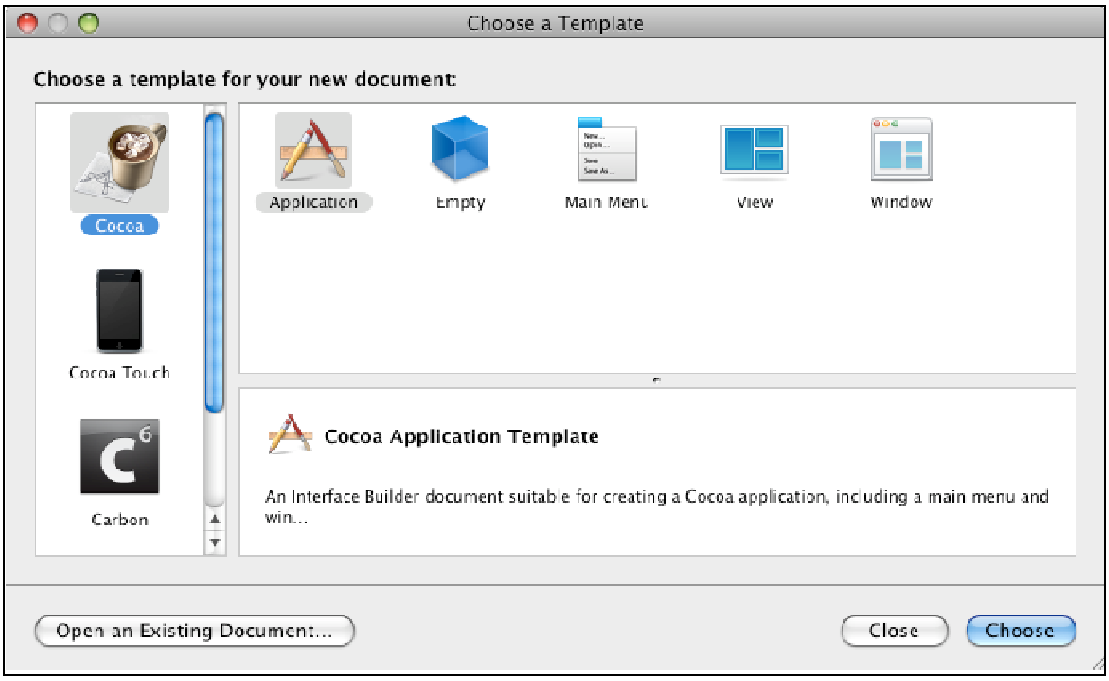


Figura 10 - Configurações disponíveis para *desktop*

O Interface Builder trabalha com arquivos cuja extensão é ".nib", conhecida como *Nextstep Interface Builder*. Estes arquivos são representações estáticas de objetos de interfaces e suas ligações (tal como a ação de clicar um botão). O Interface Builder ainda possui recursos de testes de interface e integração com o ambiente XCode, utilizado para a codificação.

2.4.8 XCODE

XCode (2009) é um conjunto de ferramenta integradas para o desenvolvimento na plataforma Mac OS X. Entre estas ferramentas, incluem-se compiladores e aplicativos, juntamente com um extenso conjunto de bibliotecas e interfaces. O ponto central desta ferramenta é a própria aplicação XCode, que oferece uma interface organizada para gerenciar projetos de desenvolvimento de software.

O XCode suporta códigos escritos em Objective-C, C, C++, por exemplo. No caso particular da linguagem Objective-C, os códigos são divididos em arquivos com extensão .h (arquivo de cabeçalho, na qual são feitas as declarações de funções) e .m (arquivo com a implementação dos métodos descritos no arquivo .h)

A Figura 12 demonstra a tela inicial e a Figura 12 demonstra o ambiente de desenvolvimento do XCode.

Figura 11 - Tela inicial do XCode

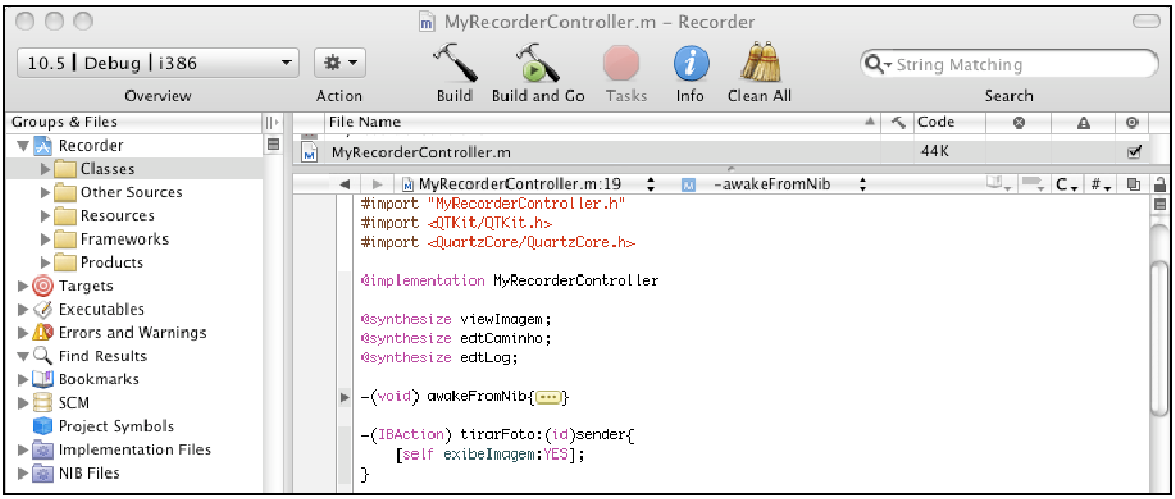
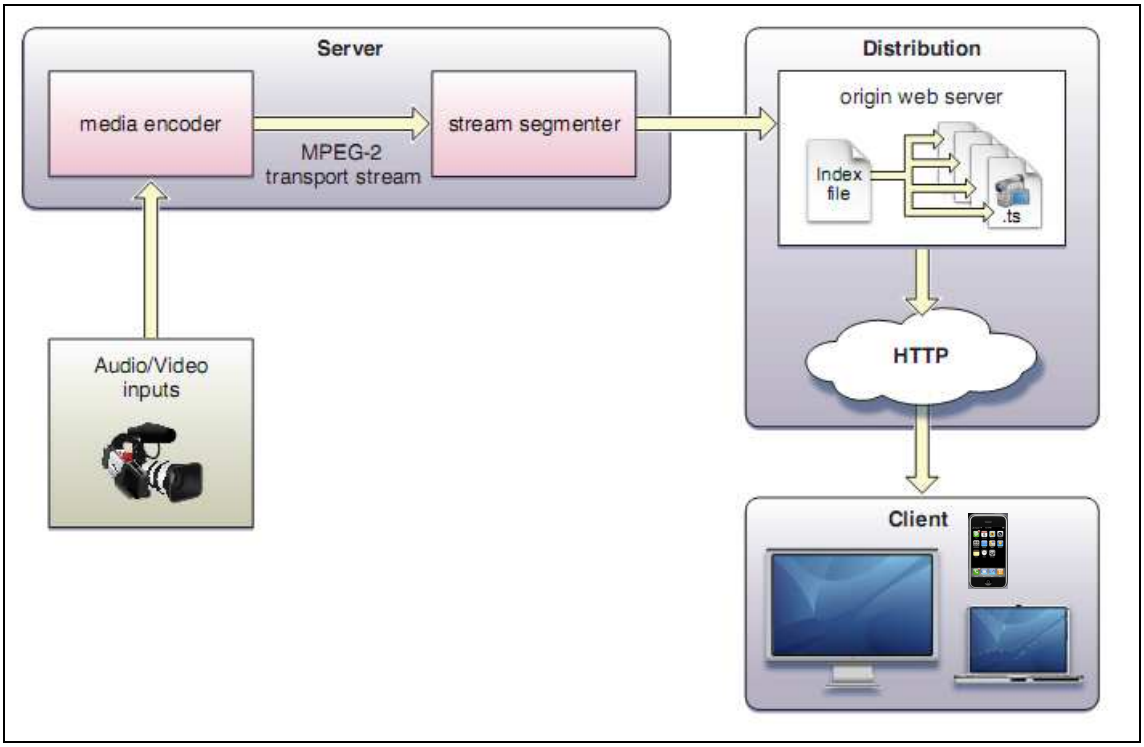


Figura 12 - Ambiente de desenvolvimento

3 DESENVOLVIMENTO

O desenvolvimento do visualizador de imagens no iPhone envolveu as fases de instalação e configuração do ambiente de desenvolvimento XCode, introdução ao desenvolvimento e depuração de aplicações, levantamento de requisitos, especificação e posterior implementação da aplicação, bem como a documentação das operacionalidades com o intuito de orientar o usuário a fazer uso das funcionalidades que foram disponibilizadas. Neste capítulo, além do detalhamento das fases acima, são apresentados os resultados obtidos.

O trabalho foi desenvolvido em duas partes, uma aplicação chamada Gerador e uma aplicação chamada Cliente. No Gerador, que deve ser instalado em um computador pessoal ou notebook (possuindo uma câmera de vídeo), desenvolveu-se uma aplicação responsável por capturar as imagens oriundas da câmera. No Cliente, que por sua vez deve ser instalado no iPhone, desenvolveu-se uma aplicação que carrega as imagens capturadas pelo Gerador. A Figura 13 apresenta o contexto do sistema.



Fonte: Streaming Media Guide (2009).

Figura 13 - Exemplo retirado do guia *Streaming* Media Guide

3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

Para o desenvolvimento do sistema, foram levantados requisitos funcionais e não funcionais, descritos nos Quadro 2 e Quadro 3, respectivamente.

REQUISITOS FUNCIONAIS	CASO DE USO
RF01: Permitir ao usuário visualizar, a partir do iPhone, imagens geradas em computador com uma *webcam*.	UC03, UC04, UC05
RF02: Permitir ao usuário gravar e excluir os locais de acesso do iPhone.	UC01 e UC02
RF03: Permitir que a aplicação geradora capture imagens instantâneas.	UC06
RF04: Deverá ser desenvolvido o programa gerador, responsável pela captura das imagens da câmera do computador.	
RF05: Deverá ser desenvolvido o programa cliente, responsável por receber e exibir as imagens geradas pelo Gerador.	

Quadro 2 - Requisitos funcionais

REQUISITOS NÃO FUNCIONAIS
RNF01: Ser desenvolvido utilizando análise orientada a objetos.
RNF02: As aplicações devem ser desenvolvidas em sistema operacional Mac OS.
RNF03: As aplicações devem ser desenvolvidas utilizando ambiente XCode (para codificação) e Interface Builder (para criação das interfaces).
RNF04: O acesso deve ser utilizando-se rede TCP/IP.

Quadro 3 – Requisitos não funcionais

3.2 ESPECIFICAÇÃO

A especificação do presente trabalho foi desenvolvida utilizando a notação *Unified Modeling Language* em conjunto com a ferramenta Enterprise Architect, gerando como produtos os diagramas de casos de uso, de classes e de seqüência apresentados nas seções seguintes.

3.2.1 Casos de uso

O diagrama de casos de uso é representado pela Figura 14 com as interações com o usuário.

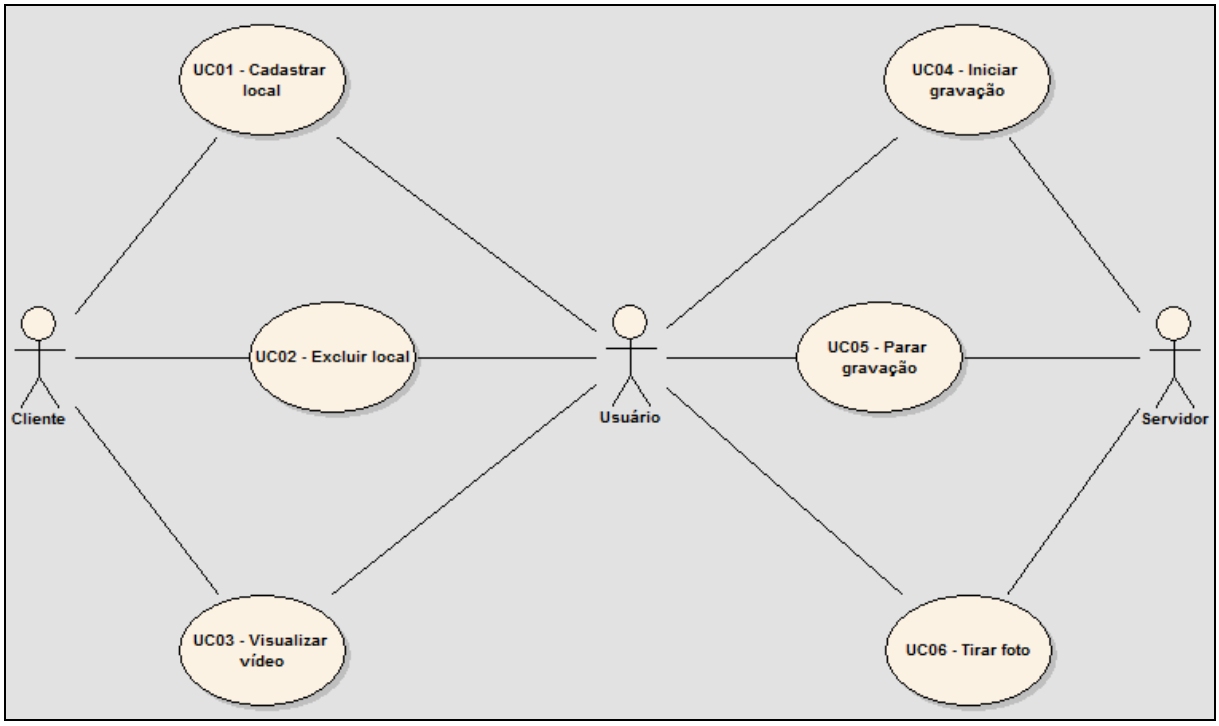


Figura 14 - Diagrama de casos de uso

3.2.1.1 Cadastrar local

O primeiro caso de uso (Quadro 4), designado `Cadastrar local`, descreve como um usuário deve proceder para cadastrar um novo local de monitoramento.

UC01 – Cadastrar local: possibilita ao usuário cadastrar locais de monitoramento.	
Requisitos atendidos	RF02.
Pré-condições	Não possui.
Cenário principal	1) O usuário inicia a aplicação Cliente. 2) O usuário clica no botão `+ (mais)`. 3) O usuário preenche os dados solicitados, ou seja, o nome e endereço. 4) O usuário clica no botão `Save`.
Fluxo alternativo 01	Cancelar inclusão: 1) No passo 3 do cenário principal, caso o usuário desejar cancelar a inclusão de um novo local, este deve clicar no botão `Cancel`.
Pós-condições	Um novo local é cadastrado no banco de dados.

Quadro 4 - Caso de uso UC01

3.2.1.2 Excluir local

O segundo caso de uso (Quadro 5), designado `Excluir local`, descreve como um usuário deve proceder para excluir um local de monitoramento.

UC02 – Excluir local: possibilita ao usuário excluir locais de monitoramento.	
Requisitos atendidos	RF02.
Pré-condições	O caso de uso UC01 deve ter sido executado pelo menos uma vez, ou seja, deve haver pelo menos um local cadastrado.
Cenário principal	1) O usuário inicia a aplicação Cliente. 2) O usuário clica no botão `Edit`. 3) O usuário clica no botão `– (menos)`. 4) O usuário clica no botão `Delete`. 5) O usuário clica no botão `Done`.
Fluxo alternativo 01	Cancelar exclusão: 1) No passo 3 do cenário principal, caso o usuário desejar cancelar a exclusão do local, este deve clicar novamente no botão – (menos) e então no `Done`.
Pós-condições	Um local é excluído do banco de dados.

Quadro 5 - Caso de uso UC02

3.2.1.3 Visualizar vídeo

O terceiro caso de uso (Quadro 6), designado `Visualizar vídeo`, descreve como um usuário deve proceder para visualizar um local de monitoramento.

UC03 – Visualizar vídeo: possibilita ao usuário visualizar locais de monitoramento.	
Requisitos atendidos	RF01.
Pré-condições	O caso de uso UC01 deve ter sido executado pelo menos uma vez, ou seja, deve haver pelo menos um local cadastrado. Também é necessário que o Gerador esteja em execução no computador.
Cenário principal	1) O usuário inicia a aplicação Cliente. 2) A lista de locais disponíveis é exibida. 3) O usuário seleciona um local, clicando sobre a linha. 4) O vídeo é exibido.
Fluxo alternativo 01	Cancelar exibição do vídeo: 1) No passo 4 do cenário principal, caso o usuário desejar cancelar a exibição do vídeo, este deve clicar no botão `Done` do iPhone
Pós-condições	O vídeo no local selecionado é exibido.

Quadro 6 - Caso de uso UC03

3.2.1.4 Iniciar gravação no Gerador

O quarto caso de uso (Quadro 7), designado `Iniciar gravação`, descreve como um usuário deve proceder para iniciar a gravação do vídeo em um local de monitoramento.

UC04 – Iniciar gravação: possibilita ao usuário gravar o vídeo de um local de monitoramento.	
Requisitos atendidos	RF01.
Pré-condições	O computador deve possuir uma câmera de vídeo instalada e configurada.
Cenário principal	1) O usuário inicia a aplicação Gerador. 2) O usuário clica o botão `iniciar gravação`.
Pós-condições	O vídeo é gravado.

Quadro 7 - Caso de uso UC04

3.2.1.5 Parar gravação

O quinto caso de uso (Quadro 8), designado `Parar gravação`, descreve como um usuário deve proceder para finalizar a gravação do vídeo em um local de monitoramento.

UC05 – Parar gravação: possibilita ao usuário finalizar a gravação do vídeo de um local de monitoramento.	
Requisitos atendidos	RF01.
Pré-condições	O caso de uso UC04 deve ter sido executado, ou seja, a aplicação Gerador deve ter iniciado uma gravação.
Cenário principal	1) O usuário clica o botão `parar`.
Pós-condições	O vídeo é finalizado.

Quadro 8 - Caso de uso UC05

3.2.1.6 Tirar foto

O sexto caso de uso (Quadro 9), designado `Tirar foto`, descreve como um usuário deve proceder para fotografar um local de monitoramento.

UC06 – Tirar foto: possibilita ao usuário fotografar um local de monitoramento.	
Requisitos atendidos	RF03.
Pré-condições	Não possui.
Cenário principal	1) O usuário inicia a aplicação Gerador. 2) O usuário pressiona o botão `snapshot`.
Pós-condições	Uma imagem é gravada no local apontado pelo campo `Local`. O campo último `snapshot` é preenchido com a imagem gravada. O campo `Log` recebe o endereço da imagem gravada.

Quadro 9 - Caso de uso UC06

3.2.2 Diagrama de classes

O diagrama de classes apresenta uma visão de como as classes estão estruturadas e relacionadas. Nesta seção são descritas as classes necessárias para o desenvolvimento da aplicação Gerador e da aplicação Cliente.

3.2.2.1 Diagrama de classes do Gerador

A Figura 15 demonstra a classe `MyRecorderController`, utilizada na aplicação Gerador.

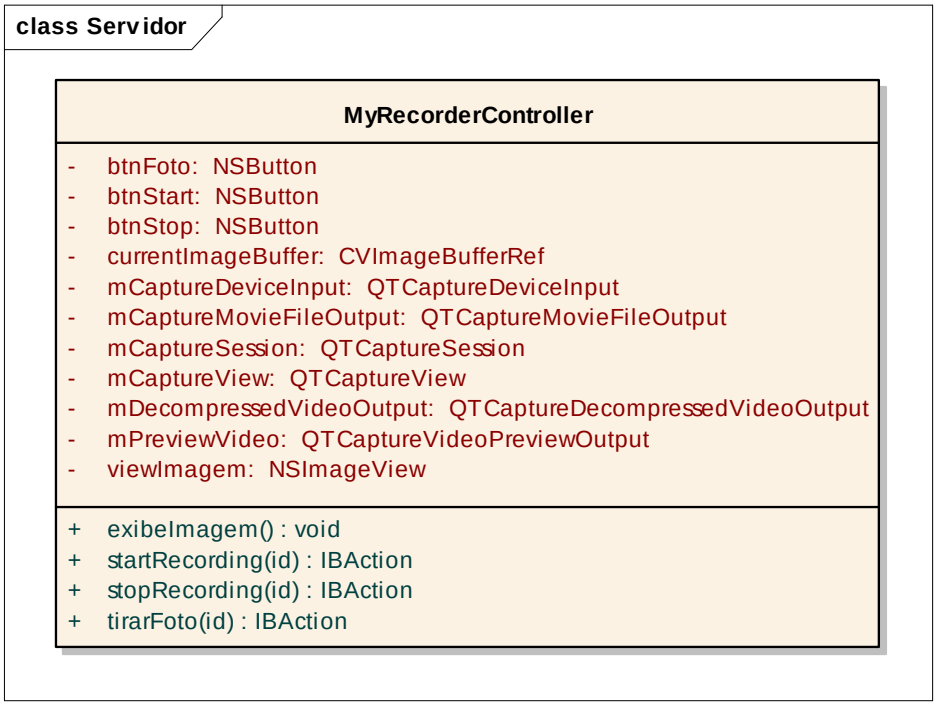


Figura 15- Diagrama de classes aplicação Gerador

A classe `MyRecorderController` possui os seguintes métodos:

a) `exibeImagem`: método que fotografa um quadro e salva a imagem em um diretório;

b) `startRecording`: método que inicia a gravação dos vídeos;

c) `stopRecording`: método que para a gravação dos vídeos;

d) `tirarFoto`: este método é executado por um botão, e chama o método `exibeImagem`.

3.2.2.2 Diagrama de classes do Cliente

A Figura 16 demonstra as classes envolvidas no desenvolvimento da aplicação Cliente.

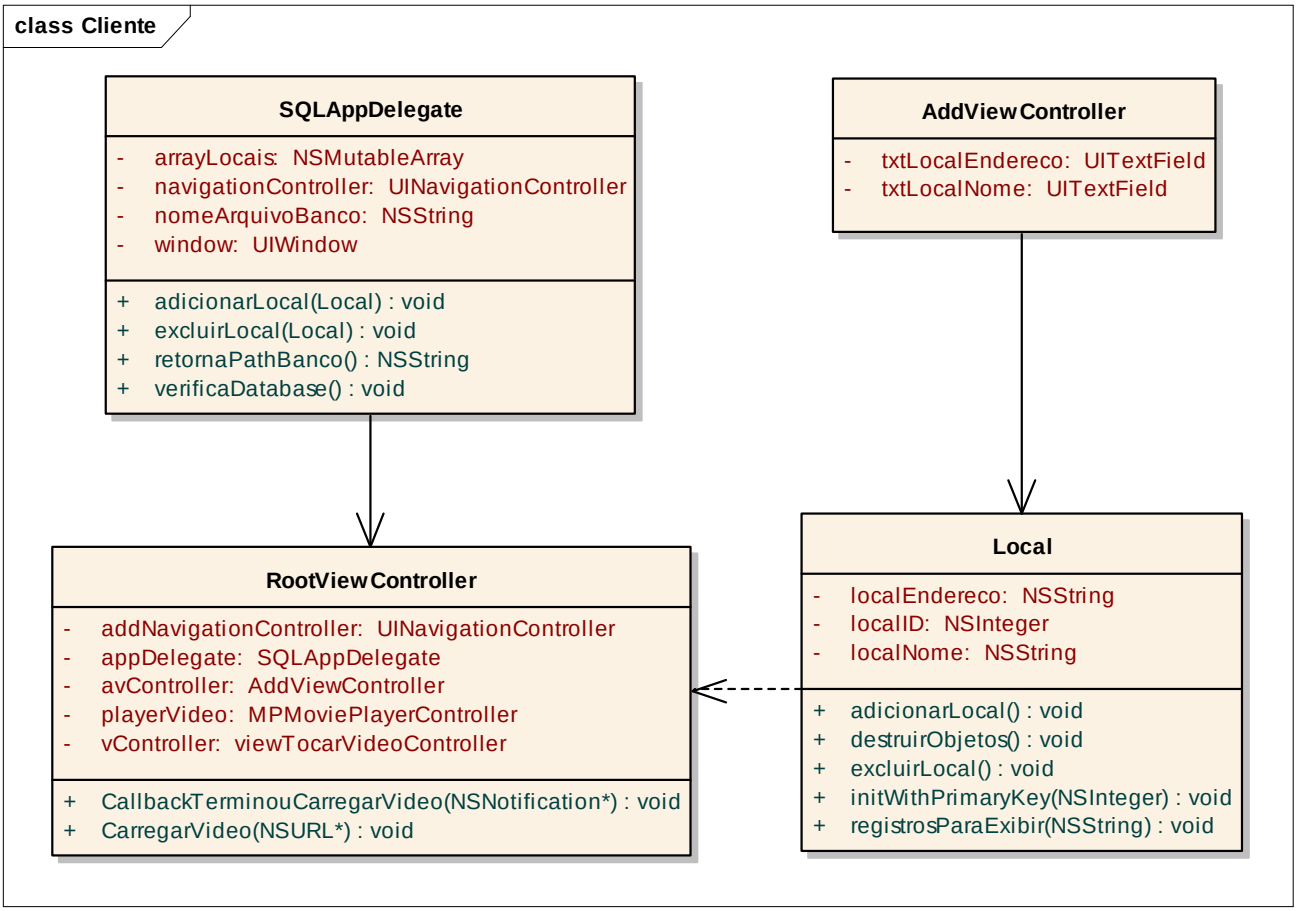


Figura 16 - Diagrama de classes aplicação Cliente

A classe `SQLAppDelegate` é responsável por exibir a janela do iPhone, e possui os seguintes métodos:

a) `adicionarLocal`: este método insere um registro do banco de dados (executa o método `adicionarLocal` da classe `Local`);
b) `excluirLocal`: este método exclui um registro do banco de dados (executa o método `excluirLocal` da classe `Local`);
c) `retornaPathBanco`: método que retorna o caminho onde o banco de dados foi criado;
d) `verificaDatabase`: método que verifica se existe banco de dados criado. Se o banco não existir, o mesmo é criado.

Já a classe `RootViewController` é responsável pelo exibição dos vídeos. Esta classe possui os seguintes métodos:

a) `CallbackTerminouCarregarVideo`: este método é executado quando um vídeo termina de carregar;
b) `CarregarVideo`: método que carrega um vídeo a partir de um endereço.

A classe `Local`, por sua vez, é responsável pela manutenção do cadastro de locais de exibição, e possui os seguintes métodos:

a) `adicionarLocal`: método responsável por adicionar um registro de local no banco de dados;
b) `destruirObjetos`: método responsável por desalocar da memória os objetos desta classe;
c) `excluirLocal`: método responsável por excluir um registro do banco de dados;
d) `initWithPrimaryKey`: método que retorna uma nova chave primária, para inserção de um novo local de monitoramento;
e) `registrosParaExibir`: este método executa uma consulta no banco de dados do cliente, na tabela de locais e retorna todos os registros encontrados.

3.2.3 Diagrama de sequência

O diagrama de seqüência apresenta uma visão interna do processo e da comunicação entre as classes para cada caso de uso. Na seção 3.2.3.1 serão apresentados dos diagramas de sequência referentes aos casos de uso `UC05` e `UC06`. Na seção 3.2.3.2 serão apresentados os diagramas de sequência referentes aos casos de uso `UC01` e `UC03`.

3.2.3.1 Diagrama de sequência do Gerador

A Figura 17 demonstra o diagrama de sequência referente ao caso de uso `UC05`.

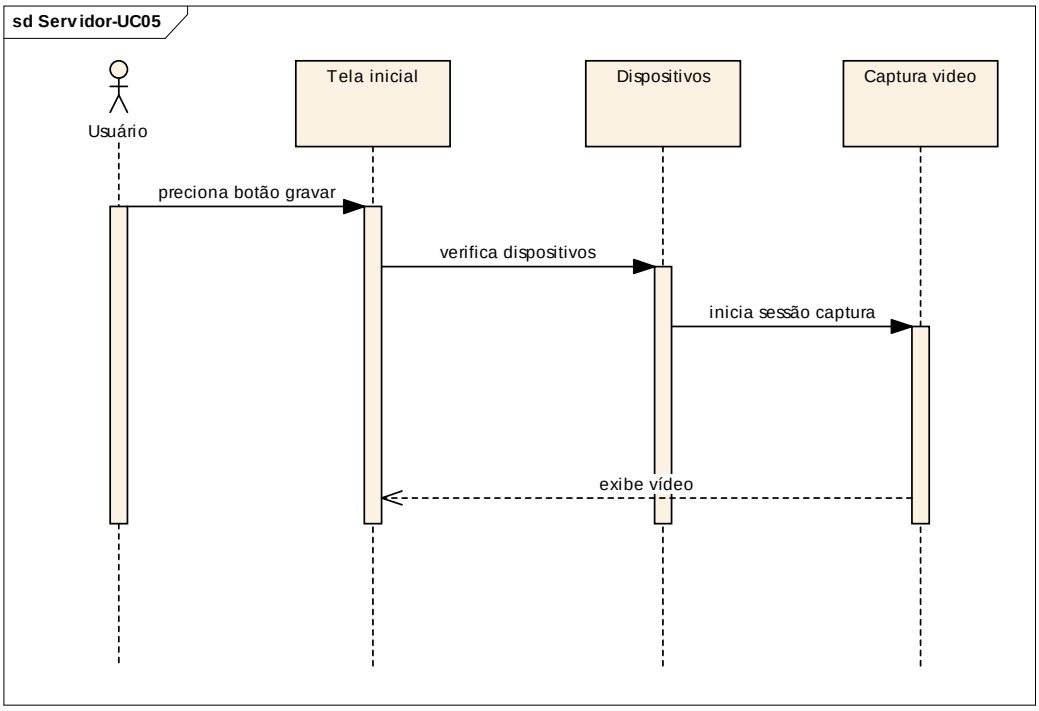


Figura 17 - Diagrama sequência referente caso de uso UC05

A Figura 18 demonstra o diagrama de sequência referente ao caso de uso `UC06`.

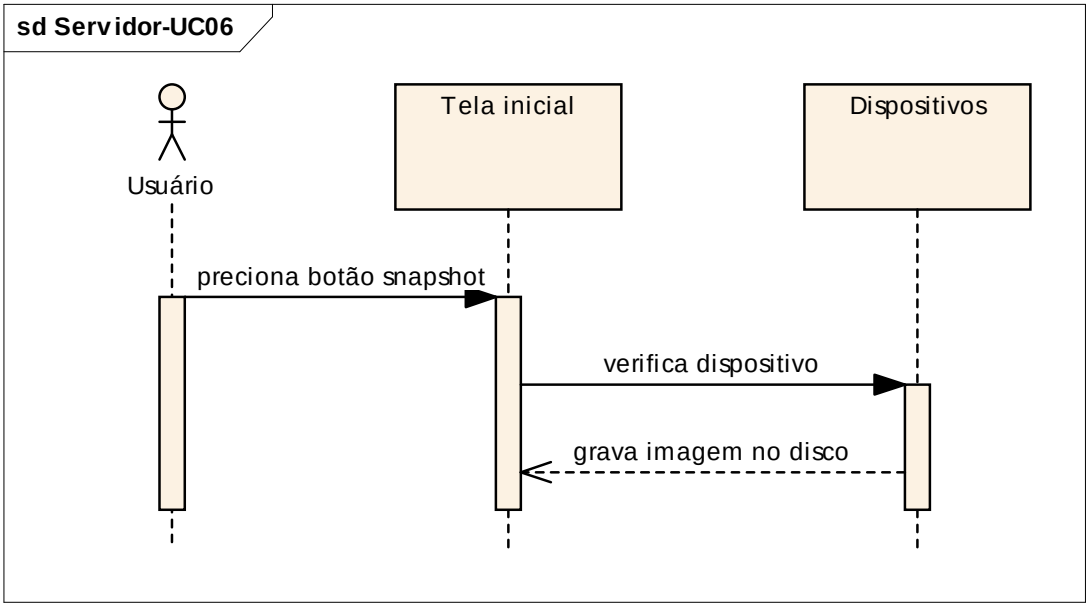


Figura 18 - Diagrama sequência referente caso de uso UC06

3.2.3.2 Diagrama de sequência do Cliente

A Figura 19 demonstra o diagrama de sequência referente ao caso de uso `UC01`.

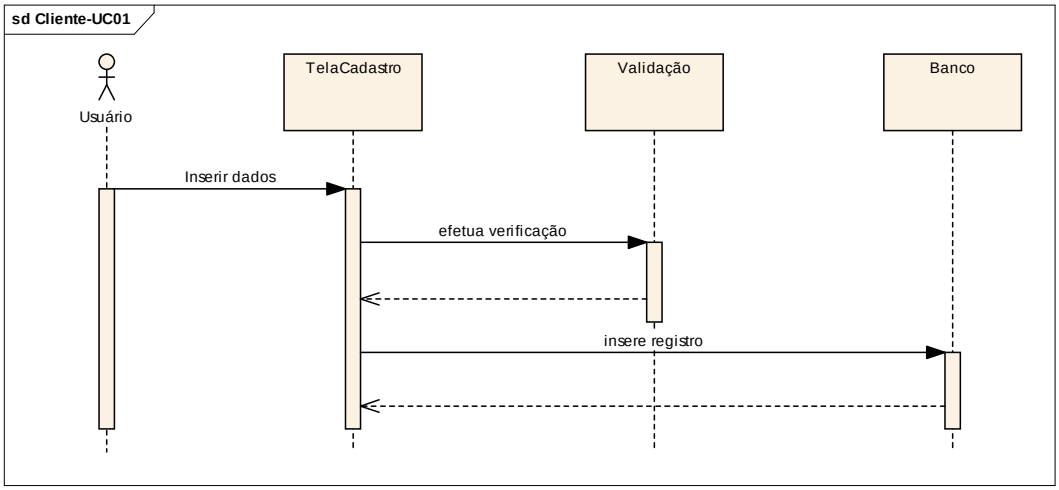


Figura 19 - Diagrama sequência referente caso de uso UC01

A Figura 20 demonstra o diagrama de sequência referente ao caso de uso `UC03`.

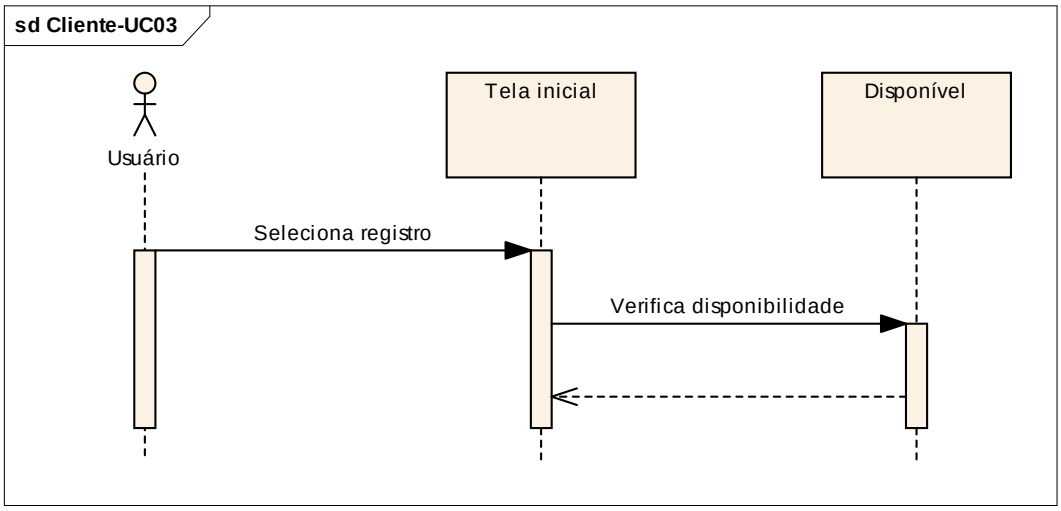


Figura 20 - Diagrama sequência referente caso de uso UC03

3.2.4 APLICACÃO GERADOR

A aplicação Gerador é instalado no computador ou notebook e é responsável pela captura e pelo envio das imagens. O objetivo do mesmo é capturar as imagens oriundas da câmera do computador e gravar em um arquivo de vídeo com tamanho de 240 *pixels*[4] de largura e 300 *pixels* de altura, com compactação H.264, que é, segundo Divx (2009), "[...] o formato avançado de codificação digital para vídeo de alta definição [...]". Este formato de tela e codificação de vídeo foi escolhido pelo fato de iPhone conseguir ler vídeos nesta codificação. A Figura 21 demonstra a tela da aplicação Gerador.

[4] *Pixel*: Termo utilizado para designar cada ponto na tela do computador.

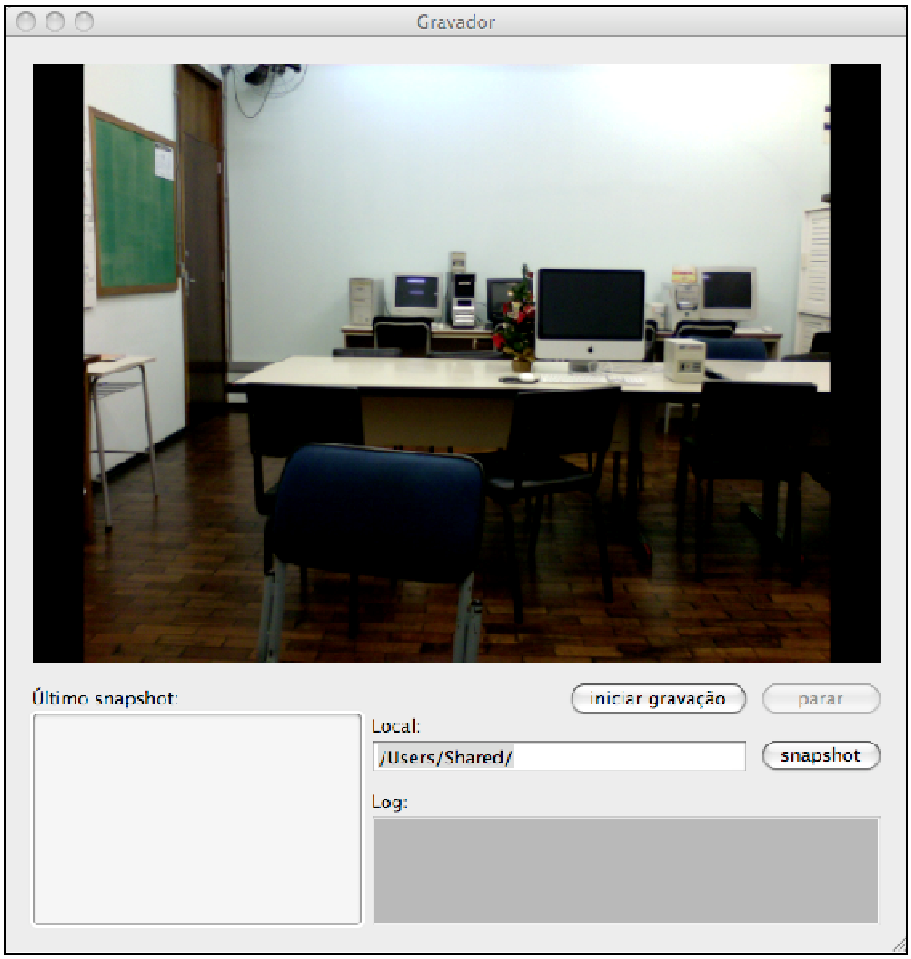


Figura 21 - Tela da aplicação Gerador

3.2.5 APLICAÇÃO CLIENTE

A aplicação Cliente é instalada no iPhone e é responsável por ler e exibir na tela os arquivos disponibilizados pelo Gerador. No Cliente é possível gravar locais de monitoramento, ou seja, o usuário consegue armazenar no próprio iPhone os locais onde possuem *webcams* disponíveis. A Figura 22 a tela inicial da aplicação Cliente.

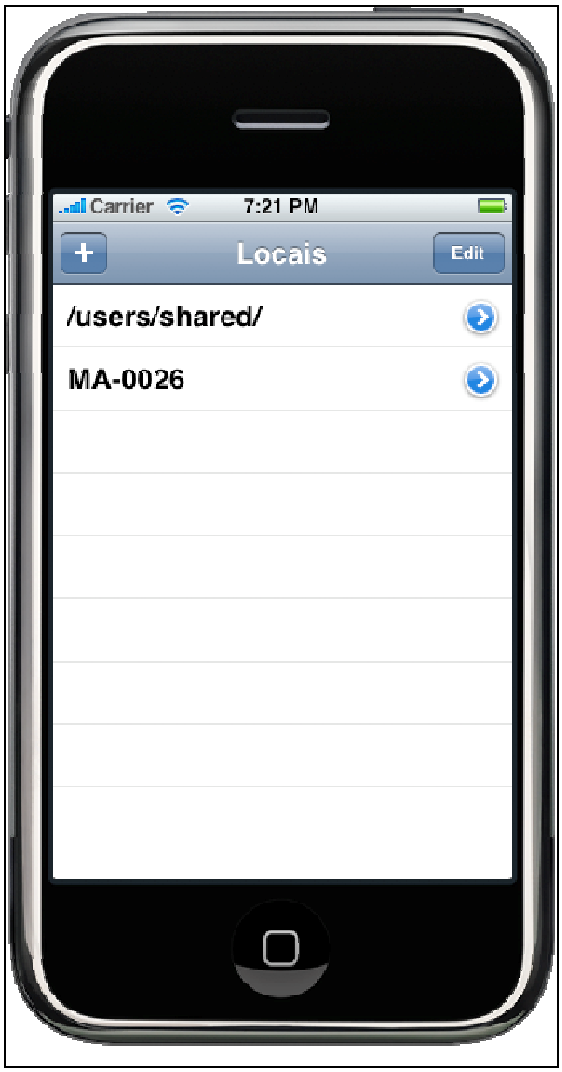


Figura 22 - Tela inicial da aplicação Cliente

3.3 IMPLEMENTAÇÃO

A seguir são descritas as técnicas e ferramentas utilizadas na implementação, bem como detalhes das principais classes e rotinas implementadas durante o desenvolvimento das aplicações Gerador e Cliente.

3.3.1 Técnicas e ferramentas utilizadas

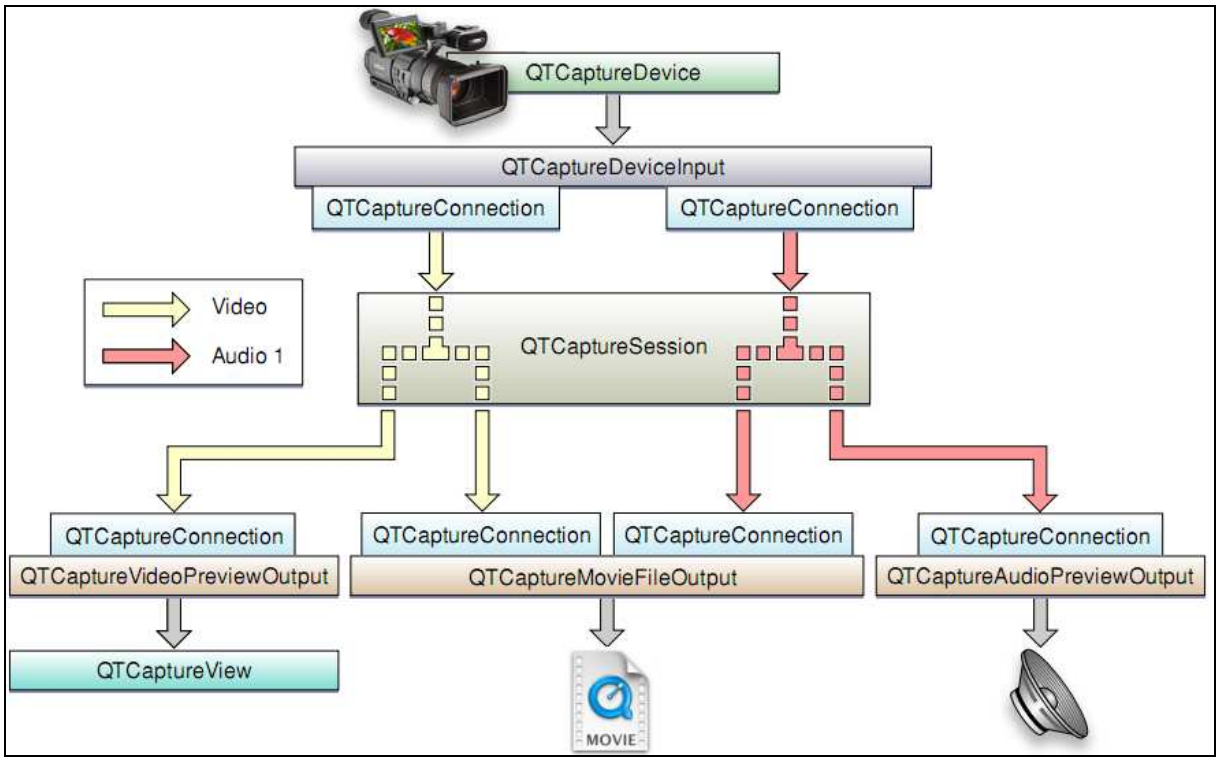
Nesta seção são demonstradas as técnicas e ferramentas utilizadas no desenvolvimento

da aplicação Gerador e da aplicação Cliente.

3.3.1.1 Gerador

A aplicação Gerador foi implementada sob o paradigma da orientação a objetos na linguagem programação Objective-C, utilizando-se o ambiente de desenvolvimento XCode versão 3.1.3 (para codificação), ambiente Interface Builder versão 3.1.2 (para desenho de interfaces), ambos em sistema operacional Mac OS X versão 10.5.7. Este conjunto oferece todos os recursos necessários para o desenvolvimento da aplicação.

Para efetuar a captura e gravação das imagens da câmera, foram utilizadas as classes do QuickTime versão 7.2.1 (API QTKit), que é o tocador oficial de áudio e vídeo para o sistema operacional Mac OS. A Figura 23 exemplifica os passos para captura de áudio e vídeo.



Fonte: QTKit Capture (2009).

Figura 23 - Passos necessários para captura de áudio e vídeo

A captura das imagens é feita em uma sessão, definida pela classe `QTCaptureSession`. Na sessão, são instanciados os dispositivos de captura de imagens, definidos pelas classes `QTCaptureDevice` e `QTCaptureDeviceInput`. A conexão dos dispositivos com a sessão de captura é feita com objetos da classe `QTCaptureConnection`.

Dentro da sessão de captura podem ser adicionados objetos das classes `QTCaptureVideoPreviewOutput` (para exibição do vídeo), `QTCaptureAudioPreviewOutput` (para áudio) e `QTCaptureMovieFileOutput` (para gravação de arquivo de vídeo).

No Quadro 10 é demonstrado cabeçalho da classe desenvolvida para criação e ligação dos componentes de captura.

```
01 #import <Cocoa/Cocoa.h>
02 #import <QTKit/QTKit.h>
03
04
05 @interface MyRecorderController : NSOpenGLView {
06
07    IBOutlet QTCaptureView      *mCaptureView;
08    QTCaptureSession            *mCaptureSession;
09    QTCaptureMovieFileOutput    *mCaptureMovieFileOutput;
10    QTCaptureDeviceInput        *mCaptureDeviceInput;
11    QTCaptureVideoPreviewOutput *mPreviewVideo;
12    QTCaptureDecompressedVideoOutput *mDecompressedVideoOutput;
13
14    IBOutlet NSButton *btnStart;
15    IBOutlet NSButton *btnStop;
16    IBOutlet NSButton *btnFoto;
17
18    IBOutlet NSImageView  *viewImagem;
19    CVImageBufferRef currentImageBuffer;
20
21 }
22
23 @property (nonatomic, retain) IBOutlet NSImageView *viewImagem;
24
25 -(IBAction) startRecording:(id)sender;
26 -(IBAction) stopRecording:(id)sender;
27 -(IBAction) tirarFoto:(id)sender;
28 -(void) exibeImagem;
29
30 @end
```

Quadro 10 - Definição do cabeçalho da classe de captura

Nas linhas `1` e `2` são importados os frameworks necessários. Na linha `5` é definida a classe `MyRecorderController`. Nas linhas `7` a `12` são definidos os objetos das classes citadas acima, responsáveis pela criação da sessão de captura de vídeo, instanciação dos dispositivos de captura, e definição dos objetos de conexão dos dispositivos com a sessão. Nas linhas `14` a `16` são definidos os botões da aplicação. Nas linhas `25` a `28` são definidos as ações dos botões.

No Quadro 11 é demonstrada a instanciação da sessão de captura e definição tipo de mídia capturada.

```
                                    ...
01 mCaptureSession = [[QTCaptureSession alloc] init];
02 QTCaptureDevice *device = [QTCaptureDevice
defaultInputDeviceWithMediaType:QTMediaTypeVideo];
                                    ...
```

Quadro 11 - Sessão de captura

A variável `mCaptureSession` é responsável pela criação da sessão de captura. O trecho de código `[[QTCaptureSession alloc] init]` fará a alocação de espaço na memória para a variável `mCaptureSession`, através do método `alloc` e iniciará a mesma com o método `init`.

Na linha `2`, é criado um dispositivo de captura e definido tipo de mídia que o mesmo irá capturar, neste caso, vídeo, definido pela constante `QTMediaTypeVideo`.

O Quadro 12 demonstra a instanciação do dispositivo de entrada, que faz a intermediação entre o dispositivo de captura (dispositivo físico) e a conexão com a sessão de captura.

```
                                        ...
09 mCaptureDeviceInput = [[QTCaptureDeviceInput alloc]
initWithDevice:device];
10 sucess = [mCaptureSession addInput:mCaptureDeviceInput error:&error];
                                        ...
```

Quadro 12 - Dispositivo de entrada

Na linha `9`, a variável `mCaptureDeviceInput` é alocada na memória com o procedimento `[QTCaptureDeviceInput alloc]` e iniciada com o dispositivo de captura, através do procedimento `initWithDevice`. Logo abaixo, na linha `10`, o dispositivo de entrada é adicionado à sessão de captura.

No Quadro 13 é demonstrado o código que define o objeto para exibir a prévia do vídeo, ou seja, exibir as imagens que a câmera está capturando.

```
                                        ...
20 mDecompressedVideoOutput = [[QTCaptureDecompressedVideoOutput alloc]
init];
21 [mDecompressedVideoOutput setDelegate: self];
22 [mCaptureSession addOutput:mDecompressedVideoOutput error:&error];
23
24 mPreviewVideo = [[QTCaptureVideoPreviewOutput alloc] init];
25 [mCaptureSession addOutput:mPreviewVideo error:&error];
                                        ...
```

Quadro 13 - Prévia do vídeo de saída

Na linha `20` do trecho de código acima, é instanciado um objeto da classe `QTCaptureDecompressedVideoOutput`, responsável pela captura dos quadros da câmera. Este objeto então deve ser adicionado também na sessão de captura. Também é definido um objeto da classe `QTCaptureVideoPreviewOutput`, para que este seja responsável por exibir a prévia do vídeo.

Para a gravação do vídeo gravado em um arquivo de vídeo, o código utilizado é demonstrado no Quadro 14.

```
                                        ...
35 mCaptureMovieFileOutput = [[QTCaptureMovieFileOutput alloc] init];
36 sucess = [mCaptureSession addOutput:mCaptureMovieFileOutput
error:&error];
                                        ...
```

Quadro 14 - Código para gravação do vídeo em arquivo

A classe responsável pela gravação em um arquivo de vídeo é a `QTCaptureMovieFileOutput`. Este objeto também deve ser adicionado à sessão de captura. Depois de efetuadas as ligações dos componentes necessários para a gravação do vídeo, deve ser definida a codificação de saída. Optou-se pela codificação definida na constante `QTCompressionOptions240SizeH264Video`, pois esta contém as proporções de tela ideais para o iPhone, além do formato de vídeo suportado pelo mesmo.

O Quadro 15 demonstra a definição da codificação.

```
                                        ...
50 // Cria um enumerador para percorrer todos os dispositivos de captura
conectados ao computador
51 NSEnumerator *connectionEnumerator = [[mCaptureMovieFileOutput
connections] objectEnumerator];
52 QTCaptureConnection *connection;
53
54 // Para cada dispositivo, define a codificação de vídeo
55 while ((connection = [connectionEnumerator nextObject])){
56    NSString *mediaType = [connection mediaType];
57    QTCompressionOptions *compressionOptions = nil;
58
59    if ([mediaType isEqualToString:QTMediaTypeVideo]){
60          compressionOptions = [QTCompressionOptions
compressionOptionsWithIdentifier:@"QTCompressionOptions240SizeH264Video"];6
61
62    }
63    else{
64          compressionOptions = [QTCompressionOptions
compressionOptionsWithIdentifier:@"QTCompressionOptionsHighQualityAACAudio"
];
65    }
66    // Define a compressão do vídeo no arquivo de saída
67    [mCaptureMovieFileOutput setCompressionOptions:compressionOptions
forConnection:connection];
68
69    [mCaptureView setCaptureSession:mCaptureSession];
70 }
71
72 [mCaptureSession startRunning];
                                        ...
```

Quadro 15 - Definição da codificação do vídeo

O código do Quadro 15 define a codificação de vídeo para todos os dispositivos conectados ao computador. Na linha `72`, é iniciada a sessão de captura.

O Quadro 16 demonstra o código utilizado para a gravação do vídeo.

```
80 -(IBAction) startRecording:(id)sender{
81    gravando = YES;
82    // Caption do botão
83    [btnStart setTitle:@"Gravando..."];
84
85    NSString *str;
                                    ...
89    // Local de gravação
90    NSURL *url = [NSURL fileURLWithPath:str];
91    [mCaptureMovieFileOutput recordToOutputFileURL:url
bufferDestination:QTCaptureFileOutputBufferDestinationNewFile];
92
93 }
```

Quadro 16 - Gravação do vídeo

O método `recordToOutputFileURL` inicia a gravação do vídeo no local especificado pelo parâmetro `url`.

O Quadro 17 demonstra o código utilizado para parar a gravação de vídeo e gravar o arquivo em disco.

```
100 -(IBAction) stopRecording:(id)sender{
101   gravando = NO;
102   [btnStart setTitle:@"Start"];
103   [mCaptureMovieFileOutput recordToOutputFileURL:nil];
104 }
```

Quadro 17 - Parar gravação

Quando o método `recordToOutputFileURL` recebe `nil` (valor nulo) como parâmetro, vídeo é finalizado e gravado em disco, no local especificado pela variável `url` (na gravação do vídeo).

O Quadro 18 demonstra o código utilizado para gravar uma imagem.

```
106 -(IBAction) tirarFoto:(id)sender{
107   NSBitmapImageRep *bits = [[NSBitmapImageRep alloc]
initWithCIImage:[CIImage imageWithCVImageBuffer:currentImageBuffer]];
108
109   NSData *data = [bits representationUsingType: NSPNGFileType
properties: nil];
                                    ...
110   //Salva a imagem no disco
111   [data writeToFile:url atomically:NO];
112
113   //joga o buffer da imagem na view
114   NSImage *outputImage = [[NSImage alloc] initWithData:data];
115   [viewImagem setImage:outputImage];
116 }
```

Quadro 18 - Método para gravação de imagem em disco

Na linha `107` é instanciado um objeto da classe `NSBitmapImageRep`, responsável pela renderização de imagens. O objeto é iniciado com um objeto da classe `CCImage`, responsável pela representação da imagem. O objeto da classe `CCImage` é iniciado com o quadro atual, capturado pela câmera.

Logo em seguida, na linha `109`, é instanciado um objeto do tipo `NSData`, que nada mais

é que um *buffer*. Este *buffer* é carregado com o objeto `bits` (que neste ponto contém a imagem), transformando a imagem em um *buffer*. Então este *buffer* é gravado em disco, através do método `writeToFile` (linha `111`).

As últimas linhas (`114` e `115`) iniciam um objeto da classe `NSImage`, que é uma classe para manipulação de imagens, com o *buffer* criado. Logo em seguida a imagem é apresentada na tela, através do objeto `viewImage`, com o método `setImage`.

3.3.1.2 Cliente

A aplicação Cliente também foi implementada sob o paradigma da orientação a objetos na linguagem programação Objective-C, utilizando-se o ambiente de desenvolvimento XCode versão 3.1.3 (para codificação), ambiente Interface Builder versão 3.1.2 (para desenho de interfaces), ambos em sistema operacional Mac OS X versão 10.5.7. Esta aplicação possui recursos de gravação de locais de visualização e visualização de captura.

3.3.1.2.1 Gravação de locais

O iPhone possui um banco de dados chamado SQLite, no qual é possível definir tabelas e atributos e ligação entre as mesmas. Para a criação do banco de dados a ser utilizado no iPhone é necessário abrir um terminal de comando, disponível no sistema operacional Mac OS, e digitar o comando apresentado no Quadro 19.

```
:~ Sqlite3 LocaisDB.sql
```

Quadro 19 - Comando para criação do banco de dados

O comando apresentado cria o arquivo `LocaisDB.sql`, que representa o banco de dados. Este arquivo deve ser então adicionado ao projeto criado pelo XCode, na pasta *Resources*. Deve-se então, criar a tabela que armazenará os registros. A tabela é criada com o comando apresentado no Quadro 20.

```
:~ CREATE TABLE locais (id INTEGER PRIMARY KEY, nome VARCHAR(50),
endereco VARCHAR(80));
```

Quadro 20 - Comando para criação de tabelas

Depois de criada a tabela, podem ser gravados os locais. Para gravar registros no banco de dados criado, é utilizado o procedimento `adicionarLocal`, apresentado no Quadro 21.

```
05 - (void) adicionarLocal {
06
07    if(addStmt == nil){
08          // SQL de inserção
09          const char *sql = "INSERT INTO locais (nome, endereco) VALUES
(?, ?)";
10          // Cria a query
11          if(sqlite3_prepare_v2(database, sql, -1, &addStmt, NULL) !=
SQLITE_OK)
12                NSAssert1(0, @"Erro ao criar query de insercao: '%s'",
sqlite3_errmsg(database));
13    }
14    // Definição dos parâmetros
15    sqlite3_bind_text(addStmt, 1, [localNome UTF8String], -1,
SQLITE_TRANSIENT);
16    sqlite3_bind_text(addStmt, 2, [localEndereco UTF8String], -1,
SQLITE_TRANSIENT);
17    // Verifica se a query foi executada com sucesso
18    if(SQLITE_DONE != sqlite3_step(addStmt))
19          NSAssert1(0, @"Erro ao inserir registro: '%s'",
sqlite3_errmsg(database));
20    else
21          localID = sqlite3_last_insert_rowid(database);
22    // Reinicia a query, para uma nova inserção
23    sqlite3_reset(addStmt);
24 }
```

Quadro 21 - Procedimento `adicionarLocal`

A exclusão de registros é realizada através do procedimento `excluirLocal`, apresentado no Quadro 22.

```
25 - (void) excluirLocal {
26
27    if(deleteStmt == nil) {
28          const char *sql = "DELETE FROM locais WHERE id = ?";
29          if(sqlite3_prepare_v2(database, sql, -1, &deleteStmt, NULL) !=
SQLITE_OK)
30                NSAssert1(0, @"Erro ao criar a query de exclusao '%s'",
sqlite3_errmsg(database));
31    }
32    // Passagem do parâmetro
33    sqlite3_bind_int(deleteStmt, 1, localID);
34
35    if (SQLITE_DONE != sqlite3_step(deleteStmt))
36          NSAssert1(0, @"Erro ao excluir registro: '%s'",
sqlite3_errmsg(database));
37    // Reinicia a query, para uma nova exclusão
38    sqlite3_reset(deleteStmt);
39 }
```

Quadro 22 - Procedimento `excluirLocal`

A seleção de registros é feita através do procedimento `registrosParaExibir`, apresentada no Quadro 23.

```
41 + (void) registrosParaExibir:(NSString *)pathBanco {
42
43    SQLAppDelegate *appDelegate = (SQLAppDelegate *)[[UIApplication
sharedApplication] delegate];
44    // Cria uma conexão com o banco de dados
45    if (sqlite3_open([pathBanco UTF8String], &database) == SQLITE_OK) {
46          // Query de seleção de registros
47          const char *sql = "SELECT * FROM locais";
48          // Define o resultset, ou seja objeto que receberá o resultado
da consulta
49          sqlite3_stmt *resultSet;
50          if(sqlite3_prepare_v2(database, sql, -1, &resultSet, NULL) ==
SQLITE_OK) {
51
52                // Neste passo é percorrido o resultset, criando um
objeto da classe Local, atribuindo os valores selecionados e inserindo esse
objeto no array
53                while(sqlite3_step(resultSet) == SQLITE_ROW) {
54
55                      NSInteger primaryKey =
sqlite3_column_int(resultSet, 0);
56                      Local *localObj        = [[Local alloc]
initWithPrimaryKey:primaryKey];
57                      localObj.localNome     = [NSString
stringWithUTF8String:(char *)sqlite3_column_text(resultSet, 1)];
58                      localObj.localEndereco = [NSString
stringWithUTF8String:(char *)sqlite3_column_text(resultSet, 2)];
59
60                      localObj.isDirty = NO;
61
62                      // Adiciona o objeto criado no array
63                      [appDelegate.arrayLocais addObject:localObj];
64
65                    // Libera o objeto da memória
66                      [localObj release];
67                }
68          }
69    }
70    else
71          // Fecha a conexão com o banco de dados
72          sqlite3_close(database);
73
74    }
```

Quadro 23 - Seleção de registros

3.3.1.2.2 Visualização de vídeos capturados

Para a exibição dos vídeos, é utilizado um objeto da classe `MPMoviePlayerController`. O código apresentado no Quadro 24 demonstra a alocação de memória e inicialização do tocador de vídeo.

```
70 -(void)CarregarVideo:(NSURL *)url{
71
72    // cria o player do vídeo, passando o caminho
73    playerVideo = [[MPMoviePlayerController alloc]
initWithContentURL:url];
74
75    if (playerVideo){
76          // Cria uma notificação para iniciar a tocar o vídeo depois de
carregá-lo totalmente (aparece a tela “Carregando...”
77          [[NSNotificationCenter defaultCenter] addObserver:self
             selector:@selector(CallbackTerminouCarregarVideo:)
             name:MPMoviePlayerContentPreloadDidFinishNotification
object:nil];
78    }
79    else{
80          UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@"Erro"
message:@"Não foi possível reproduzir o vídeo!" delegate:nil
cancelButtonTitle:@"OK" otherButtonTitles:nil];
81          [alert show];
82          [alert release];
83    }
84 }
```

Quadro 24 - Código para reprodução do vídeo

3.3.2 Operacionalidade da implementação

A operacionalidade da aplicação é apresentada em função dos casos de uso da aplicação, fazendo o uso de imagens para facilitar o entendimento de cada uma das funcionalidades disponibilizadas.

Na seção 3.3.2.1 são apresentadas as funcionalidades da aplicação Gerador e na seção 3.3.2.2 são apresentadas as funcionalidades da aplicação Cliente.

3.3.2.1 Funcionalidades da aplicação Gerador

Nesta seção serão apresentadas as seguintes funcionalidades: `Iniciar gravação`, `Parar gravação` e `Tirar foto`.

3.3.2.1.1 Iniciar gravação

Esta funcionalidade permite ao usuário iniciar a gravação do vídeo em disco. O caso de

uso criado para esta funcionalidade é o `UC01 – Iniciar gravação`. Ao iniciar a gravação, o vídeo será criado no diretório apontado pelo campo `Local`, conforme demonstrado na Figura 24.

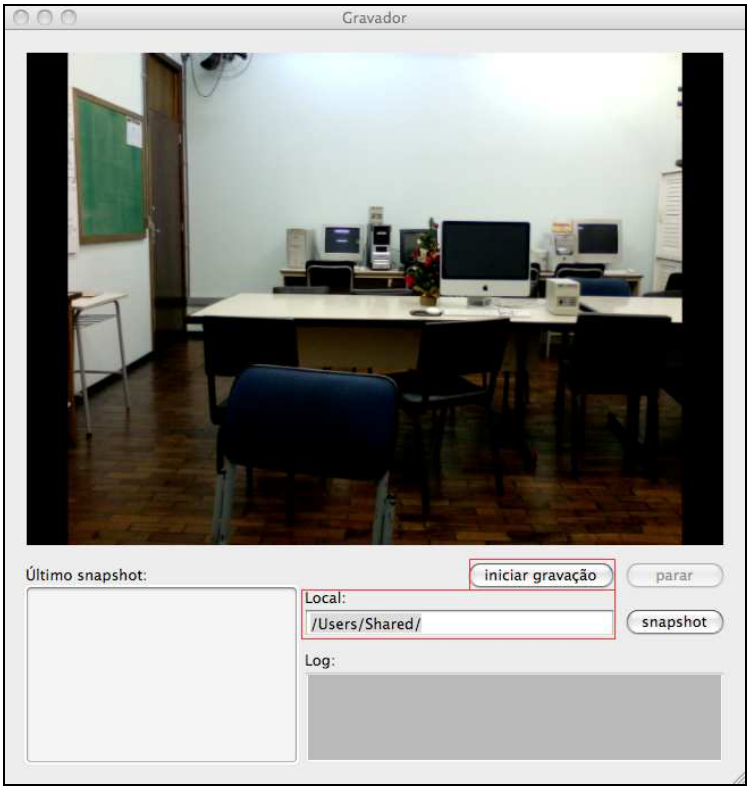


Figura 24 - Iniciar gravação

Para iniciar a gravação, o usuário deve seguir os seguintes passos:

a) iniciar a aplicação Gerador;
b) clicar no botão `Iniciar gravação`.

3.3.2.1.2 Parar gravação

Esta funcionalidade permite ao usuário parar a gravação do vídeo em disco. O caso de uso criado para esta funcionalidade é o `UC02 – Parar gravação`. A Figura 25 demonstra a como parar a gravação de um vídeo.

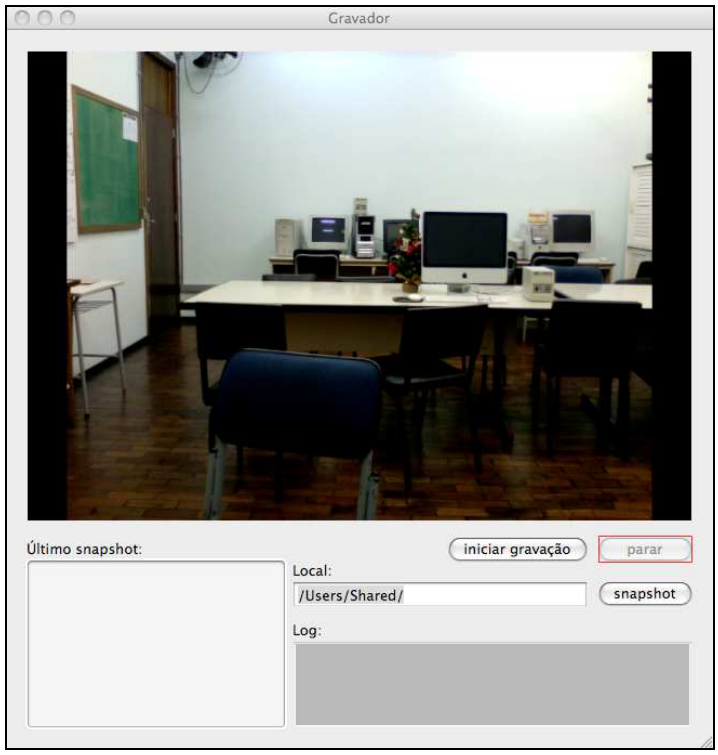


Figura 25 - Parar gravação

Para parar a gravação, o usuário deve seguir os seguintes passos:

a) iniciar a aplicação Gerador;

b) clicar no botão `Parar gravação`.

3.3.2.1.3 Tirar foto

Esta funcionalidade permite ao usuário tirar uma foto e gravar em disco. O caso de uso criado para esta funcionalidade é o `UC03 – Tirar foto`. A foto será exibida no campo `Último snapshot` e será gravada no local apontado pelo campo `Local`. A Figura 26 demonstra como tirar uma foto do local.

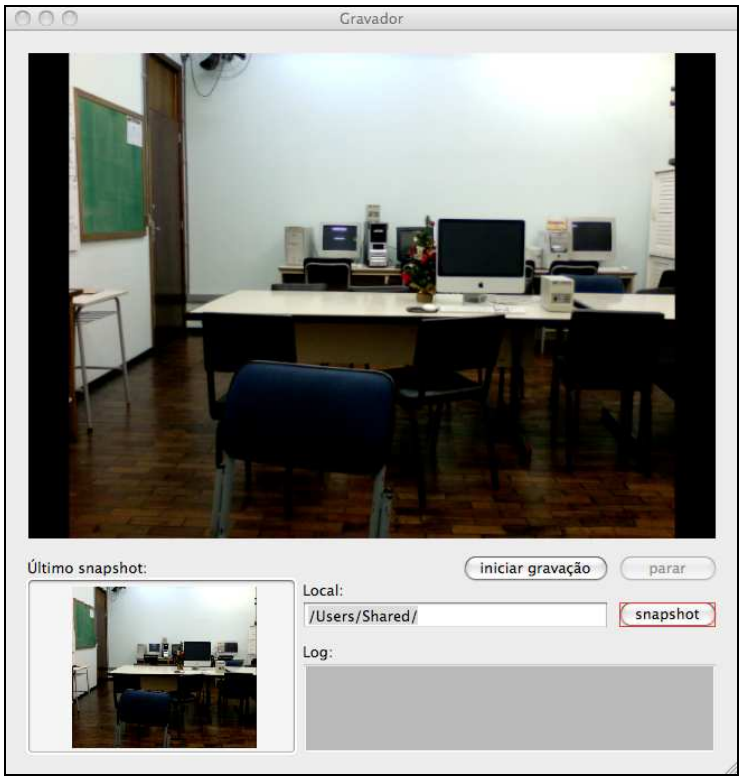


Figura 26 - Tirar foto

Para tirar uma foto, o usuário deve seguir os seguintes passos:

a) iniciar a aplicação Gerador;

b) clicar no botão `snapshot`.

3.3.2.2 Funcionalidades da aplicação Cliente

Nesta seção serão apresentadas as seguintes funcionalidades: `Adicionar local`, `Excluir local` e `Visualizar vídeo`.

3.3.2.2.1 Adicionar local

Esta funcionalidade permite ao usuário adicionar um local de monitoramento de imagens. O caso de uso criado para esta funcionalidade é o `UC01 – Cadastrar local`. A Figura 27 demonstra a tela para adicionar locais de monitoramento.

Figura 27 - Tela para adicionar locais de monitoramento

Para adicionar um local de monitoramento (nova *webcam*), o usuário deve seguir os seguintes passos:

a) iniciar a aplicação Cliente;

b) clicar no botão `+` (`mais`);

c) digitar o nome para o local no campo `Nome`;

d) digitar o endereço para o local no campo `Host`;

e) clicar no botão `Save`.

3.3.2.2.2 Excluir local

Esta funcionalidade permite ao usuário excluir um local de monitoramento de imagens previamente cadastrado. O caso de uso criado para esta funcionalidade é o `UC02 – Excluir local`. A Figura 28 demonstra a tela de exclusão de locais.

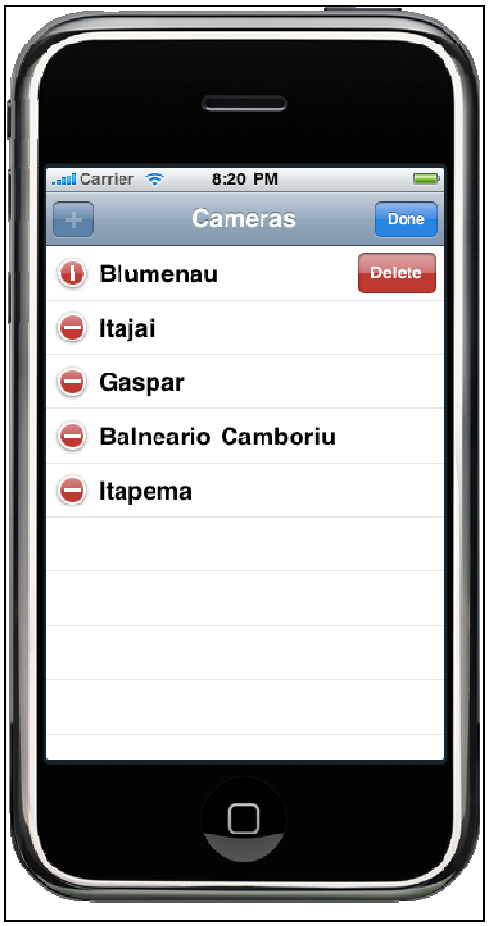


Figura 28 - Tela de exclusão de locais

Para excluir um local, o usuário deve seguir os seguintes passos:

a) iniciar a aplicação Cliente;

b) clicar no botão `Edit`;

c) clicar no botão `–` (`menos`);

d) clicar no botão `Delete`;

e) clicar no botão `Done`.

O procedimento `b` também pode ser feito através do movimento de arrastar o dedo para o lado direito.

3.3.2.2.3 Visualizar vídeo

Esta funcionalidade permite ao usuário visualizar um vídeo a partir de um local previamente cadastrado. O caso de uso criado para esta funcionalidade é o `UC03 – Visualizar vídeo`. A Figura 29 demonstra a tela com os locais disponíveis.

Figura 29 – Tela com locais disponíveis

Depois de selecionado o local, é aberta uma nova tela, onde são exibidos todos os vídeos gravados no local escolhido. Na parte superior, é exibida a imagem atual referente a este local. A Figura 30 demonstra a tela com os vídeos gravados para o local escolhido.

Figura 30 – Vídeos gravados

Para visualizar um vídeo, o usuário deve seguir os seguintes passos:

a) iniciar a aplicação Cliente;
b) clicar no local desejado;
c) clicar no vídeo desejado.

3.4 RESULTADOS E DISCUSSÃO

Nesta seção serão apresentados os resultados obtidos com o desenvolvimento do trabalho. O cenário proposto para os testes envolveu algumas estações servidoras e uma Cliente. O Quadro 25 apresenta os requisitos funcionais novamente, evidenciado os que foram concluídos com sucesso.

REQUISITOS FUNCIONAIS	CONCLUÍDO
RF01: Permitir ao usuário visualizar, a partir do iPhone, imagens geradas em computador com uma *webcam*.	Sim
RF02: Permitir ao usuário gravar e excluir os locais de acesso do iPhone.	Sim
RF03: Permitir que a aplicação geradora capture imagens instantâneas.	Sim
RF04: Deverá ser desenvolvido o programa gerador, responsável pela captura das imagens da câmera do computador.	Sim
RF05: Deverá ser desenvolvido o programa cliente, responsável por receber e exibir as imagens geradas pelo Gerador.	Sim

Quadro 25- Requisitos concluídos

Foi constatado que a performance das aplicações Gerador e Cliente está relacionado mais diretamente à dois fatores, sendo eles o tempo de processamento e o espaço ocupado em disco, já que o tempo de transmissão está relacionado apenas com a aplicação Cliente.

3.4.1 Aplicação Gerador

No Gerador, o propósito é capturar os vídeos oriundos de uma *webcam*, em um computador ou notebook, disponibilizando as mesmas para o Cliente efetuar a leitura, tendo-se como principal objetivo efetuar auditoria em imagens de uma aplicação de CFTV. Todos os testes foram realizados utilizando-se uma resolução de gravação com 300 *pixels* de altura e 240 *pixels* de largura, com profundidade de 32 *bits* de cores e compactação H.264.

Durante os testes realizados não foi percebido qualquer travamento na captura do vídeo, mas é importante evidenciar que não existia nenhuma aplicação em paralelo, salvo as rotinas básicas do sistema operacional instalado, sendo ele Mac OS X, versão 10.5.8. O hardware utilizado possui a seguinte descrição básica:

a) computador *desktop* modelo iMac;
b) processador Intel Core 2 Duo 2.4 Gigahertz, com 6 Megabytes de cache L2;
c) um gigabyte de memória RAM DDR2 800Mhz
d) placa de vídeo ATI Radeon HD 2400 PCIe x16, com 128 Megabytes de memória RAM;
e) resolução de vídeo 1680 x 1050, com profundidade de 32 *bits* de cores;
f) disco rígido de 250 Gigabytes;

g) câmera *iSight* (embutida), de resolução 640 *pixels* de altura e 480 *pixels* de largura, com possibilidade de captura de 30 frames por segundo (FPS).

Em relação aos fatores testados, o tempo de processamento nas máquinas com o hardware especificado acima, as ações de `iniciar gravação`, `parar gravação` e `snapshot` (bater foto) foram quase instantâneas, ou seja, não se percebeu nenhum atraso nos processos internos, nem tão pouco na interação do usuário com as opções da aplicação Gerador.

O outro fator, espaço em disco utilizado, foi testado gravando-se os vídeos em diferentes faixas de tempo, a fim de se concluir qual seria a mais adequada para utilização, sem que se tenha perda de desempenho. O Quadro 26 apresenta as diferentes faixas de tempo utilizadas e seu tamanho ocupado em disco. Vale lembrar que os vídeos foram gravados utilizando-se uma resolução de 300 *pixels* de altura e 240 *pixels* de largura, com a codificação H.264, para que o iPhone pudesse efetuar a leitura.

TEMPO	TAMANHO APROXIMADO
20 segundos	580 Kb
65 segundos	2.1 Mb
90 segundos	2.9 Mb
02 minutos	3.4 Mb
04 minutos	9.4 Mb
22 minutos	45.2 Mb

Quadro 26 - Faixas de tempo e tamanho do vídeo

Depois de efetuados os testes acima, optou-se arbitrariamente pela gravação dos vídeos com tempo de 20 segundos. Na aplicação Gerador não se teve problemas em questão à memória para executar as rotinas de gravação e visualização, sendo a única preocupação a quantidade disponível de espaço em disco.

Entre as ações permitidas pela aplicação Gerador, sendo elas `iniciar gravação` e `parar gravação`, já explicadas acima, tem-se ainda que testar a ação de fotografar. As fotos são gravadas pelo Gerador com a extensão ".jpg", conhecida como *Joint Photographic Experts Group*. Optou-se por este formato pois o tamanho do arquivo é consideravelmente menor em relação aos outros formatos de arquivos disponíveis na linguagem Objective-C, e, sendo um arquivo de menor tamanho, reflete diretamente numa transmissão mais rápida. As imagens foram comparadas visualmente, não se constatando perda de qualidade, o que é válido para uma aplicação CFTV, uma vez que a qualidade necessária se restringe a poder interpretar o conteúdo da imagem. O Quadro 27 demonstra a comparação entre os formatos

disponíveis.

FORMATO	TAMANHO APROXIMADO
".bmp" – *Bitmap*	5.2 Mb
".gif" – *Graphics Interchange Format*	540 Kb
".jpg" – *Joint Photographic Experts Group*	252 Kb
".png" – *Portable Network Graphics*	2.2 Mb

Quadro 27 - Comparação entre os formatos disponíveis

3.4.2 Aplicação Cliente

Já a aplicação Cliente, o propósito era desenvolver uma aplicação na qual é possível ler os vídeos gravados pelo Gerador. Todos os testes na aplicação Cliente foram executados utilizando-se o emulador do iPhone versão 2.2.1, disponibilizado no ambiente XCode versão 3.1.3. Optou-se por utilizar o simulador, pois, para carregar uma aplicação no iPhone, se faz necessário possuir uma conta de desenvolvedor na loja virtual da Apple.

É importante destacar que no iPhone, apenas a aplicação Cliente estava sendo executada. Uma particularidade do iPhone, é que as aplicações que rodam nele não são executadas simultaneamente, o que torna a realização dos testes mais simples. Os mesmos fatores observados na aplicação Gerador foram observados na aplicação Cliente, ou seja, tempo de processamento e espaço em disco, além do tempo de transmissão.

Em relação ao tempo de transmissão, este é altamente influenciado pela rede utilizada, ou seja, está limitada à rede na qual o iPhone está conectado. O poder de processamento no iPhone é de 620 MHz, e, apesar deste poder de processamento ser um pouco menor em relação à aplicação Gerador, também não foram percebidos travamentos ou problemas na execução da aplicação. É importante destacar que o simulador do iPhone emula também a frequência do processador do mesmo. Tanto a ação de cadastrar um local de monitoramento (que envolvia rotinas de banco de dados), quanto à ação de ler os arquivos de vídeo deste local foram executadas sem problemas. Em relação à leitura dos arquivos de vídeo, é justificável não ter ocorrido nenhum problema, já que sua arquitetura e seus componentes de programação possuem todo um aperfeiçoamento decorrente dos aparelhos iPods.

No iPhone, o gerenciamento da memória é feito através das telas, na qual, para este trabalho, foram desenvolvidas quatro. Uma tela inicial, onde é realizada uma consulta ao

banco de dados e é exibida uma lista com os locais encontrados; uma tela onde são exibidos os vídeos cadastrados para um local previamente escolhido; outra tela onde são cadastrados novos locais de monitoramento, possuindo apenas dois campos, nome e endereço. E uma quarta tela, onde é possível efetuar a exclusão de locais cadastrados.

Os espaço ocupado em disco no iPhone foi de 4 Kb para o banco de dados (SQLite), sendo que este possui cinco registros (locais) cadastrados e, a aplicação em si ocupou um espaço em disco de 80 Kb.

4 CONCLUSÕES

Apesar de existirem atualmente trabalhos na área de vigilância com CFTV tendo como visualizadores os dispositivos móveis, o trabalho proposto foi desenvolvido tendo como base o iPhone.

Uma das principais motivações para que fosse realizado o trabalho é explorar a tecnologia que o iPhone dispõe, criando um receptor e visualizador de arquivos de vídeo, com possibilidade de ser usado em qualquer lugar, através da internet.

O trabalho foi desenvolvido em duas partes, uma aplicação chamada Gerador e outra chamada Cliente. O Gerador, que deve ser instalado em um computador *desktop*, pessoal ou *notebook*, com uma câmera instalada, foi desenvolvida uma aplicação responsável por capturar imagens e vídeos oriundos da câmera, gravando-os na máquina local, a fim de que o Cliente fosse capaz de efetuar a leitura e visualização dos mesmos, tendo-se como principal objetivo realizar auditoria em imagens de uma aplicação de CFTV.

Para o Cliente, que por sua vez deve ser instalado no iPhone, foi desenvolvido uma aplicação que efetua a leitura e visualização das imagens e vídeos gerados pelo Gerador, com a possibilidade de configuração de vários locais de monitoramento, gravando-os em banco de dados SQLite.

Já os trabalhos correlatos serviram como exemplo de aplicações disponíveis que estão em uso, facilitando por exemplo, a definição dos leiautes das telas deste trabalho.

Este trabalho pode também ser um ambiente de testes para a implementação do *streaming* de áudio e vídeo em tempo real, além da possibilidade de visualização de várias *webcams* ao mesmo tempo. Também pode ser desenvolvida uma aplicação que possua integração com sistemas de alarmes domésticos, onde se poderia utilizar sensores de movimento para detectar a abertura e/ou fechamento de janelas e portões, sendo enviadas mensagens do Gerador para Cliente.

Como material de apoio, além das informações dos trabalhos correlatos, foram utilizadas as informações disponíveis no *site* do iPhone e da Apple sobre a linguagem de programação Objective-C e suas ferramentas, o XCode (para codificação) e o Interface Builder (para criação das telas).

4.1 EXTENSÕES

Durante o desenvolvimento do trabalho, foram verificadas diversas possíveis melhorias para o mesmo, tanto na aplicação Gerador quanto na aplicação Cliente, que não puderam ser contempladas neste trabalho. O Quadro 28 demonstra as sugestões para extensão deste trabalho.

TAREFA
Permitir à aplicação Cliente fotografar os locais de monitoramento, armazenando as imagens no iPhone.
Permitir que a aplicação Gerador suporte várias câmeras ao mesmo tempo e em tempo real.
Criar recursos de integração entre a aplicação Gerador com o sistema de alarme doméstico, detectando a abertura e/ou fechamento de portas e janelas, enviando alertas à aplicação Cliente.
Criar mecanismo de troca de mensagens entre a aplicação Cliente e Gerador, a fim de implementar maiores controles sobre as mesmas.
Permitir à aplicação Gerador enviar *streaming* em tempo real para a aplicação Cliente.
Permitir que as aplicações Gerador e Cliente ofereçam recursos de gravação de vídeo com áudio.

Quadro 28 - Lista de tarefas

REFERÊNCIAS BIBLIOGRÁFICAS

3G. **Redes 3G**. [S.l.], 2009. Disponível em: <http://pt.wikipedia.org/wiki/3G>. Acesso em: 31 mar. 2009.

APPLE. **iPhone – recursos**. [S.l.], 2009. Disponível em: <http://www.apple.com/br/iphone/features>. Acesso em: 31 mar. 2009.

APPSHOPPER. **CAVU free vídeo surveillance**. [S.l.], 2009. Disponível em: <http://appshopper.com/utilities/cavu-free-video-surveillance>. Acesso em: 20 ago. 2009.

CAMCONTROL. **CamControl iPhone**. [S.l.], 2009. Disponível em: <http://www.heitel.com/en/products/software/camcontrol-iphone>. Acesso em: 07 nov. 2009.

CAVU. **Cavu**. [S.l.], 2009. Disponível em: <http://www.cavu.me>. Acesso em: 07 nov. 2009.

COMO TUDO FUNCIONA. **Como funciona a rede WiFi**. [S.l.], 2009. Disponível em: <http://informatica.hsw.uol.com.br/rede-wifi.htm>. Acesso em: 29 mar. 2009.

DEVMEDIA. **Devmedia – SQL Magazine**. [S.l.], 2009. Disponível em: <http://www.devmedia.com.br/articles/viewcomp_forprint.asp?comp=7100>. Acesso em: 29 mar. 2009.

DICWEB. **Dicionário de informática**. [S.l.], 2008. Disponível em: <http://www.dicweb.com/ss.htm>. Acesso em: 19 mar. 2009.

DIVX. [S.l.], 2009. Disponível em: <http://www.divx.com/pt-br/technologies/h264>. Acesso em: 31 out. 2009.

GSTECH. **Integrador de sistemas**. [S.l.], 2009. Disponível em: <http://www.gstech.com.br>. Acesso em: 29 mar. 2009.

HORUS. **Monitoração 24hs**. [S.l.], 2009. Disponível em: <http://muar.sytes.net/www.horus24horas.com/index.asp>. Acesso em: 29 mar. 2009.

INTERFACE BUILDER. **Interface Builder**. [S.l.], 2009. Disponível em: <http://developer.apple.com/tools/interfacebuilder.html>. Acesso em: 08 nov. 2009.

ITUNES STORE. **iTunes Store**. [S.l.], 2009. Disponível em: <http://www.apple.com/itunes>. Acesso em: 07 nov. 2009.

JUMICAM. **Jumicam – Stream PC-webcam to your iPhone**. [S.l.], 2009. Disponível em: <http://www.jumitech.com>. Acesso em: 07 nov. 2009.

KATAOKA, K. **3G**. [S.l.], 2008.Disponível em: <http://disciplinas.dcc.ufba.br/svn/MATA85/2008.1/apresentacoes/3G_Karina.ppt?revision=11&content-type=text%2Fplain&pathrev=11>. Acesso em: 28 mar. 2009.

MARTINS, A. **Objective-C**. [S.l.], 2005. Disponível em: <http://www.astro.iag.usp.br/~algol/computacao/ObjCtutorial.html>. Acesso em: 25 mar. 2009.

MORAES, R. F. **Sistemas de CFTV (Circuito Fechado de Televisão)**: seu funcionamento e sua manutenção. [S.l.], 2006, Recife. Disponível em: <http://www.poli.br/arquivos/DOWNLOADS/RELAT%D3RIO%20DE%20ESTAGIO/ELETRONICA/Rebeca%20Ferreira/Relatorio%20Final.pdf>. Acesso em: 24 mar. 2009.

NEXTVIEW. **Nextview remote video camera surveillance**. [S.l.], 2009. Disponível em: <http://nextviewcam.com>. Acesso em: 07 nov. 2009.

OLIVEIRA FILHO, K. S. **Fundamentos de radiodiagnóstico por imagem**. [S.l.], 1999. Disponível em: <http://www.if.ufrgs.br/ast/med/imagens/node14.htm>. Acesso em: 25 mar. 2009.

PÉRICAS, F. A. **Redes de computadores**: conceitos e a arquitetura Internet. Blumenau: Edifurb, 2003. 158 p.

PROCURADORIA REGIONAL DO TRABALHO 19ª REGIÃO. **Projeto Circuito Fechado de TV - CFTV**. [S.l.], 2006, Maceió. Disponível em: <http://www.prt19.mpt.gov.br/paginas/PHP/PROJETO_CFTV.pdf>. Acesso em: 10 mar. 2009.

QTKIT CAPTURE. **QTKit application programming guide:** QTKit capture. [S.l.], 2009. Disponível em: <http://developer.apple.com/mac/library/documentation/Cocoa/Conceptual/QTKitApplicationProgrammingGuide/UsingQTKit/UsingQTKit.html >. Acesso em: 07 nov. 2009.

SCRÓCARO, R.; SATURINO D. **Meios de transmissão de dados**. [S.l.], 2005. Disponível em: <http://www2.dc.uel.br/~sakuray/Espec-Comunicacao%20de%20dados/R%A3bia%20-%20Danilo/Meios%20de%20transmiss%92o%20de%20dados%20nas%20redes.htm>. Acesso em: 25 mar. 2009.

SQLITE. [S.l.], [2009]. Disponível em: <http://www.sqlite.org/>. Acesso em: 07 nov. 2009.

SQLITEBR. **SQLite Brasil**. [S.l.], [2009]. Disponível em: <http://www.dicas-l.com.br/dicas-l/20050212.php>. Acesso em: 07 nov. 2009.

STREAMING MEDIA GUIDE. [S.l.], 2009. Disponível em: <http://developer.apple.com/iphone/library/documentation/NetworkingInternet/Conceptual/StreamingMediaGuide/HTTPStreamingArchitecture/HTTPStreamingArchitecture.html>. Acesso em: 07 nov. 2009.

XCODE. [S.l.], 2009. Disponível em: <http://developer.apple.com/mac/library/documentation/DeveloperTools/Conceptual/A_Tour_of_Xcode/000-Introduction/qt_intro.html>. Acesso em: 08 nov. 2009.

WEBCLAUDIO. **Curso de programação para iPhone**. [S.l.], 2009. Disponível em: <http://webclaudio.wordpress.com/2009/02/01/curso-de-programacao-para-iphone-parte-1>. Acesso em: 31 mar. 2009.

YOICS. **Yoics, secure, instant remote access and management**. [S.l.], 2009. Disponível em: <http://www.yoics.com>. Acesso em: 29 mar. 2009.