Capítulo

2

Explorando HTML5, CSS3 e JQueryMobile no Controle e Monitoramento de Casas Inteligentes

José Antonio Camacho Guerrero[1] e Alessandra Alaniz Macedo[2]

[1]Innolution Sistemas de Informática Ltda. {jose.camacho@innolution.com.br}
[2]FFCLRP/Universidade de São Paulo, Campus Ribeirão Preto {ale.alaniz@usp.br}

Abstract

This chapter is about the use of recent web patterns (HTML5, CSS3, JQuery and JQueryMobile) to develop web applications in different scenarios. As a use case, these technologies are exploited in order to develop an application to domotics context. In the past, automatic environments were limited to big industries. Nowadays this scenario is rapidly changing. Home Control System (HCS) are became frequent and part of modern home. Users of HCS are looking for quality of life and sustainability. We are planning to present and discuss the following items: 1) Development of human computer interface of HCS using web: html5, CSS3, JQuery and JQueryMobile. 2) Concepts and components of HCS. 3) Examples of HCS and Primary Health System (PHS) 4) Use Cases.

Resumo

Neste capítulo, pretende-se apresentar e exemplificar o uso dos padrões mais recentes, HTML5, CSS3, JQuery e JQueryMobile, para o desenvolvimento de aplicações web em diferentes contextos. Como estudo de caso, essas tecnologias são exploradas na construção de um aplicativo para automação e criação de casas inteligentes. No passado, a automação se limitava a indústrias, porém hoje possui crescente e diversificado uso em espaços comunitários como shoppings, restaurantes e até mesmo em residências. Automação busca o aumento da qualidade de vida e a sustentabilidade por meio do uso eficiente de equipamentos, energia e serviços. Planeja-se apresentar e discutir os seguintes itens: (1) Desenvolvimento de interfaces de usuário para controle de hardware via web: HTML5, CSS, JQuery e JQueryMobile, (2) Conceitos e componentes físicos de automação, (3) Exemplos de automação residencial e para a área de saúde, e (4) Estudo de Caso para treinamento.

2.1 Introdução

Automação visa integração de funções de um sistema para que um dispositivo seja controlado. A cada dia, a automação está mais presente nos ambientes buscando aumento da qualidade de vida e sustentabilidade por meio do uso eficiente de equipamentos, energia e serviços. Há alguns anos, a automação se limitava a indústrias, porém hoje possui ampla utilização em residências, shoppings e outros tipos de instituições comunitárias.

Os sistemas de controle domésticos (*Home Control System* – HCS) estão se tornando mais comuns e parte integrante de habitações modernas. O controle computadorizado de alarmes, sistemas de climatização e outras aplicações para habitações são tecnologias que podem favorecer residências em todas as classes sociais. Algumas tecnologias, como os sistemas para controle de iluminação, estão presentes em casas, apartamentos e escritórios de médio e alto padrão, além de grandes empresas, teatros, hotéis e hospitais.

A área da saúde mostra-se necessitada da automação de processos, visto que a maioria dos locais de prestação de serviços e administrativos ainda realiza procedimentos de forma manual, dificultando o controle de dados e o gerenciamento de informações. Pesquisadores da Universidade de Hertfordshire, no Reino Unido, desenvolveram um projeto para cuidar de idosos, por meio da identificação de batimentos cardíacos e outros sinais. Sensores foram instalados para captar os sinais vitais e, em caso de alterações graves, são emitidos avisos para parentes e/ou centrais médicas[1]. No Brasil, agentes de saúde integrantes do programa APS (Atenção Primária a Saúde) utilizam um aplicativo PHCS (*Primary Health Care System*) instalado em um dispositivo móvel, sob a infraestrutura de rede, para realização de acompanhamento e ações preventivas em seus pacientes [Barros et. al, 2012].

Os vários ramos de automação têm em comum os mesmos princípios de controle, utilizando softwares e hardwares, controladores lógicos e linguagens de programação, além de diversos tipos de dispositivos sensores e de atuadores. Sob esse aspecto, o conceito de automação deve estabelecer condições para que todos os subsistemas envolvidos (controles de iluminação, segurança, ar condicionado, controle de energia, incêndio, etc) possam trabalhar em conjunto e de forma otimizada. Para isso é preciso conhecer todas as possíveis potencialidades de utilização nos lares e elencar as tecnologias disponíveis para aplicá-las. O fato de a web estar presente em diferentes hardwares fez com que esse ambiente se torne alvo de profissionais relacionadas com automação.

Durante muito tempo, a ideia de desenvolvimento web ficou exclusivamente associada à construção de páginas que ofereciam aos usuários o acesso a um determinado conteúdo. Desenvolvimento web era normalmente associado ao desenvolvimento de web sites, na internet ou intranet. Associado ao conteúdo, o desenvolvimento web pode ser usado para se referir ao projeto visual das páginas e ao desenvolvimento de comércio eletrônico. Porém, com a popularização da internet, novas necessidades foram surgindo em diversas áreas. As novas necessidades englobam

[1] www.nipponstudios.com.br/automacao-a-favor-da-saude

ferramentas de comunicação, emails, jogos, redes sociais, e-commerce, gerenciamento a distancia de segurança, automação e telemetria de indústrias, prédios e residências.

Liderado por Tim Berners-Lee, o W3C (Consórcio World Wide Web) foi criado com o intuito de definir padrões, protocolos e diretrizes para desenvolvimento de aplicações na web.

Neste capítulo, pretende-se apresentar e exemplificar o uso dos padrões mais recentes para o desenvolvimento de aplicações web: HTML5, CSS3, JQuery e JQueryMobile. Essas tecnologias auxiliam a visualização e o controle de informações estáticas e dinâmicas. Como estudo de caso, planeja-se discutir e aplicar tecnologias na construção de um aplicativo para controle de automação de residências.

2.2 Desenvolvimento de Interfaces de Usuário para Controle de Hardware via Web

Desde que foi concebida por Tim Bernes-Lee na década dos 1990, a web visava à disponibilização de informação na internet. No entanto, rapidamente se torno um meio para efetivar a comunicação, o comércio, o entretenimento, o monitoramento remoto de segurança e diversos outros serviços.

Para atender a essas necessidades, surgiram diversas abordagens utilizando HTML (*Hypertext Markup Language*) e linguagens de programação gerenciadas pelo servidor web, como Python, PHP, ASP entre outras.

Em 2002, o conceito de RIA (*Rich Internet Application*) foi introduzido pela Macromedia para denominar aplicações web com características e funcionalidades de softwares para desktop. A diferença entre HTML e linguagens gerenciadas pelo servidor é que as aplicações RIA transferem o processamento das interfaces para os navegadores no lado do cliente e mantém o conteúdo no servidor, proporcionando maior rapidez e possibilitando o uso de interfaces mais complexas. Soluções utilizando Applets em Java, Java EE, Adobe Flex, Microsoft Silverlight, VBScript e JavaScripts são exemplos de aplicações RIA .

Visando a criação de padrões, protocolos e diretrizes para a web foi criado o W3C (Consórcio World Wide Web) [W3C 2013]. Atualmente o W3C desenvolve especificações técnicas e orientações para aplicações web, arquitetura web, web Semântica, web Services, entre outros.

2.2.1 Linguagem HTML5

HTML (*Hypertext Markup Language*) é uma linguagem para estruturação e apresentação de conteúdo para web criada por Tim Bernes-Lee na década de 1990 visando a comunicação de resultados de pesquisas científicas.

Após a versão 4.01 do HTML, a W3C focou esforços em outra linguagem para web, o XHTML. O XHTML (eXtensible Hypertext Markup Language) é uma reformulação da linguagem de marcação HTML, baseada em XML a qual combina as tags de marcação HTML com regras da XML. O objetivo da W3C com este novo padrão era melhorar a acessibilidade de páginas web através de diversos dispositivos.

Enquanto o W3C trabalhava na segunda versão do XHTML, desenvolvedores das empresas Mozilla, Opera e Apple criaram o grupo WHATWG (*Web Hypertext Application Technology Working Group*) para dar continuidade ao padrão HTML com intuito de fornecer maior flexibilidade na produção de sistemas baseados na web [WHATWG 2013].

Em 2006, W3C e o WHATWG uniram esforços adicionando novos recursos e corrigindo eventuais problemas das versões de HTML dando origem à versão HTML5. Novas funcionalidades como semântica e acessibilidade, com novos recursos, antes só possíveis por meio de outras tecnologias (plugins proprietários e APIs), foram especificados em HTML5. A especificação para HTML5 também introduz marcações e interfaces de programação de aplicativos (APIs) para aplicações web complexas, projetadas para tornar mais fácil a inclusão e a manipulação de conteúdo gráfico e multimídia e para enriquecer o conteúdo semântico dos documentos web.

Na nova versão do HTML foi adicionado um padrão universal para a criação de seções comuns e específicas como rodapé, cabeçalho, sidebar, menus e etc. Também foi criado um padrão de nomenclatura de IDs, classes e tags. Elementos e atributos sofreram modificações na sua função e significado e que agora podem ser reutilizados de forma mais eficaz.

2.2.1.1 Por que HTML5?

O HTML5 permite criar aplicativos multiplataforma, capazes de rodar em uma variedade de dispositivos. Aplicativos em HTML5 além das funcionalidades de acesso remoto podem ser executados como aplicativos nativos permitindo a exibição de vídeo, música e animações. Adicionalmente permite armazenar dados locais, para que fiquem disponíveis mesmo quando não há acesso a internet.

Outro ponto importante para tornar o HMTL5 o padrão principal de desenvolvimento é que muitas companhias importantes participam do grupo de trabalho que definiu o HTML5 no W3C. A lista inclui fabricantes como Samsung, LG e Apple; operadoras como AT&T e France Telecom; produtoras de software como Microsoft, Adobe e Zynga; empresas de TI, como IBM e HP; e da internet, como Google e Netflix.

2.2.1.2 Compatibilidade

Atualmente, há uma grande diversidade de dispositivos pessoais desde o tradicional desktop até dispositivos portáteis como tablets e smartphones que permitem o acesso a web. Essa variedade de dispositivos traz como consequência o uso de uma variedade de navegadores para acessar a web e consequentemente problemas de compatibilidade. Uma forma segura para manter o código compatível é nivelar o desenvolvimento de acordo com os motores de renderização de cada navegador. Esses motores são responsáveis pelo processamento do código da página. Na Tabela 2.1 é apresentada uma lista de motores de navegação utilizados na grande maioria dos dispositivos.

O Webkit é o motor que suporta o maior número de especificações dos padrões do HTML5 e do CSS3. O Webkit foi criado pela Apple, de acordo com o motor de renderização de código aberto KHTML, presente em browsers para Linux, como o Konqueror. A Apple modificou o código, fazendo melhorias e aperfeiçoando o Webkit para criar o browser Safari. Contudo, outros motores já estão trabalhando para atender as novas especificações do HTML5 e do CSS3.

Tabela 2.1 Motores de Renderização

Motor	Browser
Webkit	Safari, Google Chrome
Gecko	Mozilla: SeaMonkey, Camino, Firefox, Thunderbird
Trident	Internet Explorer
Presto	Opera

Para solucionar problemas de incompatibilidade existem técnicas para verificar se o navegador suporta ou não os elementos do HTML5. É possível verificar se uma determinada propriedade existe em objetos globais como WINDOW ou NAVIGATOR. Outra maneira de detecção de compatibilidade é criar um elemento e consultar suas propriedades ou executar algum de seus métodos e avaliar seu retorno.

O Modernizr é uma compacta biblioteca Javascript, criada por Faruk Ates e Paul Irish. Essa biblioteca tem como principal finalidade verificar no navegador a presença de propriedades que permitam o suporte de tags HTML5 e CSS3. Para utilizar a biblioteca Modernizr é necessário fazer sua chamada no *head* do documento. O Modernizr roda automaticamente assim que é referenciado.

O código da Figura 2.1 mostra como verificar se o browser tem suporte da funcionalidade de vídeo utilizando a biblioteca Modernizr.

```
if (Modernizr.video) {
   alert(“Aceita”)
} else {
   alert(“Não Aceita”)
}
```

Figura 2.1 Verificação de compatibilidade utilizando Modernizr

2.2.1.3 Sintaxe e Estrutura do HTML5

A estrutura básica do HTML5 segue a mesma especificação que suas versões anteriores. Um documento é composto por elementos que possuem tags, atributos, valores e conteúdo. Cada elemento deve obrigatoriamente conter um tag, pode conter atributos e valores e seu conteúdo pode ser um texto simples ou outro elemento. Os elementos básicos de HTML, cuja presença é altamente recomendada nas páginas, são representados pelos seguintes tags:

- **<html>:** define o início de um documento HTML e indica ao navegador que todo conteúdo posterior deve ser tratado como uma série de códigos HTML. Este tag possui um atributo importante, o LANG, utilizado para que agentes identifiquem a linguagem principal do documento.
- **<head>:** é onde ficam os metadados que contém informações sobre a página e o conteúdo publicado.

- **<body>**: define o conteúdo principal, o corpo do documento. Esta é a parte do documento HTML que é exibida no navegador.

No tag **<head>** do cabeçalho, pode-se encontrar os seguintes elementos:

- **<title>:** define o título da página, que é exibido na barra de título dos navegadores.
- **<style type="text/css">:** define formatação em CSS. Recomenda-se o uso de arquivo separado para definir a formatação em CSS. Para isso, o arquivo na tag <**link**> é referenciado da seguinte forma: <*link rel="stylesheet" type="text/css" href=" login.css" />***.**
- **<script type="text/javascript">:** define programação de certas funções em página com scripts, podendo adicionar funções de JavaScript.
- **<link>:** define ligações da página com outros arquivos como feeds, CSS e scripts.
- **<meta>:** define propriedades da página, como codificação de caracteres, descrição da página, autor, etc. Um exemplo de uso do tag <meta> seria a declaração de tabela de caracteres utilizada, <meta charset=”utf-8” >.

```
<!DOCTYPE html>
 <html  lang =”pt-br”>
    <head >
          <meta charset =”UTF -8">
           <title >Exemplo da estrutura básica de um documento HTML </ title >
   </ head >
   <body >
       <p style="bottom:2%; left:80%" > Este é o elemento parágrafo</p>
   </ body >
</ html >
```

Figura 2.2 Exemplo de Estrutura do HMTL

A primeira observação do exemplo apresentado na Figura 2.2 diz respeito à declaração do Doctype. O Doctype não é um elemento do HTML, mas é uma instrução para que o browser tenha informações sobre qual versão de código a marcação foi escrita. O Doctype deve ser a primeira linha de código do documento antes do tag HTML, uma vez que indica para o navegador e para outros meios a especificação de código a ser utilizada. Em versões anteriores, era necessário referenciar o DTD diretamente no código do Doctype. Com o HTML5, a referência por qual DTD utilizar é responsabilidade do browser.

Na Figura 2.2 é utilizado o elemento principal HTML, representado pelo tag <HTML> a qual é uma série de elementos em uma árvore cujos elementos são filhos de outros. Essa árvore é denominada árvore DOM.

Os atributos de um elemento são definidos dentro da declaração do tag. No exemplo anterior, o tag <p> tem definido o atributo “style” com valor “bottom:2%;left:80%” e de conteúdo um texto simples: “Este é o elemento parágrafo”.

De acordo com a especificação do W3C, cada elemento tem um propósito específico que deve ser respeitado para a correta interpretação do conteúdo das páginas web. Nesta seção são apresentados os elementos, atributos e tipos mais utilizados no desenvolvimento de aplicativos web.

a) Modelos de Conteúdo

Os elementos no HTML podem ou não fazer parte de um grupo de elementos com características similares. Assim, foram criadas as seguintes categorias de elementos:

- **Metadata content:** Estes elementos estão antes do corpo da página, formando uma relação com o documento e seu conteúdo com outros documentos que distribuem informação por outros meios.
- **Flow content:** Estes elementos suportam o fluxo de informação. Fazem parte desta categoria todas as categorias seguintes.
- **Sectioning content:** Estes elementos definem um grupo de cabeçalhos e rodapés. Basicamente são elementos que juntam grupos de textos no documento.
- **Heading content:** Estes elementos se encarregam da apresentação de cabeçalhos, *h1, h2, h3, h4, h5, h6 e hgroup* e podem estar contidos na categoria Sectioning content**.**
- **Phrasing content:** Estes elementos permitem o tratamento de informação. Eles fazem parte da categoria de elementos que marcam o texto do documento, bem como os elementos que marcam um texto dentro do elemento de parágrafo.
- **Embedded content:** Estes elementos fazem referencia a objetos multimídia que importam outra fonte de informação, áudio, vídeo, canvas, imgens, objetos SVG, etc.
- **Interactive content:** Estes elementos permitem a interação com usuário, menus, botões, entradas, seleção, entre outros.

Dentre todas as categorias de modelos de conteúdo, existem dois tipos de elementos: elementos de linha e elementos de bloco. Os elementos de linha marcam, na sua maioria das vezes, texto, por exemplo, **a, strong, em, img, input, abbr, span**. Já os elementos de blocos dividem o conteúdo nas seções do layout como, por exemplo, **header, nav, article, aside, footer e address**.

Na Figura 2.3 é apresentado como se relacionam as categorias de elementos. Esse relacionamento permite identificar quais elementos podem conter outros elementos dependendo das categorias. Os elementos de linha podem conter outros elementos de linha. Entretanto, os elementos de linha nunca podem conter elementos de bloco.

Elementos de bloco sempre podem conter elementos de linha. Elementos de bloco podem conter elementos de bloco, dependendo da categoria que ele se encontra. Por exemplo, um parágrafo, elemento **p**, não pode conter um elemento **div**, mas o contrário é possível.

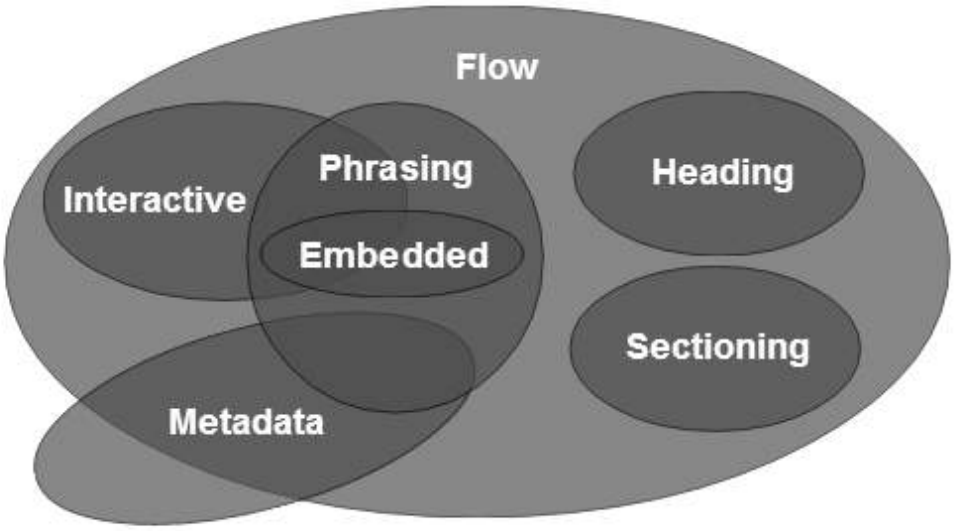

Figura 2.3 Relacionamento entre categorias de acordo com WHATWG

(fonte: http://www.whatwg.org/)

b) Elementos, Atributos e Tipos

Como apresentado na Seção 2.2.1.3, documentos HTML são formados por uma série de elementos aninhados os quais tem atributos com valores e conteúdo. Os elementos permitem que desenvolvedores estruturem a informação que será apresentada no navegador. Versões anteriores do HTML permitiam marcar diversos elementos do layout, estruturando a página de forma que as informações ficassem em suas áreas específicas, contudo careciam de significado semântico.

Na versão 5 do HTML, foram adicionados elementos e atributos que permitem diferenciar semanticamente áreas importantes do documento. No exemplo da Figura 2.4, é apresentado o código em HTML5 que define um formulário para cadastro de contatos. Nesse exemplo, foram utilizados elementos, atributos e tipos novos que auxiliam na segmentação, na apresentação e dão significado ao conteúdo da página.

O elemento **<header>** indica ao navegador que a palavra "Contatos" é um cabeçalho da página. Já o atributo **"placeholder"** utilizado nos elementos **<input>** permite mostrar ao usuário o *que* e *como* se espera que seja digitado o conteúdo desse campo. Também no elemento **<input>** do exemplo, o atributo **"type"** foi utilizado. Esse atributo existe em versões anteriores de HTML, mas na versão 5 são especificados novos valores que permitem comportamentos diferenciados para cada elemento, como por exemplo, *tel, search, email, url, color,* entre outros. Exemplos do uso desses novos tipos são apresentados nas seções seguintes.

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8" />
  <title>Exemplo HTML5- Testando novos elementos, atributos e tipos</title>
  <link href="style.css" rel="stylesheet" type="text/css" media="screen" />
 </head>
<body>
 <div id="contact">
  <header> <h1>Contatos</h1> </header>
   <form action="#" method="post">
            <fieldset>
              <label for="campoNome">Nome:</label>
              <input type="text" id="campoNome" placeholder="Digita teu nome" />
              <label for="campoEmail">Email:</label>
              <input type="email" id="campoEmail" placeholder="email@servidor"/>
              <label for="campoPhone">Phone:</label>
              <intput type=”tel” id=”campoPhone” placeholder=”(ddd)1234.5678”/>
              <input type="submit" value="Send message" />
            </fieldset>
  </form>
 </div>
</body>
</html>
```

Figura 2.4 Exemplo do uso de elementos HTML5

c) Novos Elementos, Atributos e Tipos.

Nesta seção são apresentados os elementos, atributos e tipos especificados na nova versão do HTML. Para cada elemento é descrito brevemente sua funcionalidade. As Figuras 2.5 e 2.6 apresentam exemplos dos elementos.

ELEMENTOS

section- permite dividir documento em segmentos ou seções. Por exemplo, uma loja virtual de música pode classificar seus álbuns por tipo de ritmo, rock, jazz, etc.

article representa uma parte da página que poderá ser distribuída e reutilizável em FEEDs, por exemplo. Isto pode ser um post, um artigo, um bloco de comentários de usuários ou apenas um bloco de texto comum.

aside- é um bloco de anotação para conteúdos de outros elementos próximos.

header- representa um grupo de introdução ou elementos de navegação. O elemento header pode ser utilizado para agrupar índices de conteúdos, campos de busca ou até mesmo logos.

hgroup- agrupa elementos de título (h1 até h6) organizando-os em uma hierarquia de títulos e subtítulos.

```
<section id=”rock”>
  <article>
    <hgroup>
       <h1>Titulo 1</h1>
       <aside>Este titulo faz parte do um album Teste de rock lançado em 2007<aside>
       <h2>Subtitulo 1</h2>
   </hgroup>
   <p>Descricao</p>
 </article>

 <article>
   <hgroup>
      <h1>Titulo 2</h1>
      <h2>Subtitulo 2</h2>
   </hgroup>
   <p>Descricao</p>
 </article>

   <!-- vários elementos article podem vir aqui -->
</section>

<!-- vários elementos section podem vir aqui -->
```

Figura 2.5 Exemplo do uso dos elementos section, article, aside, header e hgroup

footer- é o último elemento a ser apresentado no navegador antes de fechar a tag HTML. O elemento footer ou rodapé pode ser declarado em qualquer lugar do HTML não necessariamente no final de uma seção.

nav- representa uma seção da página que contém links e pode ser utilizado para criar um menu de navegação do site.

time- representa um horário ou uma data precisa no calendário gregoriano.

```
< footer  data-position="fixed">

  < nav class="siam-navbar">
      <a href="#inicio"    class="home-json-bt"  data-iconpos="notext"></a>
     <a href="#ambientes" class="envs-json-bt"  data-iconpos="notext"></a>
     <a href="#perfis"    class="perf-json-bt"  data-iconpos="notext"></a>
     <a href="#cameras"   class="cam-json-bt"   data-iconpos="notext"></a>
     <a href="#panico"    class="panic-json-bt" data-iconpos="notext"></a>
 </nav>

 <p>Publicado em <time> 2013-08-18</time> </p>

</footer>
```

Figura 2.6 Exemplo do uso dos elementos footer, nav e time do HTML5

ATRIBUTOS

Na nova versão HTML5, também foram especificados novos atributos para alguns elementos, auxiliando na interação e na apresentação. Os atributos que foram adicionados são:

autofocus- permite a seleção automática do elemento que especifica este atributo. Os elementos que possuem este atributo são *input, textarea, select* e *button*.

media- é um atributo já utilizado pelo elemento *link*, o qual permite especificar para qual media ou dispositivos o documento que aponta o link está otimizado.

novalidate- permite o envio de dados do elemento **form** sem validação.

reversed- é um atributo do elemento *ol* o qual permite apresentar a lista na ordem inversa.

disabled- já utilizado em outros elementos e agora pode ser utilizado no elemento **fieldset**, permitindo em uma única indicação a desativação de os filhos do **fieldset**.

placeholder- permite apresentar nos elementos de entrada, *inputs* e *textareas*, formatos ou dicas do conteúdo esperado para esse campo.

hreflang e rel- utilizado no elemento *link* em outras versões estes atributos podem ser utilizados no elemento **area.** Esses novos atributos permitem especificar o idioma e a relação do documento apontado pela url.

TIPOS

O elemento **input** do HTML, utilizado para solicitar informação que será inserida por usuários, foi estendido permitindo a formatação e a apresentação de seu conteúdo de forma mais representativa. Para permitir comportamentos diferenciados foram adicionados no elemento **input** os seguintes valores para o atributo **type:** *tel, search, email, url, color, datetime, date, month, week, time* e *datetime-local*, *number* e *range.*

Os tipos de *data e hora, number* e *range* permitem o uso do atributo **step** utilizado para modificar o conteúdo do elemento na quantidade definida. Exemplos de uso de tipos e do atributo **step** são apresentados respectivamente na Figura 2.7 e na Figura 2.8.

```
<!DOCTYPE html>
<html lang=”pt-BR”>
   <head>
        <meta charset=”UTF-8” />
        <title>Number type</title>
  </head>

  <body>
       <input name=”valuex” type=”number” value=”12.4” step=”0.2” min=”0” max=”20” />
       <input name=”valuex” type=”range” value=”12” step=”1” min=”0” max=”20” />
       <input type="date" name="data">
  </body>
</html>
```

Figura 2.7 Uso dos novos tipos do elemento input

Figura 2.8 Visualização dos elementos no navegador

2.2.1.4 Drag-n-Drop

A funcionalidade de arrastar e de soltar especificada no HTML5 é das mais importantes em termos de interação. O HTML5 permite por meio de atributos, tornar um objeto plausível de ser arrastado para outro elemento que permite objetos serem jogados dentro de sua abrangência. Ao adicionar o atributo **draggable=”true”** em um elemento, este pode ser arrastado para outro elemento.

A API Drag-n-Drop implementa eventos que podem ser monitorados e tratados, **dragstart, drag, dragend, dragenter, dragleave, dragover e drop.** Esses eventos permitem desenvolvedores implementar métodos que atuem em determinados estados de interação com as páginas web. Por exemplo, o evento **dragstart** será disparado imediatamente se um elemento **draggable** for arrastado.

Na Figura 2.9 são declarados dois elementos, o *img* e *div*. O primeiro dos elementos, na declaração do elemento *img* é adicionado o atributo **draggable=”true”,** permitindo que a imagem seja arrastada, e o atributo **ondragstart=”drag(event)”**, que avisa o navegador que quando seja detectado o evento **dragstart** execute a função drag.

O segundo elemento vai permitir que o objeto seja adicionado como conteúdo. Para isso foram criadas funções em JavaScript para serem executados quando seja detectado um evento **drop** nesse segundo elemento.

```
<html lang="en">
    <head>
        <title>Drag and Drop - Exemplo 1</title>
        <style type="text/css">
            #DivDestino { width:350px; height:70px; padding:10px; border:1px solid #aaaaaa; }
        </style>
        <script type="text/javascript">
            function allowDrop(ev){
                ev.preventDefault();
            }

            function drag(ev){
                ev.dataTransfer.setData("Text",ev.target.id);
            }

            function drop(ev){
                var data = ev.dataTransfer.getData("Text");
                ev.target.appendChild(document.getElementById(data));
                ev.preventDefault();
            }
        </script>
    </head>
    <body>
        <img id="Drag" src="logo.png" draggable="true" ondragstart="drag(event)" />
        <div id="DivDestino" ondrop="drop(event)" ondragover="allowDrop(event)"></div>
    </body>
</html>
```

Figura 2.9 Exemplo do uso de Drag-n-Drop

2.2.1.5 Elementos de Áudio, Vídeo e Canvas

O elemento de ***audio*** representa um stream de áudio a ser exibido. Atributos como, **src, type, preload, autoplay, loop e controls** podem ser utilizados neste elemento. No atributo **type** do elemento é definido o formato de áudio a ser utilizado e pode ter os valores *Ogg, MP3* e *Wav*. O atributo **preload** indica ao navegador como será carregado o arquivo de áudio ou vídeo quando a página carrega, completo, unicamente metadados ou só quando for acessado pelo usuário.

Para evitar erro de compatibilidade no navegador, pode-se codificar quatro opções, como ilustrado na Figura 2.10. Pode-se observar que a última alternativa será apresentada ao usuário, caso o navegador não suporte nenhum dos formatos de áudio.

```
<audio controls="controls"><br />
      <source src="http://server/audio.ogg" type="audio/ogg" /><br />
      <source src="http://server/audio.mp3" type="audio/mpeg" /><br />
      <source src="http://server/audio.wav" type="audio/wav" /><br />
      Você pode  baixar a musica <a href="http://server/audio.mp3">aqui</a><br />
</audio>
```

Figura 2.10 Exemplo de utilização de um player de áudio

O elemento ***video*** é um dos recursos de HTML5 que tem sido mais utilizado e bem aceito pelos usuários e desenvolvedores. A principal vantagem é a integração natural com os outros elementos e camadas como, por exemplo, CSS e JavaScript. Os formatos suportados são **webm**, **mp4** e **ogv**. Este elemento utiliza os mesmos atributos que o elemento **audio** e pode-se da mesma forma declarar todos os formatos para evitar erros de compatibilidade, ver exemplo na Figura 2.11.

```
<video>
   <source src="movie.mp4" type='video/mp4; codecs="avc1.42E01E, mp4a.40.2"'/>
   <source src="movie.webm" type='video/webm; codecs="vp8, vorbis"'/>
  <source src="movie.ogb" type='video/ogg; codecs="theora, vorbis"' />
  Você pode baixar o video <a href="http://server/video.mp4"> aqui</a><br />
</video>
```

Figura 2.11 Exemplo de tag <video>

2.2.1.6 HTML5 Avançado

Em HTML5 é possível fazer com que objetos interajam com outros objetos por meio de atributos, eventos ou interações com usuários. Nesta seção são ilustradas algumas possibilidades utilizando o elemento ***video*** apresentado anteriormente. Foi escolhido este elemento para demonstrar o impacto que as novas funcionalidades trazem ao desenvolvimento de páginas web. Em versões anteriores do HTML o uso de vídeos era muito complexo. Eles tinham que ser utilizados em applets externos e os codificadores tinham que escrever enormes funções e, JavaScript para interagir com os vídeos. Atualmente é possível manipular desde a apresentação até o conteúdo de um vídeo com comandos simples.

a) Utilizando o Elemento Vídeo em Conjunto com outro Elemento HTML

Novos atributos no elemento de vídeo podem ser usados, os quais também são comuns ao áudio, como, por exemplo, *loop* e *autoplay*. O atributo *pôster* indica qual imagem será mostrada quando o vídeo estiver começando a ser carregado. E o atributo *preload* permite fazer download do vídeo em segundo plano. O uso destes atributos é ilustrado na Figura 2.12.

```
<video poster="star.png" autoplay loop controls tabindex="0">
  <source src="movie.webm" type='video/webm; codecs="vp8, vorbis"' />
  <source src="movie.ogv" type='video/ogg; codecs="theora, vorbis"' />
</video>
```

Figura 2.12 Definição do elemento vídeo avançado

b) Utilizando o Elemento Vídeo e o Manipulado com JavaScript

O elemento de vídeo possui um conjunto de atributos, métodos e eventos que permite controlar o vídeo por meio de código JavaScript. Na Figura 2.13, ***listener*** é adicionado no JavaScript ao evento ***canplay*** para começar a manipular o vídeo.

```
video.addEventListener('canplay', function(e) {
  this.volume = 0.4;
  this.currentTime = 10;
  this.play();
}, false);
```

Figura 2.13 Uso de eventos do elemento vídeo em JS

c) Utilizando o Elemento Vídeo em Conjunto com o Elemento Canvas

Canvas é outro elemento HTML5 com muitas funcionalidades. Com o uso de canvas é possível explorar métodos para edições e apresentações avançadas em vídeo. No exemplo da Figura 2.14, dois recursos do elemento <canvas> para importar e exportar imagens são utilizados.

O método *drawImage* importa imagens a cada execução. Posteriormente, o quadro atual do vídeo é importado e processado no canvas. O segundo recurso utilizado é o método *toDataURL*, que permite exportar o conteúdo do canvas para uma imagem. No exemplo é gerada uma imagem do vídeo a cada 1,5 segundo.

```
video_dom.addEventListener('play', function() {
    video_dom.width = canvas_draw.width = video_dom.offsetWidth;
    video_dom.height = canvas_draw.height = video_dom.offsetHeight;
    var ctx_draw = canvas_draw.getContext('2d');
    draw_interval = setInterval(function() {

// import the image from the video
    ctx_draw.drawImage(video_dom, 0, 0, video_dom.width, video_dom.height);

 // export the image from the canvas
    var img = new Image();
    img.src = canvas_draw.toDataURL('image/png');
    img.width = 40;
    frames.appendChild(img);
  }, 1500)
}, false);
```

Figura 2.14 Explorando o elemento canvas para interagir com vídeo

2.2.2 Estilos CSS3

Paralelamente com o lançamento do HTML4 em 1997, foram criadas as folhas de estilo CSS (*Cascading Style Sheets*). CSS são utilizadas para especificar aspectos de apresentação separadamente da estrutura de páginas web. A separação de estrutura dos aspectos da apresentação simplifica a manutenção e a extensão das páginas web [Deitel & Deitel 2008].

Basicamente existem duas maneiras de descrever os estilos CSS: no arquivo HTML ou em um arquivo CSS. A descrição de estilos em arquivos separados permite que desenvolvedores utilizem esses estilos em diferentes páginas criando uniformidade na apresentação de conteúdos de um mesmo site web.

2.2.2.1 Sintaxe e Propriedades do CSS3

A declaração de estilos em CSS é formada por seletores e as propriedades, conforme Figura 2.15. O *seletor* representa uma estrutura, usada como uma condição para determinar quais elementos de um grupo de elementos serão formatados. A declaração das propriedades dos seletores deve respeitar a sintaxe CSS e devem ser separadas por ponto e vírgula.

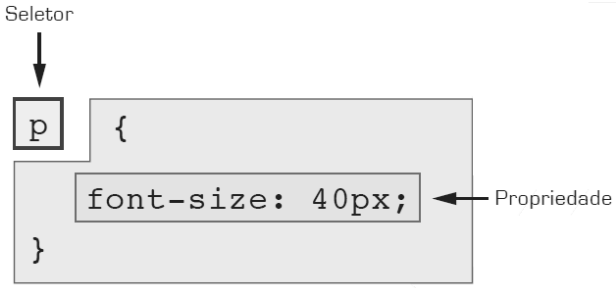

Figura 2.15 Sintaxe CSS

Existem três tipos de seletores, os encadeados, os agrupados e os complexos. Os encadeados são aqueles seletores que representam um elemento que está contido em outros elementos na sequência. Por exemplo, para formatar um link (elemento **a**) que se encontra dentro de um elemento **strong**, que sua vez está dentro do elemento **p** e este encontra-se em um elemento **div,** utiliza-se o seguinte seletor encadeado:

```
div p strong a {
      color: red;

}
```

Os seletores agrupados permitem declarar mais de um elemento com as mesmas propriedades. Os elementos, que são parte do agrupamento, são separados por vírgula. Por exemplo:

```
strong, em, span {
      color: red;

}
```

Já os seletores complexos permitem acessar elementos de difícil acesso pelos tipos anteriores, por exemplo, acessar descendentes, irmãos, filhos, que contenham atributos específicos ou pertençam a uma classe definida. Na Tabela 2.2 é apresentada uma tabela com os seletores complexos, seu significado e a versão do CSS onde foi especificado.

Tabela 2.2 Seletores complexos CSS

PADRÃO	SIGNIFICADO	CSS
elemento[atr]	Elemento com um atributo específico.	2
elemento[atr=”x”]	Elemento que tenha um atributo com um valor específico igual a “x”.	2
elemento[atr~=”x”]	Elemento com um atributo cujo valor é uma lista separada por espaços, sendo que um dos valores é “x”.	2
elemento[atr^=”x”]	Elemento com um atributo cujo valor começa exatamente com string “x”.	3
elemento[atr$=”x”]	Elemento com um atributo cujo valor termina exatamente com string “x”.	3
elemento[atr*=”x”]	Elemento com um atributo cujo valor contenha a string “x”.	3
elemento[atr|=”en”]	Um elemento que tem o atributo específico com o valor que é separado por hífen começando com EN (da esquerda para direita).	2
elemento:root	Elemento root do documento. Normalmente é o HTML.	3
elemento:nth-child(n)	Seleciona um objeto N de um determinado elemento.	3
elemento:nth-last-child(n)	Seleciona um objeto N começando pelo último objeto do elemento.	3
elemento:empty	Seleciona um elemento vazio, sem filhos, incluindo elementos de texto.	3
elemento:enabled elemento:disabled	Seleciona um elemento de interface que esteja habilitado ou desabilidade, como selects, checkbox, radio button etc.	3
elemento:checked	Seleciona elementos que estão “checados”, como radio buttons e checkboxes.	3
E > F	Seleciona os elementos E que são filhos diretos de F.	2
E + F	Seleciona um elemento F que precede imediatamente o elemento E.	2

2.2.2.2 O Novo CSS3

a) Gradiente

O uso de imagens para definir layout e background de páginas web gera requisições de download do servidor. Esse comportamento faz com que navegadores demorem mais para carregar as páginas quando o tráfego da rede fica congestionado. Com CSS3, é possível gerar gradientes unicamente com comandos próprios de CSS. Sendo que o CSS é processado pelo motor de renderização no cliente-side, diminui o número de requisições ao servidor.

Um gradiente é utilizado na propriedade background de qualquer elemento. Quando o CSS é processado pelo motor de renderização do navegador, é necessário fazer chamadas diferenciadas para cada motor, como ilustrado na Figura 2.16.

```
/* Para Mozilla/Gecko (Firefox etc) */

background: -moz-linear-gradient(top, #666, #fff) repeat-X;

/* Para WebKit (Safari, Google Chrome etc) */

background: -webkit-gradient(linear, left top, left bottom, from(#666), to(#fff)) repeat-X;

/* Para IE 8 */

-ms-filter:
"progid:DXImageTransform.Microsoft.gradient(startColorstr=#666,endColorstr=#FFFFFF)";

 /* Para IE 5.5 - 7 */

filter:progid:DXImageTransform.Microsoft.gradient(startColorstr=#666,endColorstr=#FFFFFFF);

/* Exemplo de gradiente complexo utilizando core e linhas intercaladas*/

background-color:#312d28;

background-image: linear-gradient(to right, rgba(255,255,255,1) 1px, transparent 1px), linear-
gradient(to right, rgba(255,255,255,.3) 1px, transparent 1px);

background-size: 50px 50px, 10px 30px;

background-repeat: repeat-x;
```

Figura 2.16 Declaração de gradiente suportados nos diferentes navegadores

Para definir o gradiente no exemplo da Figura 2.16, foi definido o tipo (linear), a direção (top, bottom), o início (#666) e o fim (#FFFFFF) das cores utilizadas na transição. Existem outros tipos de gradiente, como gradiente radial para fazer a transição desde um ponto inicial, aumentando o rádio do círculo em cada transição de cor.

Na Figura 2.17 são apresentados diferentes fundos utilizando os gradientes declarados na Figura 2.16.

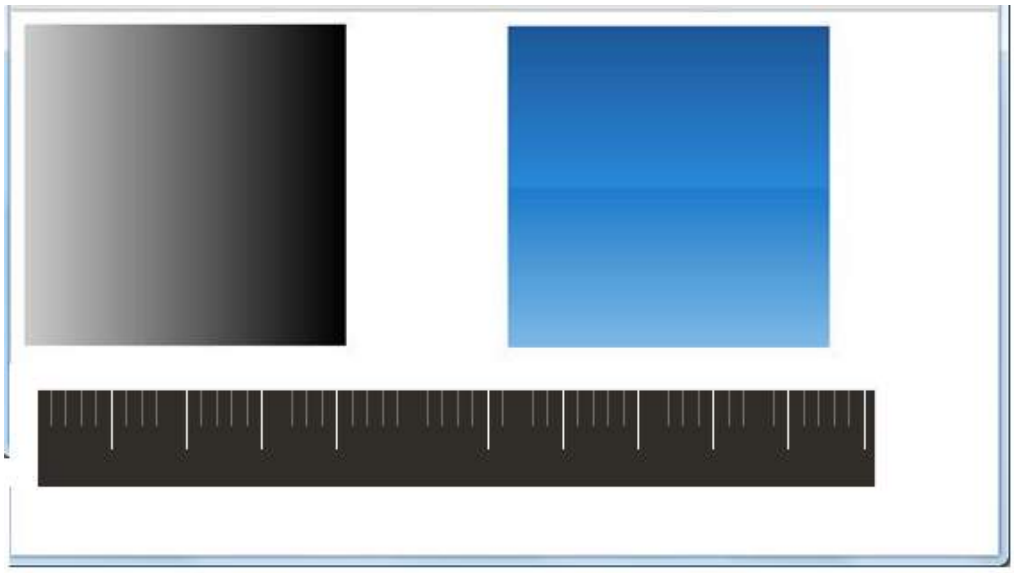

Figura 2.17 Exemplo de uso de gradientes

b) Bordas

A propriedade CSS *border-radius* é responsável por aplicar bordas arredondadas em um elemento HTML. A sintaxe da propriedade *border-radius* permite a especificação do grau de curvatura dos cantos das bordas. A curvatura pode ser igual nos quatro cantos da borda, especificando um único valor, ou cada borda pode ter seu grau de curvatura, como ilustrado na Figura 2.18.

```
# main {
  border - radius : 10 px; // Quatro cantos iguais
 }
# menu {
      border - radius : 10 px 20 px; // Os dois cantos superiores com 10px e os inf com 20px.
 }
# side {
  border - radius : 40 px 30 px 20 px 10 px;//Cada canto com curvaturas distintas.
}
```

Figura 2.18 Exemplo do uso de border-radius

c) Sombras

No CSS3, existem dois tipos de sombras: a sombra que é aplicada a "caixa" do elemento e a sombra que é aplicada no texto de um elemento.

O parâmetro *box-shadow* é responsável por aplicar uma sombra na parte externa ou interna da "caixa" de um elemento. A "caixa" é limitada pelo retângulo definido pela borda do elemento de acordo com o *box model*. Referências de uso do box model podem ser encontradas na especificação CSS. O parâmetro *box-shadow* recebe os valores que contêm informação de deslocamento, desfocamento, propagação e cor. A Figura 2.19 exemplifica como todos os elementos de classe "cam-json-bt" terão uma imagem de tamanho mínimo de 75 px e com sombreamento externo deslocado na horizontal em 1px e sem deslocamento na vertical, desfocado 1px na cor preta.

```
.cam-json-bt{
        background-image: url("images/camerasButtonNormal@2x.png");
        min-height:75px;
        min-width:100%;
        background-repeat: no-repeat;
        background-size:96% 98%;
        background-position:top center;b
        box-shadow:1px 0px 1px #FFFFFF;
}
```

Figura 2.19 Declaração de propriedades para a classe "cam-json-bt"

Para gerar sombras na parte interna da "caixa", deve-se adicionar o valor *inset* na propriedade *box-shadow*. Exemplo: box-shadow: *inset* 1px 0px 1px #FFFFFF.

d) Transformações

A propriedade *transform* manipula o translado, escalonamento, distorção, perspectiva e rotação do elemento do HTML ao ser apresentado. No CSS3, os seguintes valores para a propriedade *transform* foram introduzidos:

- ***scale*** modifica a dimensão do elemento.
- ***skew*** modifica os ângulos dos elementos.
- ***translation*** move o elemento no eixo X e Y.
- ***rotate*** rotaciona o elemento levando em consideração seu ângulo, especialmente quando o ângulo é personalizado com o transform-origin.

Na Figura 2.20 são aplicadas transformações de escalonamento e de rotação para elementos *img.* Essa transformação está sendo realizada para todos os motores de renderização.

```
img {
-webkit-transform: scale(1.5) skew(30deg); /* para webkit */
-moz-transform: scale(1.5) skew(30deg); /* para gecko */
-o-transform: scale(1.5) skew(30deg); /* para opera */
transform: scale(1.5) skew(30deg); /* para browsers sem prefixo */
}
```

Figura 2.20 Exemplo de uso de transformações

2.2.3 Interatividade para páginas Web

2.2.3.1 Transições e animações

As propriedades *transition* e *animation,* e a regra *keyframe* entre outras apresentadas nas seções anteriores, mudaram completamente o uso do CSS. Atualmente é possível fazer interfaces de interação animadas utilizando unicamente regras CSS.

A propriedade *animation* tem uma série de propriedades que podem ser resumidas em um *shortcode* simples. A Tabela 2.3 apresenta as propriedades e os significados.

Tabela 2.3 Propriedades para animação

Propriedade	Definição
animation-name	Especifica o nome da função de animação
animation-duration	Define a duração da animação. O valor é declarado em segundos.
animation-timing-function	Descreve qual a progressão da animação a cada ciclo de duração. Os valores possíveis são: *ease, linear, ease-in, ease-out, ease-in-out, cubic-bezier(<number>, <number>, <number>, <number>) [, ease, linear, ease-in, ease-out, ease-in-out, cubic-bezier(<number>, <number>, <number>, <number>)]** O valor padrão é *ease*.
animation-interation-count	Define o número de vezes que o ciclo deve acontecer. O padrão é um. Para ser declarado como infinito com o valor *infinite*.
animation-direction	Define se a animação irá acontecer ou não no sentido inverso em ciclos alternados. Ou seja, se a animação está acontecendo no sentido horário, ao acabar a animação, o browser faz a mesma animação no elemento, mas no sentido anti-horário. Os valores são alternate ou normal.
animation-play-state	Define se a animação está acontecendo ou se está pausada. Desenvolvedores web poderão, por exemplo, via script, pausar a animação se ela estiver acontecendo. Os valores são running ou paused.
animation-delay	Define quando a animação irá começar, ou seja, o desenvolvedor web define um tempo para que a animação inicie. O valor 0 significa que a animação começará imediatamente.

2.2.3.2 JavaScript, JQuery e JQueryMobile

Uma grande maioria de regras e de funções de interatividade com páginas Web já é possível deixar sob responsabilidade do CSS. No entanto, páginas web não podem existir sem ter scripts que gerenciem e manipulem informação vinda de elementos do HTML. Nesta seção são apresentadas, de forma sucinta, a linguagem JavaScript e as bibliotecas JQuery e JQueryMobile e as informações básicas para habilitar o uso no desenvolvimento de páginas interativas.

a) JavaScript

JavaScript é uma linguagem de script orientada a objetos do tipo client-side. Era originalmente codificada como parte dos navegadores web para que scripts pudessem ser executados do lado do cliente e interagissem com o usuário.

b) JQuery

JQuery é uma biblioteca JavaScript desenvolvida para simplificar os scripts client-side que interagem com o HTML e solucionar problemas de incompatibilidade entre navegadores [JQuery 2013]. JQuery também permite reutilização de código por meio do uso de plugins de outros desenvolvedores.JQuery manipula AJAX e DOM de forma simples e se integra facilmente aos recursos de CSS3. Na Figura 2.21 é comparada a manipulação de elementos.

```
<head>
  <script type =" text / javascript " src =" https :// ajax . googleapis . com / ajax / libs / jquery-
     /1.7.2/ jquery . min .js">
  </ script >
</head>
// Javascript para atribuir a uma id o valor "5".
document.getElementById( 'Teste' ).value = 5;
// O mesmo código em jQuery.
$( '#Teste' ).val( 5 );
```

Figura 2.21 Manipulação de Elementos, JavaScript vs. JQuery

Para utilizar a biblioteca JavaScript JQuery, é necessário adicionar a referência para o arquivo js na tag <script>, conforme Figura 2.21.

A sintaxe básica para executar uma ação sobre determinados elementos é: *$(seletor).acao()*. O símbolo *$* é um método para criar o objeto JQuery, o *(seletor)* serve para consultar e encontrar os elementos HTML e a *acao()* define a operação JQuery que será executada nos elementos. O exemplo da Figura 2.21 apresenta a atribuição do valor 5 no elemento "*Teste"* do HTML.

O *(seletor)* pode ser uma expressão para encontrar um ou mais elementos do HTML. Por exemplo, para selecionar todos os elementos de um documento se utiliza a expressão *$(*)* ou, para selecionar todos os elementos, utiliza-se *div, $(div)*. Além disso, podem ser utilizadas expressões complexas como por exemplo, *$('input [name ^=" curso "] ')$* para selecionar todos os elementos <*input* > cujo atributo *name* comece com a palavra "curso". Informações detalhadas e tutorias da linguagem podem ser encontradas em http://jquery.com/.

c) JQueryMobile

JQueryMobile é um framework para desenvolvimento web, otimizado para interação por toque e que se destina à criação de aplicações para smartphones e tablets. Seu desenvolvimento está voltado para fornecer mecanismos capazes de criar sistemas unificados de interface de usuário, baseados em HTML5 e CSS3. Os scripts em

JQueryMobile são capazes de rodar em todas as plataformas móveis tendo como base de construção as bibliotecas jQuery e jQuery UI.

Para utilizar a biblioteca JQueryMobile, deve-se obrigatoriamente declarar um link para a biblioteca JQuery, outro para o framework JQueryMobile e um outro para a folha de estilo CSS3, padrão do framework, como apresentado na Figura 2.22 .

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <title>Exemplo JQueryMobile</title>
    <link rel="stylesheet" href="http://code.jquery.com/mobile/latest/jquery.mobile.min.css “ />
    <script src="http://code.jquery.com/jquery.min.js"></script>
    <script src="http://code.jquery.com/mobile/latest/jquery.mobile.min.js “></script>
  </head>
  <body>
  </body>
</html>
```

Figura 2.22 Declarações mínimas para uso de JQueryMobile

A *metatag viewport,* declarada no exemplo da Figura 2.22, é fundamental para o funcionamento das páginas em dispositivos móveis, informando ao navegador como deve ser renderizado o conteúdo do documento HTML. É possível determinar a largura e a altura da *viewport* e atributos de escalabilidade das interfaces.

Informações detalhadas e tutorias da linguagem podem ser encontradas em http://jquerymobile.com/.

2.3 Conceitos e Componentes Físicos para Automação

Os sistemas controle domésticos (*Home Control System* – HCS) ou automação residencial é composto de uma rede de comunicação que permite a interconexão de uma série de dispositivos, equipamentos e outros sistemas. O objetivo principal da automação é melhorar a qualidade de vida, aumentar a segurança e viabilizar o uso racional dos recursos para seus usuários. Para atingir este objetivo, os equipamentos ligados na automação enviam informações sobre o ambiente residencial e o meio em que ele se insere, efetuando determinadas ações a fim de supervisioná-lo ou gerenciá-lo (BOLZANI, 2004).

2.3.1 Automação Residencial

A automação residencial é iniciou-se suportada por a automação industrial, porém, em virtude da diferente realidade entre os dois cenários, atualmente têm sido criados sistemas dedicados para ambientes onde não se dispõe de espaço para grandes centrais controladoras e extensos sistemas de cabeamento. Na automação residencial pode-se

encontrar maior diversidade de dispositivos. Os dispositivos residenciais abrangem desde controle de luzes, equipamentos multimídia até o tráfego de dados de telemetria. No ambiente residencial, diferente do ambiente industrial, essas tecnologias são utilizadas por pessoas que não necessariamente possuem qualquer conhecimento técnico. Isso torna primordial o desenvolvimento de interfaces de usuário simples e intuitivas.

Automação residencial traz a possibilidade de facilitar tarefas diárias como irrigar jardins, estabelecer níveis de temperatura e de iluminação, evitar preocupações com janelas em dias de chuva ou portas abertas ao sair de casa, dentre outros. O auxilio ou a total execução destas tarefas contribui para oferecer níveis de conforto e ao mesmo tempo conduzir a economias de eletricidade, gás e água. A automação residencial também é eficiente quanto à segurança, detectando ou sinalizando situações de emergência ou situações de intrusão, podendo acionar imediatamente as centrais de segurança (Dias, C. L. A, Pizzolato N. D. ,2004).

Os sistemas de automação podem ser desenvolvidos em dois tipos de arquiteturas: centralizadas ou distribuídas. Na arquitetura centralizada, todos os dispositivos são conectados a uma unidade central de controle a qual é responsável pelo controle e gerenciamento das reações aos eventos recebidos e das ações ou comandos a serem executados pelos dispositivos.

Na arquitetura distribuída ou descentralizada, os diversos dispositivos são capazes de processar ações com critérios próprios. Cada ação uma função específica dentro das inúmeras necessidades do sistema de automação, sendo interligados por uma rede, comunicando-se e enviando sinais entre sensores e atuadores. Esses sensores e atuadores podem se encontrar próximos ou integrados ao ponto de controle e monitoração (Dias, C. L. A, Pizzolato N. D. ,2004).

Um sistema de automação residencial deve ser desenvolvido de acordo com os requisitos propostos pelo usuário. Somente o usuário pode decidir suas prioridades e o quanto vai investir. Assim a automação residencial é específica para cada casa a ser automatizada. Na Figura 2..23 são apresentados os dispositivos mais comuns que podem ser monitorados e controlados pela automação.

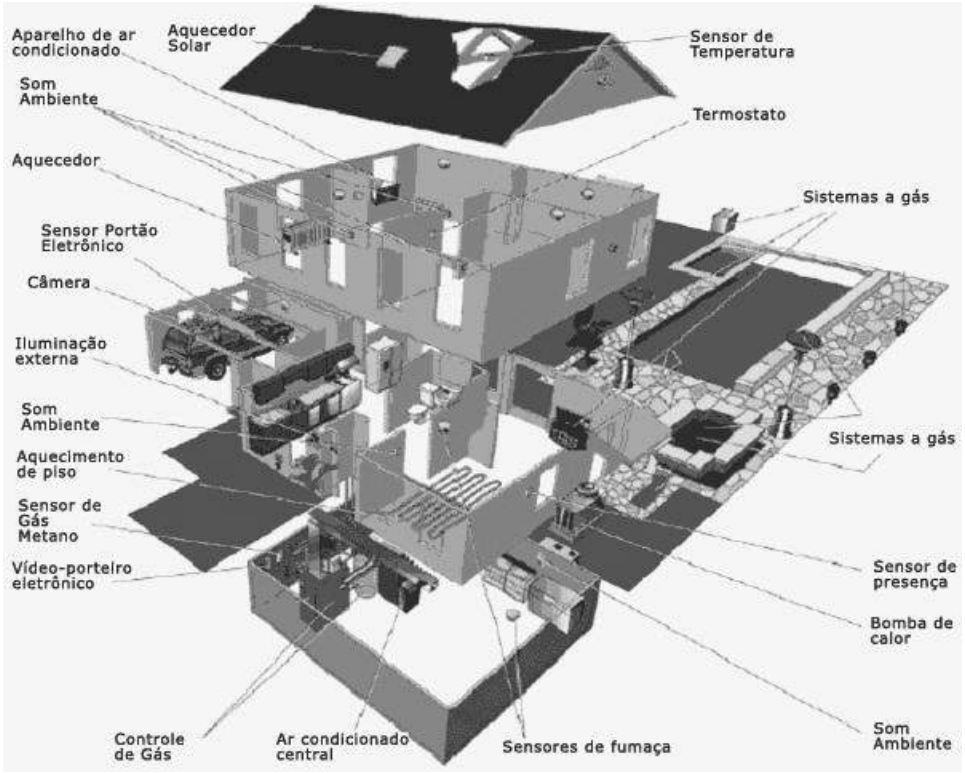

Figura 2..23 - Automação Residencial (Fonte: www.idealhome.com acesso em 20/08/2013).

A representação da automação residencial da Figura 2.23 ilustra uma automação organizada como todo sistema de automação, que tem uma central considerada o cérebro da casa. Os componentes mais comuns utilizados para criar a central de automação residencial são PIC, Arduino, CLP (comando lógico programável). Esses componentes tem inteligência para gerenciar regras de execução. Eles que detêm toda a lógica da automação.

Os *dispositivos de controle e monitoramento* têm um endereçamento ou id e tem funções específicas. Existem também *atuadores liga/desliga*, *atuadores dimmer, atuadores para motores*, *atuadores infra-vermelho*, *interfaces para sensores para botões* e *controladoras de acesso* (RFID, Teclados, Biometria, Controle remoto, etc). Os *sensores* são os que leem informação da casa. Eles podem ser de movimento (para liga/desliga de luzes, câmeras e som ambiente), luminosidade (para abertura de persianas), abertura (para portões e janelas), gás (para controle de gás), temperatura (para, por exemplo, aparelho de ar condicionado, aquecimento de piso, aquecedor), umidade (para ligar/desligar irrigação), nível de água (para bomba de água e irrigação), consumo de energia (para gerar relatórios de consumo por mês, liga/desliga automático de luzes), etc.

Existe uma rede de comunicação entre os sensores, que normalmente utiliza o protocolo RS232. Atualmente existe uma associação, a KNX, que está criando padrões de comunicação no modelo OSI para dar mais segurança na troca de informação. Basicamente, a central envia comandos aos atuadores se uma regra for atingida. Existem regras de horários, cenários, ou para interpretação das informações vindas dos sensores ou controladores de acesso.

Os dispositivos de controle de acesso e segurança permitem identificar os usuários do sistema possibilitando ao sistema de automação gerenciar o comportamento de ações personalizadas e a restringir áreas da residência. Por exemplo, prestadores de serviço podem ter acesso apenas a uma porta específica em um determinado período, um visitante pode ter acesso apenas a um ambiente e os moradores têm acesso a todos os ambientes.

Os sistemas de controle domésticos estão se tornando mais comuns e parte integrante de habitações modernas. Algumas tecnologias, como os sistemas para controle de iluminação, estão presentes em casas, apartamentos e escritórios de médio e alto padrão, além de grandes empresas, teatros, hotéis e hospitais.

2.3.2 Automação para Health Care

As áreas industrial, comercial e residencial se mostram mais familiarizadas com a prática de automação. No entanto, a área da saúde ainda é pouco automatizada[2]. As grandes motivações da automação na área da saúde são o ganho de conforto e a segurança que ela pode proporcionar. A maioria dos procedimentos de Health Care ainda é realizada manualmente, tornando-os lentos e mais propícios a erros.

Seguindo a linha de monitoramento de pacientes, Murakami et al. (MURAKAMI, 2006) desenvolveram o vMon- Gluco que implementa o monitoramento em tempo real dos níveis de glicose dos pacientes. Esse sistema foi desenvolvido sobre dispositivos móveis, sendo utilizado em Unidade de Terapia Intensiva (UTI). Esse trabalho permite que pacientes com altas taxas de glicose possam ser monitorados de forma automatizada e com uma frequência maior e mais precisa, melhorando a qualidade do atendimento do paciente. Outra vantagem do sistema é possibilitar o escalonamento mais eficiente da equipe médica, uma vez que um processo que demandava tempo e recursos humanos foi automatizado.

Um aspecto importante da automação na saúde é a usabilidade do sistema. O sistema deve ser o mais fácil e intuitivo possível. Os profissionais da saúde usam vários dispositivos que se encontram em diversas áreas, ou seja, o ambiente de atuação é volátil. Um sistema de baixa usabilidade exigiria uma nova autenticação a cada ambiente, o que dificulta o uso do sistema, criando muitas vezes rejeição da tecnologia (BARDRAM, 2005). Nesse sentido, a tecnologia RFID presente na automação residencial se mostra eficiente para a autenticação dos usuários na saúde. A identificação por radiofrequência (RFID) é uma tecnologia para identificação automatizada de objetos e pessoas (MOORE, 2011). Dessa forma, não é necessário que o profissional se identifique a cada novo ambiente, o RFID o identifica.

A automação na saúde pode utilizar várias tecnologias provenientes da automação residencial. No entanto, antes de submeter a automação em hospitais e clínicas, as prioridades e áreas mais críticas devem ser devidamente analisadas (Saúde Automatizada, 2005). Na área da saúde, a preocupação com a segurança de dados e a intolerância a falhas são requisitos essenciais.

[2] http://www.bpsolutions.com.br/bpmaga/edicao11/16_17_DICAS.pdf Dicas de automação em saúde.

2.3.3 Estudo de Caso - Automação Residencial

Como estudo de caso é utilizado o sistema de automação residencial SIAM[3] (Sistema Integrado de Automação e Monitoramento). SIAM é também uma indústria brasileira que desenvolve equipamentos e software para automação, telemetria, controle de acesso e vigilância remota.

Os produtos da SIAM são voltados para aumentar a comodidade, o controle e a segurança de seus clientes em relação às suas residências ou locais de trabalho. Por meio de uma rede de dados, é possível automatizar funções e oferecer segurança em diversas aplicações, principalmente, automação e telemetria.

2.3.3.1 Arquitetura do Sistema

Os produtos SIAM são suportados por uma arquitetura distribuída, onde cada sensor e cada atuador têm identificador único e ações associadas para monitorar e controlar dispositivos. Os sensores e atuadores são interligados por uma rede de dados que permite a troca de informação com a central de gerenciamento. A Figura 2.24 ilustra a arquitetura do sistema SIAM que suporta a comunicação de softwares e hardwares.

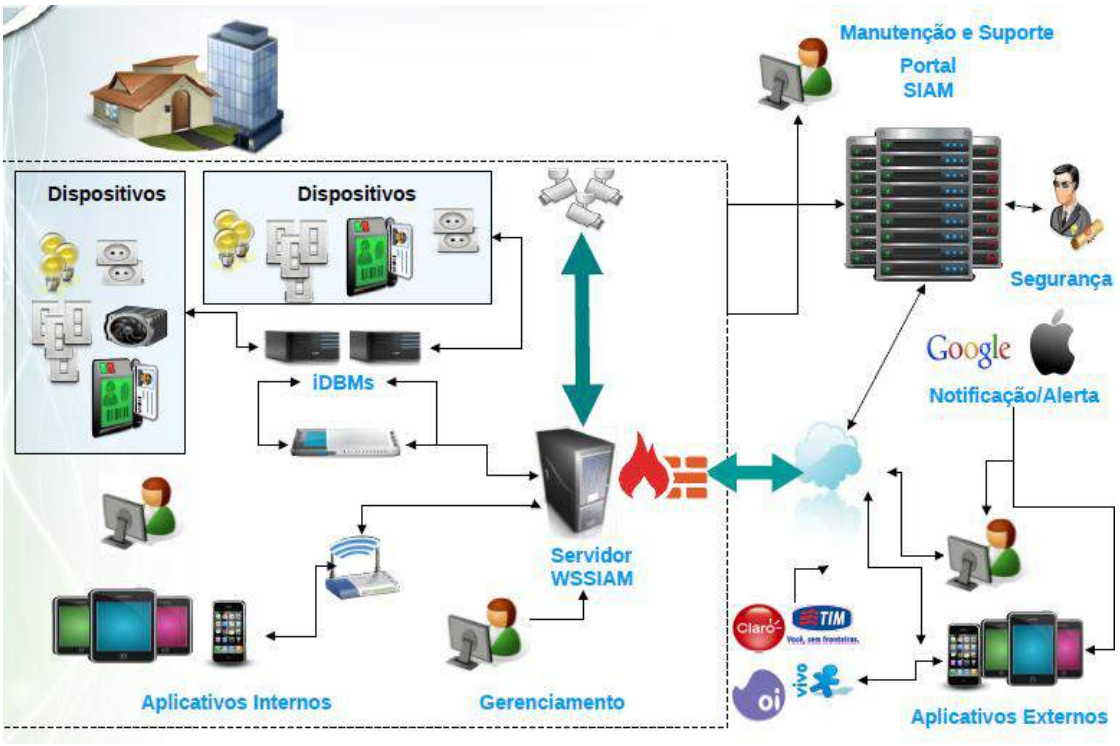

Figura 2.24 - Arquitetura do sistema de automação residencial SIAM

O Sistema Integrado de Automação e Monitoramento (SIAM) é formado pelos seguintes componentes de hardware e de software:

- **iDBM:** é a central de controle que gerencia a informação gerada pelos sensores e dispositivos. De acordo com essa informação, a iDBM executa as regras

[3] http://siam1.hospedagemdesites.ws/Site/Enterprise/Enterprise.aspx

definidas pelo instalador e comunica as ações a serem executadas pelos dispositivos.

- **WSSIAM:** é um servidor desenvolvido em Java com interfaces para web em Adobe Flash. O WSSIAM é responsável por monitorar e gerenciar a comunicação e a configuração de hardwares e de softwares. Este servidor pode ser instalado na residência e/ou em local remoto mantendo sincronização dos dados em todas suas instâncias.

- **SIAM-Mobile:** é interface do sistema com usuário final. O sistema desenvolvido com tecnologia HTML5, CSS3, Ajax e JQueryMobile, permite o controle de ações e de leitura de eventos ocorridos no sistema de automação de forma remota.

- **VRC:** é um servidor de Vigilância Remota que permite o acesso a câmeras de segurança e de controle dos ambientes onde as câmeras estão instaladas.

- **Sensores:** são utilizados para diversos tipos de monitoramento como temperatura, umidade, níveis de água, luminosidade, presença, consumo de energia, detecção de incêndio, entre outros.

- **Atuadores:** são equipamentos utilizados para realizar ações como ligar alarme, ativar irrigação, ligar/desligar luzes, abrir portas, entre outros.

- **Controle Segurança:** são dispositivos para permitir a identificação de usuários por meio de sensores de RFID, teclados para senhas, biometria e controle remoto.

2.3.3.2 Sistema SIAM-Mobile

Nesta seção vamos apresentar as interfaces desenvolvidas com as técnicas apresentadas neste curso, HTML5, CSS3 e jQueryMobile são discutidas.

Antes do desenvolvimento das interfaces web, foi necessário criar uma interface de comunicação entre os serviços que compõem o sistema de automação. As diretrizes definidas nas camadas de comunicação, apresentada na Figura 2.25, foram seguidas. Foi criado um servidor para autenticar as requisições web o qual instancia as *capabilities* que um determinado usuário tem no sistema. Uma vez definidas as *capabilities*, uma linha de comunicação é vinculada entre os serviços e o hardware de automação.

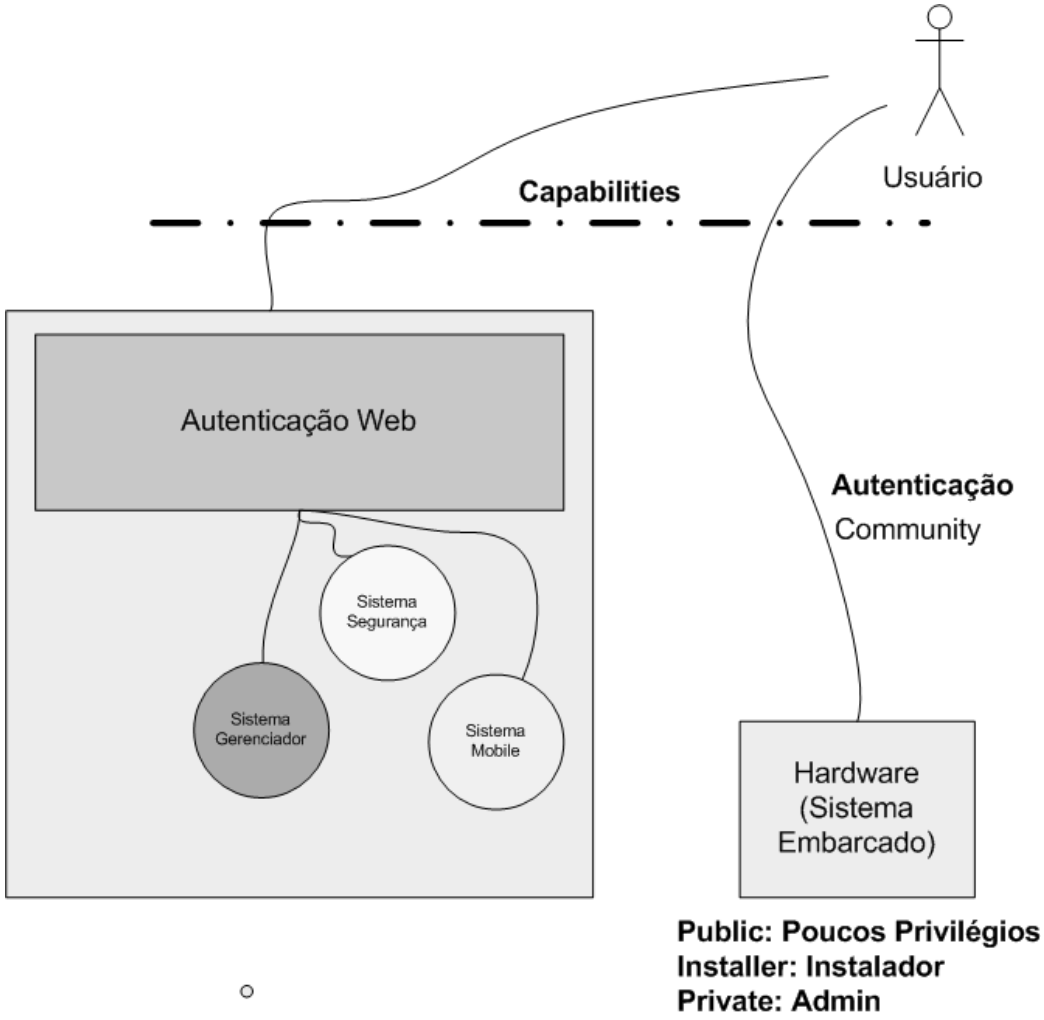

Figura 2.25 - Camadas de Comunicação do SIAM

Na Figura 2.26 é apresentada a preparação da página HTML5 do sistema web de automação. Na página principal do sistema web móbile foi definido inicialmente o atributo *lang="pt-br"* no elemento principal <HTML> para indicar o navegador que a linguagem principal do documento é português do Brasil. Em seguida definimos a metatag *charsert="utf-8"* foi criada para informar a codificação do conteúdo. E o *viewport* é de dimensão fixa do tamanho da tela do dispositivo.

O elemento <*link*> foi utilizado para relacionar imagens que serão utilizadas quando um shortcut à página seja criado. Em seguida, foram relacionados os arquivos para uso das bibliotecas de JQuery, JQueryMobile e Storage. A biblioteca Storage permite o armazenamento e a recuperação de dados no cliente-side. Posteriormente, foram relacionados os arquivos para as folhas de estilo CSS. É importante observar que foi utilizado o atributo *media* no elemento *link* que faz referência ao CSS para determinar qual arquivo será utilizado. Se o dispositivo tiver uma largura da tela maior a 615px será utilizado o arquivo landscape.css, caso contrário o arquivo portrait.css.

Finalmente foram relacionados os arquivos que contem os scripts desenvolvidos para tratamento de informação, eventos e interação com usuário.

```
<!DOCTYPE html>
<html lang=”pt-br”>
  <head>
        <meta charset="utf-8" />
        <title>SIAM Mobile</title>
        <meta name="viewport" content="width=device-width,initial-scale=1,maximum-scale=1"/>
        <!--
        Icones para Shortcut
        -->
        <link rel="apple-touch-icon" sizes="72x72" href="src/css/images/icon@2x.png" />
        <link rel="app-icon" sizes="24x24" href="src/css/images/icon.png" />
        <link rel="shortcut icon" href="src/css/images/icon-ipad.png"/>
        <!--
        Bibliotecas java script utilizadas
        -->
        <script src="lib/jquery-1.7.2.min.js"></script>
        <script src="lib/jquery.mobile-1.2.0.js"></script>
        <script src="lib/jquery.Storage.js"></script>
        <link href="src/css/jquery.mobile.structure-1.1.0.css" rel="stylesheet">
        <link href="src/css/jquery.mobile.theme-1.1.0.css" rel="stylesheet">
        <!--
        Referencias para arquivos customizados
        siam_x.css define a fonte, localização, tamanho e todos os atributos
        -->
        <link rel="stylesheet" href="src/css/siam_landscape.css" type="text/css" media="only all and (min-width:615px)" title="no title" />
        <link rel="stylesheet" href="src/css/siam_portrait.css" type="text/css" media="only all and (max-width: 615px)" title="no title" />
        <!--
        Link para arquivo de metodos em JavaScript para comunicação com a o WSSIAM
        -->
        <script src="src/js/siam_ws.js"></script>
        <script src="src/js/siam.js"></script>
  </head>
  <body>…….....</body>
</html>
```

Figura 2.26 Definições da página inicial do sistema de automação

```
<div id="login" data-role="page" class="type-home" data-theme="a" >
   <header id="header-div" class="ui-siam-header" data-role="header" data-position="fixed" data-tap-toggle="false" data-theme="a">
        <span id="header-title" class="head-label" >Login SIAM </span>
    </header>
        <div id="main-page" data-role="content" data-tap-toggle="false">
            <form id="main-content" class="content-secondary" action="#inicio">
                <label for="usuario">Usu&aacute;rio</label>
                <input type="text" name="usuario" data-role="none">
                <label for="senha">Usu&aacute;rio</label>
                <input type="password" name="senha" data-role="none">
                <input type="submit" value="submit" onClick="login();">
             </form>
        </div>
       <footer data-position="fixed" data-tap-toggle="false">
           <nav data-role="navbar" class="siam-navbar">
            <ul>
              <li><a href="#inicio"    class="home-json-bt"  data-iconpos="notext"></a></li>
              <li><a href="#ambientes" class="envs-json-bt"  data-iconpos="notext"></a></li>
            </ul>
           </nav>
        </footer>
</div>
```

Figura 2.27 Conteúdo da página inicial

Na Figura 2.27 é apresentada a definição dos elementos que compõem a página de login e de início. Nessa página foram utilizadas as tags <header> para definir o cabeçalho e <*footer*> para instanciar um rodapé. Dentro do elemento <*footer*>, foi criado um elemento <*nav*> para referenciar o menu de navegação do sistema. No elemento <*footer*> foi utilizado o atributo *data-position="fixed"* que fixa o elemento para não ser afetado pelo scroll do navegador.

Nas Figura 2.28 e na Figura 2.29 são exibidas as interfaces de usuário correspondentes ao código da Figura 2.27 no navegador desktop e no smartphone, respectivamente.

Figura 2.28 Páginas de Login e Inicio em navegador Desktop

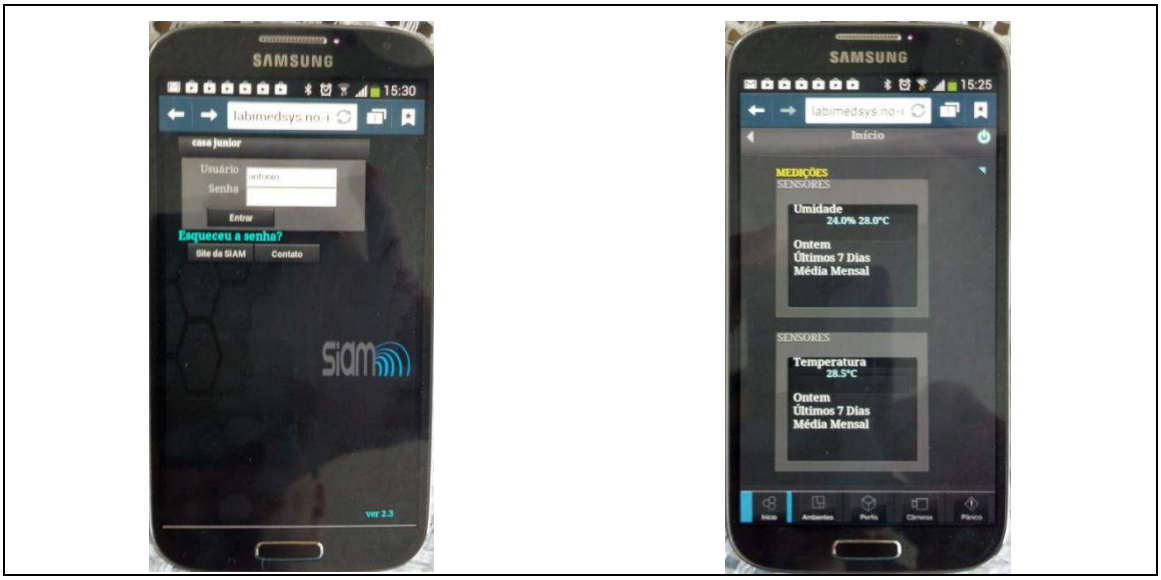

Figura 2.29 - Páginas de Login e Inicio em SmartPhone

Na Figura 2.30 são exibidos elementos de *vídeo* e elementos para controle de iluminação e acesso a portas. Quando pressionado um botão, o evento *touchstart* é acionado pelo navegador. Esse evento está sendo monitorado por um método codificado no arquivo js. Quando o evento é acionado, o script dispara uma chamada ao serviço de comandos o qual validará a permissão e, posteriormente, enviará uma requisição para o hardware de controle.

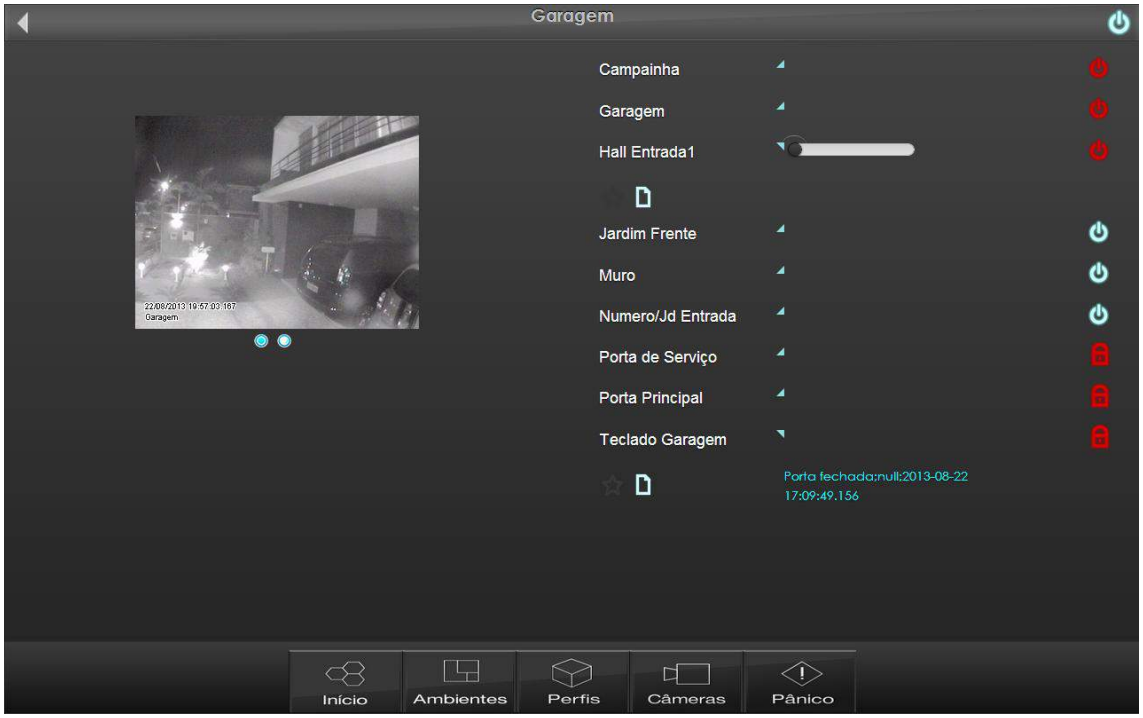

Figura 2.30 Página exibindo Vídeo e Dispositivos de controle

2.4 Biografias Resumidas dos Autores

2.4.1 José Antonio Camacho Guerrero

José Antonio Camacho Guerrero é formado em Engenharia de Sistemas Computacionais pelo Instituto Tecnológico y de Estudios Superiores de Monterrey - México (1997) e possui título de mestre em Ciência da Computação e Matemática Computacional pela Universidade de São Paulo (2002). Atualmente é fundador e coordenador de projetos das empresas Innolution - Sistemas de Informática Ltda e da i-Medsys. Tem experiência na área de Ciência da Computação, com ênfase em Gerenciamento de Informação, atuando principalmente nos seguintes temas: recuperação de informação, hipermídia, web semântica, automação residencial, software de apoio e acompanhamento médico e equipamentos móveis.

2.4.2 Alessandra Alaniz Macedo

Alessandra Alaniz Macedo possui graduação em Ciência da Computação pela Universidade Estadual de Londrina (1996), mestrado em Ciência da Computação pela Universidade de São Paulo (1999), doutorado em Ciência da Computação pela Universidade de São Paulo (2004). Atualmente é professora na Universidade de São Paulo, Ribeirão Preto. Esta pesquisadora coordenou nos últimos anos dois projetos FAPESP: Projeto Jovem Pesquisador e TIDIA-Ae. Esta pesquisadora tem desenvolvido trabalhos e publicado artigos nas áreas extração e relacionamento de informação.

2.5 Agradecimentos

Os autores agradecem à FAPESP pelo apoio na apresentação do trabalho e a empresa SIAM pelo ambiente proporcionado.

2.6 Referencias

Barros, V. F. A. and Pinto JR, J and Borges, R. C. (2011) "*Aplicativo Móvel para Automação e Monitoração do Sistema de Atenção Primária a Saúde"***.** Cadernos de Informática V.6. Universidade Federal de Goiás. Goiânia, Goiás, p 241-244.

Bardram, J. E. (2005) "The trouble with login on usability and computer security". Proceedings on Ubiquit Computing, p 355-367.

Bolzani, C. A. M. (2004) "Residências Inteligentes: um curso de Domótica" (2004). 1. ed. São Paulo: Editora Livraria da Física, p. 332.

Deitel, P. J. and Deitel H. M. (2008) "AJAX, Rich Internet Applications, and Web Development for Programmers". Person Education Inc.

Dias, C. L. A. and Pizzolato, N. D. (2004) "Domótica – Aplicabilidade e Sistemas de Automação Residencial". Vértices, volume 6, setembro/dezembro.

Murakimi, A. and Gutierrez, M. A.and Lage, S. H. G. and Rebelo, M. F. S.and Ramires, J. A. F. (2006) “A Continuous Glucose Monitoring System in Critical”. IEEE Computers in Cardiology, v. 32, 2006, p. 10-14.

Moore, B. (2011) “A Health(care) conscious Technology”. Disponível em: www.rfid.org . Acesso dia 20/08/2013.

W3C. (2013) “Consórcio World Wide Web”. Url: http://w3c.com/. Acesso dia 20/08/2013.

WHATWG. (2013) “Web Hypertext Application Technology Working Group” (2013). url: http://www.whatwg.org/. Acesso dia 20/08/2013.