



# Visualizador de imagens radiológicas 2D para iPhone

Acadêmico: Marwin Roepke

Orientador: Dalton Solano dos Reis



# ROTEIRO

- ◆ Introdução / Objetivos
- ◆ Fundamentação teórica
  - Tecnologias
  - Trabalhos correlatos
- ◆ Desenvolvimento
  - Principais requisitos
  - Implementação
  - Resultados e discussão
- ◆ Conclusão / Extensões

# INTRODUÇÃO

- ◆ Estudar o padrão de imagens radiológicas no padrão DICOM
- ◆ Imagens radiológicas são geradas pelos tomógrafos
- ◆ Imagens JPEG obtidas das imagens DICOM
- ◆ Radiografias em um iPhone, ajudando no prontuário

# OBJETIVOS DO TRABALHO

- ◆ Visualizar imagens radiológicas 2D em iPhone
- ◆ Efetuar alteração de zoom, brilho e contraste nas imagens apresentadas, bem como a marcação de regiões da imagem
- ◆ Buscar as imagens através da rede 3G e wireless



# FUNDAMENTAÇÃO TEÓRICA

# LINGUAGEM DE PROGRAMAÇÃO PARA IPHONE

- ◆ Lançamento no ano de 2007
- ◆ Criada por Brad Cox na StepStone Corporation
- ◆ Caracterizada conjunto de extensões do padrão de linguagem ANSI C
- ◆ Baseada no Smalltalk

# IMAGENS RADIOLÓGICAS

- ◆ Feixes em forma de leques que atravessam uma seção
- ◆ Recebido por arranjo circular de detectores
- ◆ Calculado um valor dentro de uma escala de tons cinza para cada pixel do corte
- ◆ As imagens são geradas para o padrão DICOM desde 1985
- ◆ Padronizadas pela NEMA e pela ACR

# VISUALIZAÇÃO DE IMAGENS - JPEG

- ◆ Os objetos 3D tratados considerar como fossem objetos 2D
- ◆ Imagem 3D utilizando várias imagens 2D, como JPEG
- ◆ JPEG compressão de imagens existam perdas de informações toleráveis





# **TRABALHOS CORRELATOS**

# CONVERSÃO DE IMAGENS DO FORMATO DICOM VISANDO A INTER-OPERACIONALIDADE DE SISTEMAS ATRAVÉS DA WEB

- ◆ Visualizar as imagens radiológicas num *browser*
- ◆ Não houve o atendimento da flexibilidade exigida para a utilização em prontuários

# ANÁLISE DE IMAGENS RADIOLÓGICAS VIA DISPOSITIVOS MÓVEIS

- ◆ Um sistema distribuído de análises de imagens médicas para a web, utilizando *web service*
- ◆ Implementa a conversão das imagens de DICOM para JPEG e clientes para PDA e celular

# OSIRIX

- ◆ Aplicação para estações Mac OS X, código livre
- ◆ Permite a navegação e visualização de imagens
- ◆ Modos visualizar em: 2D, 3D, 4D e 5D
- ◆ Em novembro de 2008 conta com uma versão para o iPhone paga.

# Osirix Mac OS X

Local Database (/Users/marwinroepke/Documents/OsiriX Data/Database.sql)

Viewers Import Export CD-Rom Email Notification Add Studies Movie Export Query Send Anonymize Burn Meta-Data Delete 2D Viewer ROIs 4D Viewer Albums & Sources Report

Local Database / No album selected / Result = 2 studies (520 images) Added: <1 mn to the database  
Acquired: <10 mn <1 hr <4 hr

Patient name	Report	Lock	Patient ID	Age	Accession Number	Study Description	Modality	ID	Comme
► q5mT3d2KZyoe0nvSL (2 series)		-	fyET5.0		1657271	Neck 1HEA...ECK_PETCT	CT\PT	39...225	
▼ P.cBrdG0VM7Y4d (1 series)		-	7edw,Q6	62 y	1417966	Specials 1...al_CTA_pre	CT	38...151	
CorCTA w-c 1.0 B20f						1CoronaryCT...iral_CTA_pre	CT	6	

CorCTA w-c 1.0  
354 Images

Image size: 512 x 512  
View size: 672 x 304

7edw,Q6 ( 62 y, 56 y)  
1CoronaryCTA\_with\_spiral\_CTA\_pre -- CorCTA w-c 1.0 B20f  
3813151  
6

Uncompressed  
Thickness: 1.00 mm Location: -125.00 mm

12/12/03 09:33:42  
Made In OsiriX

Auto-play

# Osirix iPhone





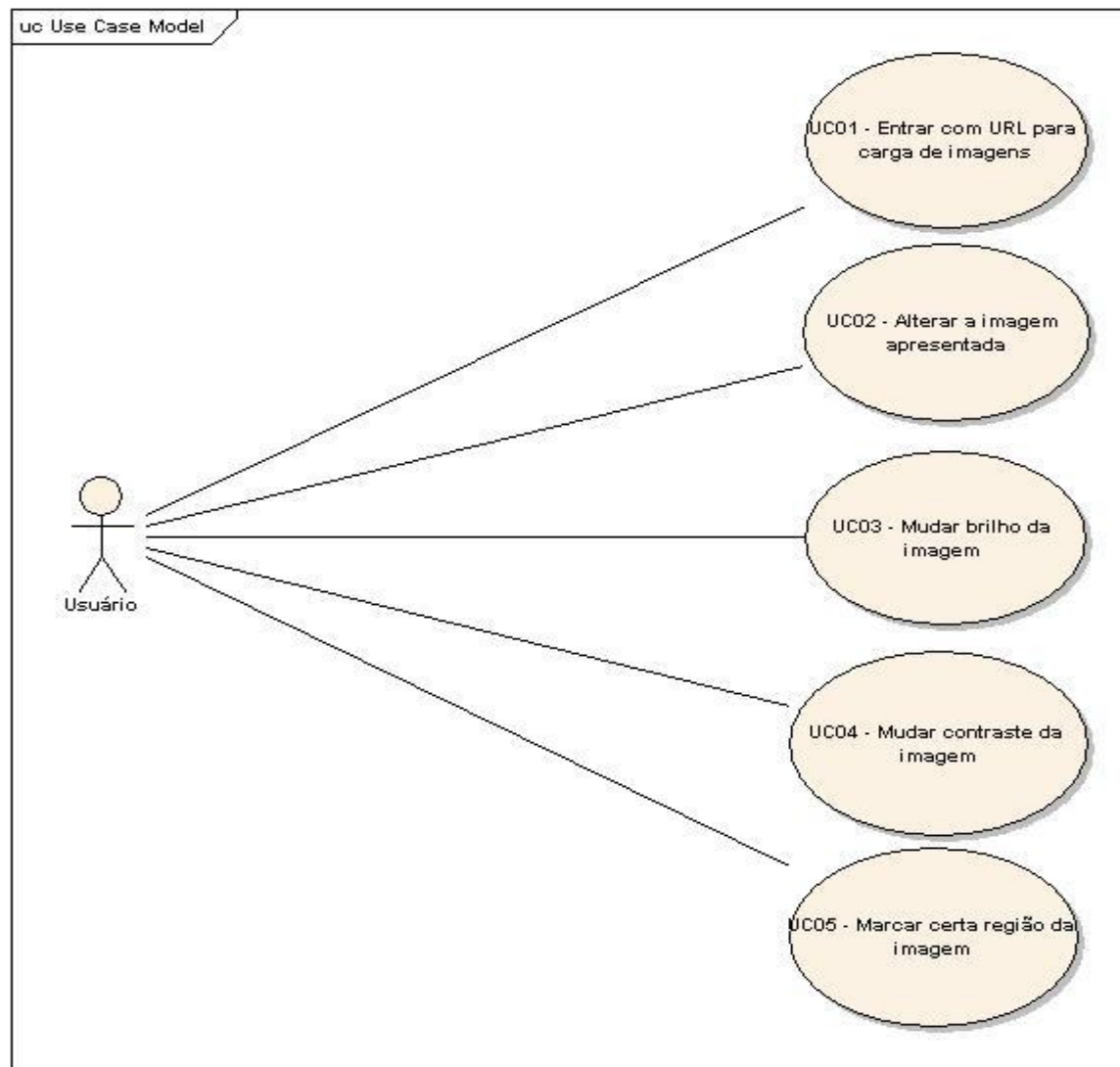
# **DESENVOLVIMENTO**

# REQUISITOS

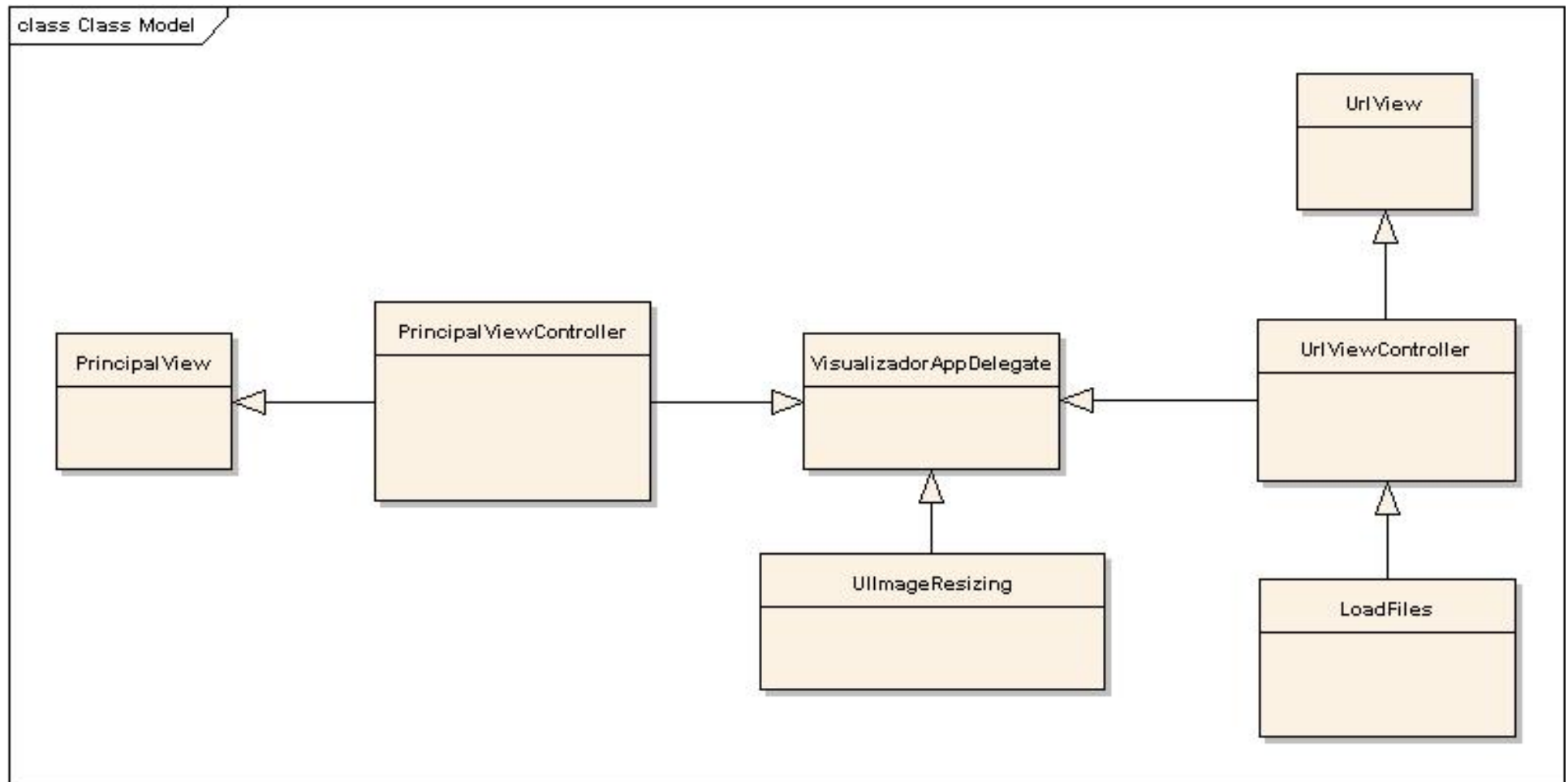
- ◆ O sistema deverá utilizar o multitoque para dar *zoom* as imagens (RF)
- ◆ O sistema deverá permitir adicionar brilho e contraste as imagens (RF)
- ◆ O sistema deverá permitir inserir marcações na imagem (RF)
- ◆ O sistema deverá utilizar o acelerômetro para retirar as marcações na tela (RF)
- ◆ O sistema deverá acessar imagens utilizando a rede 3G e/ou wireless (RNF)
- ◆ O sistema deverá ser implementado na linguagem Objective-C (RNF)



# DIAGRAMA DE CASOS DE USO



# DIAGRAMA DE CLASSES



# IMPLEMENTAÇÃO - AJUSTE DE ESCALA DAS IMAGENS

Definido um contexto de imagem

Altera a escala do sistema de coordenadas.

Desenhado a imagem

Atribuído a um objeto UIImage para ser retornado a quem chamou.

```
14 - (UIImage*)scaleToSize:(CGSize)size {
15     UIGraphicsBeginImageContext(size);
16
17     CGContextRef context = UIGraphicsGetCurrentContext();
18     CGContextTranslateCTM(context, 0.0, size.height);
19     CGContextScaleCTM(context, 1.0, -1.0);
20
21     CGContextDrawImage(context, CGRectMake(0.0f, 0.0f, size.width, size.height), self.CGImage);
22
23     UIImage* scaledImage = UIGraphicsGetImageFromCurrentImageContext();
24
25     UIGraphicsEndImageContext();
26
27     return scaledImage;
28 }
```

# IMPLEMENTAÇÃO - TRANSIÇÃO DAS IMAGENS

```
245 -(void)performTransition
246 {
247     // Primeiro crie um objeto CATransition
248     CATransition *transition = [CATransition new];
249     // Anima por 1 segundo
250     transition.duration = 1;
251     // usando a função timingFunctionDefault
252     transition.timingFunction = [CAMediaTimingFunction new];
253
254     // Seto o tipo de transição
255     transition.type = kCATransitionCrossfade;
256
257     // Finalmente, para evitar a animação de nós mesmos, nós atribuímos a nós mesmos a delegação para a animação e
258     // esperamos a animação terminar antes de executar a transição.
259     transition.delegate = self;
260     transition.duration = YES;
261
262     // Em seguida adicionamos a animação ao conteúdo.
263     [containerView.layer addAnimation:transition forKey:@""];
264
265     // Aqui nós temos a view1 e outra a View2
266
267     view01.hidden = YES;
268     view02.hidden = NO;
269
270     // E assim que vamos continuar a trocar entre as duas imagens, trocamos as variáveis de instância.
271     UIImageView *tmp = view02;
272     view02 = view01;
273     view01 = tmp;
274 }
```

Instanciado um objeto para efetuar as transições.

Definimos a delegação do objeto para controlar a animação

Definimos qual objeto é o delegado

Efetua a troca das imagens nas views

# IMPLEMENTAÇÃO - EFEITOS DE BRILHO

Um buffer de bytes,  
para armazenar os  
pixels da imagem

representação de  
bimap.

Laço que efetuará a  
alteração do RGB  
para a cor branca.

```
85     CGImageRef inImage=
86
87     CGContextRef ctx;
88
89     CFDataRef m_DataRef;
90     m_DataRef = CGDataProviderCopyData(CGImageGetDataProvider(inImage));
91     UInt8 * m_PixelBuf = (UInt8 *) CFDataGetBytePtr(m_DataRef);
92     int length = CFDataGetLength(m_DataRef);
93     CGImageGetBitsPerComponent(inImage), CGImageGetBytesPerRow(inImage);
94
95     for (int index = 0; index < length; index += 4)
96     {
97         Byte tempR = m_PixelBuf[index + 1];
98         Byte tempG = m_PixelBuf[index + 2];
99         Byte tempB = m_PixelBuf[index + 3];
100
101         int outputRed = level + tempR;
102         int outputGreen = level + tempG;
103         int outputBlue = level + tempB;
104
105         if (outputRed>255) outputRed=255;
106         if (outputGreen>255) outputGreen=255;
107         if (outputBlue>255) outputBlue=255;
```

# VISUALIZAÇÃO DE BRILHO





# IMPLEMENTAÇÃO – MARCAÇÃO NAS IMAGENS

```
64 - (void)touchesEnded:(NSSet *)touches withEvent:(UIEvent *)event {
65     if (efeito ==4) {
66         if ((firstTouch.x > 0) && (firstTouch.y > 0)) {
67             //Limpa a tela caso aja qualquer figura
68             int valor = (int)round([mySlider value]);
69             view01.image = [[imagens objectAtIndex:valor] image];
70             view01.image = [self brightness:[self brightness] image];
71             UIGraphicsBeginImageContext(self.view.frame.size);
72             [view01.image drawInRect:CGRectMake(0, 0, self.view.frame.size.width,
73             height)];
74             //Seleciona a cor vermelha para a linha
75             CGContextSetRGBStrokeColor(UIGraphicsGetCurrentContext(), 1, 0, 0, 1);
76             //Desenha a Elipse
77             CGContextAddEllipseInRect(UIGraphicsGetCurrentContext(), CGRectMake(
78             *100 / (320*100), firstTouch.x, lastTouch.x, lastTouch.y));
79             CGContextStrokePath(UIGraphicsGetCurrentContext());
80             view01.image = UIGraphicsGetImageFromCurrentImageContext();
81             UIGraphicsEndImageContext();
82             [view01 removeFromSuperview];
83             [containerView addSubview:view01];
84         }
85         firstTouch.x = -1;
86         firstTouch.y = -1;
87         lastTouch.x = -1;
88         lastTouch.y = -1;
89     }
90 }
```

Alteramos os dados de toque para -1.

Definimos um   
 base na   
 Definimos   
 Desenhamos a Elipse   
 com os dados dos   
 event

Atribuimos a imagem alterada para a view mostrada ao usuário.

# VISUALIZAÇÃO - MARCAÇÃO NAS IMAGENS





# IMPLEMENTAÇÃO – ENVIO E-MAIL

```
406 -(void)displayComposerSheet
407 {
408     MFMailComposeViewController *picker = [[MFMailComposeViewController alloc] init];
409     picker.mailComposeDelegate = self;
410
411     [picker setSubject:@"Prontuario"];
412
413     // Attach an image to the email
414     //Busca a imagem da tela
415     UIImage *picture = [view01.imageView image];
416
417     NSData *myData = UIImageJPEGRepresentation(picture, 0.8);
418
419     [picker addAttachmentData:myData mimeType:@"image/jpeg" fileName:@"anexo.png"];
420
421     [picker addAttachmentData:myData mimeType:@"image/jpeg" fileName:@"anexo.png"];
422
423     // Fill out the email body text
424     NSString *emailBody = [NSString stringWithFormat:@"%s", area.text];
425
426     [picker setMessageBody:emailBody isHTML:NO];
427
428     [self presentViewController:picker animated:YES];
429     [picker release];
430 }
```

Definimos o assunto

Instanciado um objeto para mandar e-mail.

Buscamos a imagem da tela e adicionamos a ela um anexo

Definimos o corpo do e-mail

Apresentada a tela de e-mail ao usuário.

# VISUALIZAÇÃO – ENVIO E-MAIL



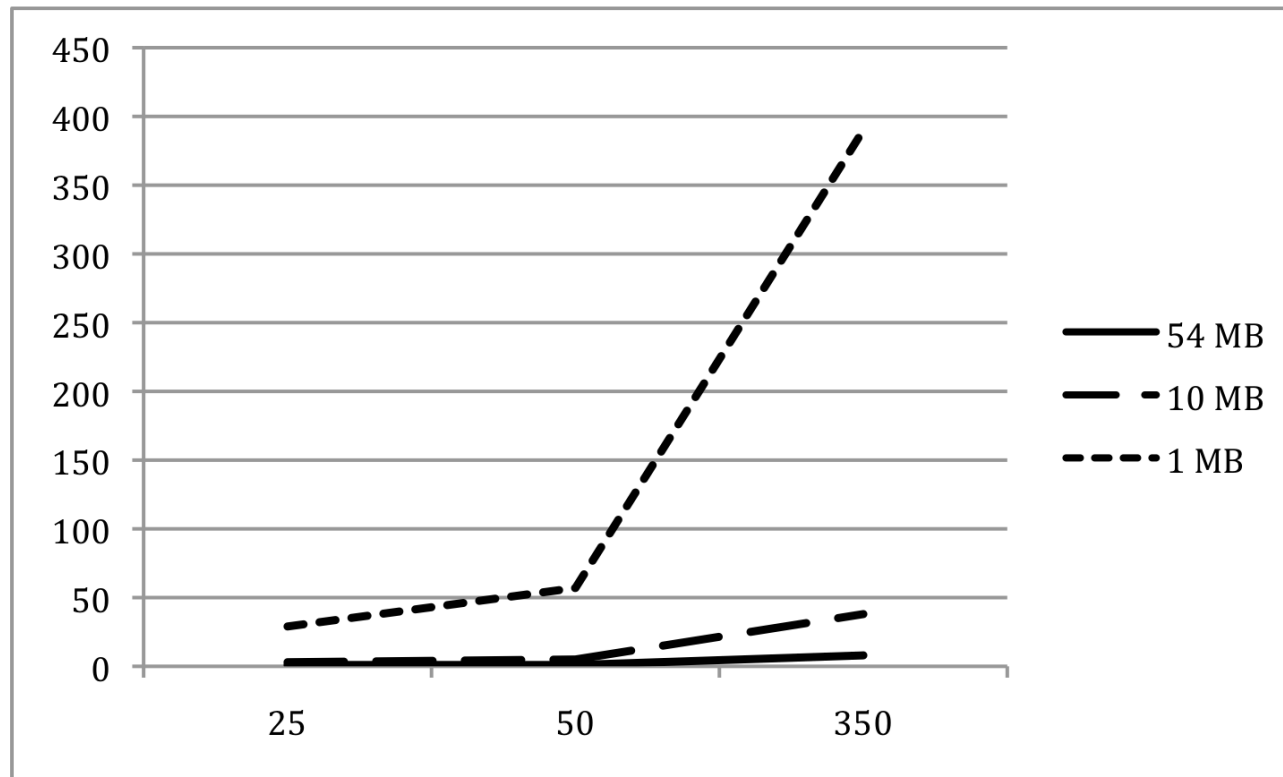
# RESULTADOS E DISCUSSÕES

- ◆ Tamanho das imagens carregadas
- ◆ Relação tempo de carga VS velocidade conexão
- ◆ Utilização de memória da aplicação
- ◆ Migrar código desktop do Osirix para iPhone

# TAMANHO DAS IMAGENS CARREGADAS

Quantidade de Imagens	Velocidade de Conexão	Tamanho Total	Tempo de Carga
25	1 MB	1,3 MB	29 segundos
50		2,6 MB	57 segundos
350		18,5 MB	6 minutos e 30 segundos
25	10 MB	1,3 MB	3 segundos
50		2,6 MB	5 segundos
350		18,5 MB	38 segundos
25	54 MB	1,3 MB	1 segundos
50		2,6 MB	1 segundos
350		18,5 MB	8 segundos

# RELAÇÃO TEMPO DE CARGA VS VELOCIDADE CONEXÃO



# UTILIZAÇÃO DE MEMÓRIA DA APLICAÇÃO

Quantidade de Imagens	Memória Real	Memória Virtual	Tamanho total das imagens	Memória Real	Memória Virtual
25	4,17 MB	53,53 MB	1,3 MB	9,45 MB	69,57 MB
50			2,6 MB	9,74 MB	69,74 MB
350			18,5 MB	15,68 MB	86,58 MB

# CONCLUSÃO

- ◆ Disponibilidade de uma forma mais prática e ágil a visualização de imagens radiológicas no formato JPEG
- ◆ Explorou a plataforma do desenvolvimento Objective-C para iPhone, com a integração de recursos como: acelerômetro, multitoque e transmissão Web
- ◆ Requisitos e objetivos em geral foram alcançados

# EXTENSÕES

- ◆ Utilizar as imagens no formato DICOM para visualização
- ◆ Implementar a persistência local dos arquivos
- ◆ Carregar as imagens em lotes
- ◆ Autenticação prévia do usuário para o acesso as imagens médicas
- ◆ Rotação das imagens usando DICOM





# **APRESENTAÇÃO PRÁTICA**



**Obrigado!**