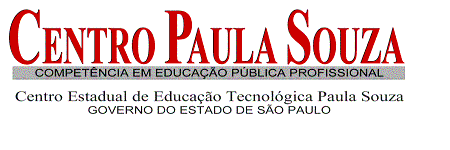
****

**ETEC FRANCISCO GARCIA**

**CLASSE DESCENTRALIZADA CAJURU**

**TÉCNICO EM INFORMÁTICA**

**ADENALDO OLIVEIRA DA SILVA**

**GUSTAVO FERREIRA DA SILVA**

**LUIZ GUSTAVO MOREIRA DA SILVA**

**AUTOMAÇÃO RESIDENCIAL COM ARDUINO**

**Cajuru**

**2013**

ADENALDO OLIVEIRA DA SILVA

GUSTAVO FERREIRA DA SILVA

LUIZ GUSTAVO MOREIRA DA SILVA

AUTOMAÇÃO RESIDENCIAL COM ARDUINO

TCC apresentado a ETEC Francisco Garcia sob a orientação do Prof. Pedro Ramires da Silva Amalfi Costa

CAJURU

2013

Dedicamos este trabalho aos nossos professores que sempre nos apoiaram e colaboraram com a realização deste. Também aos nossos familiares pelo apoio e incentivo.

Agradecemos aos professores e a todos os que colaboraram na elaboração deste projeto.

“Uma experiência nunca é um fracasso, pois sempre vem demonstrar algo” (Thomas Edison)

**RESUMO**

Este projeto consiste em uma demonstração de uma forma de automação residencial., onde através de um sistema se controla toda casa desde o portão da garagem até na luz do banhiero.

O sistema rodará sobre uma placa eletronica arduíno que gerencia toda a automação da casa e através de um software o sistema é controlado.

O software foi desenvolvido em uma linguagem de progrmação da Microsoft o C# a interface e bem definida e simplificada a onde qualquer pessoas consegui usá-lo.

O custo para automatizar é caro? Fica um pouco caro mais na parte de segurança a casa está bem protegida pois você sabe quem esta acessando o sistema.

Qualquer pessoa pode automatizar sua casa? Não a primeiramente precisa possuir qualificação suficiente para realizar o sistema. Antes de automatizar qualquer casa consulte um engenheiro para dar os parametros necessários como qual o disjuntor comprar.

Onde comprar o arduíno ? No Brasil o representante oficial é a robocore que uma loja virtual que distribui a placa pelo país 3

**Palavras-chave:** Arduino, Automação Residencial, Informática, Eletrônica, C#, Visual Studio.

**LISTA DE FIGURAS**

[Figura 1 – Diagrama da Arquitetura do Android 11](#_Toc341896602)

[Figura 2 – Ambiente de Desenvolvimento Eclipse com Plugin ADT (Android Development). 16](#_Toc341896603)

[Figura 3 – Android Virtual Device (AVD). 17](#_Toc341896604)

**SUMÁRIO**

1. INTRODUÇÃO 9

2. O ARDUINO 10

2.1 HISTÓRIA 10

2.2 O QUE COMPÕE O ARDUINO 11

2.2.1 MICROCONTROLADOR 11

3.1 RECURSOS 11

3.2 CAMADAS 13

3.2.1 CAMADA DE APLICAÇÕES 14

3.2.2 CAMADA DE FRAMEWORK DE APLICAÇÕES 14

3.2.3 CAMADA DE BIBLIOTECAS 15

3.2.4 CAMADA RUNTIME DO SISTEMA 16

3.2.5 CAMADA KERNEL LINUX 16

3.3 O DESENVOLVIMENTO ANDROID 17

3.3.1 ANDROID VIRTUAL DEVICE (AVD) 19

3. O SERVIDOR 20

4.1 APACHE 20

4.2 PHP 21

4.3 MYSQL 21

4.4 XAMPP 22

4.5 A INTEGRAÇÃO COM O ANDROID 22

4. CONCLUSÃO 23

5. REFERÊNCIAS 24

# INTRODUÇÃO

Este projeto consiste em uma demonstração de automação residencial, onde se pode controlar desde a iluminação e os ventiladores, até o portão de uma casa.

Para isto, foi utilizada a plataforma Arduino, que consiste em uma placa com um controlador, que torna possível o controle através de uma comunicação com o computador.

Para isto foi desenvolvido um programa com uma interface gráfica, visando facilitar a interação e a usabilidade.

Este programa faz a leitura da interface e permite que o usuário possa interagir ajustando, por exemplo, o brilho da lâmpada e a velocidade de rotação de um ventilador. Estas informações são lidas e transmitidas para a placa através da interface de comunicação entre o computador e a placa.

# O ARDUINO

## 2.1 HISTÓRIA

O Arduino surgiu em 2005 na Itália. Criado por um professor chamado Massimo Banzi com o intuito de ensinar programação e eletrônica para seus alunos de design para que eles pudessem utilizar em seus projetos de arte, aumentando a interação e utilizando-se da robótica.

No entanto, além de ser difícil ensinar eletrônica, trabalhar com microcontroladores era uma tarefa árdua. Pois além de exigir um bom conhecimento de eletrônica, ainda exigia outros investimentos, como por exemplo: gravadores que precisam ser comprados ou construídos, isso aumentava o tempo e o custo.

Pensando nisso, Massimo Banzi e David Cuartielles decidiram criar sua própria placa. O aluno David Mellies foi o responsável pela linguagem de programação do Arduino.

Pelo fato de ser uma placa com vários componentes já integrados, ser aberto e com uma grande comunidade pela *Internet*. O Arduino revolucionou a Eletrônica e a Mecatrônica (especialmente a parte de Robótica).

## **2.2 O QUE COMPÕE O ARDUINO**

### 2.2.1 MICROCONTROLADOR

Pode ser entendido como um computador, porém com o objetivo de controlar apenas algo específico, por isso ele geralmente é encontrado embutido em um produto comercial.

Ele é dedicado, ou seja, feito para realizar apenas aquela função. Ao contrário dos computadores que podem realizar diversas tarefas.

Assim como o computador, ele geralmente executa um programa que é armazenado em uma memória. Porém, ao contrário dos computadores tradicionais que possuem uma memória que sempre muda o que está gravado conforme vamos executando outras tarefas, o programa é armazenado numa memória em que é possível realizar apenas sua leitura. Para modificar algo, devemos gravar novamente nela, apagando o que já estava gravado.

### 2.2.2 NÚMEROS BINÁRIOS (bits)

## 3.1 RECURSOS

O *Android* é composto de sistema operacional, um *middleware*  (camada intermediária de aplicação) e das aplicações.explique o que é *middleware*  melhor

Alguns recursos são:

* *Framework* que proporciona a reutilização e substituição de componentes.
* *Dalvik:* Uma máquina virtual que permite que o sistema isole os processos em diversas instâncias e melhora a autonomia do gerenciamento de memória virtual do dispositivo.
* Naveagador de Internet integrado, baseado no *WebKit*de código aberto.
* Otimizador de gráficos utilizando uma biblioteca 2D e outra 3D, baseada em *OpenGL ES* 1.0, tornando assim a qualidade visual do aplicativo tão atraente quando seu desempenho. Aceleração de *hardware* depende do dispositivo.
* Banco de dados SQLite integrado para guardar dados estruturados.
* Suporte multimídia para áudio, vídeo e formatos de imagem nos seguintes formatos: MPEG-4, H.264, MP3, AAC, JPG, PNG, GIF.
* Suporte para tecnologia GSM. (Depende do *hardware*).
* Suporte para *Bluetooth*, EDGE, 3G e WiFi. (Depende do *hardware*).
* Câmera, GPS, compasso, bússola e acelerômetro. (Dependem do *hardware*).
* IDE (Ambiente de desenvolvimento integrado), que inclui um emulador, ferramentas de depuração, memória e perfis de desempenho, plugin para a IDE *Eclipse*, facilitando assim a vida do desenvolvedor.



Figura 1 – Diagrama da Arquitetura do Android

A figura 1 tem de ter um link no texto exemplo: como pode ser visto na figura 1 a baixo onde o diagrama da arquitetura do android funciona ....

## 3.2 CAMADAS

Conforme mostrado na Figura 1, o sistema é composto de 5 camadas principais: a das aplicações, a de *framework* das aplicações, a de bibliotecas do sistema a de *runtime* e a de *kernel linux.*

### 3.2.1 CAMADA DE APLICAÇÕES

Contém todos os programas que são executados sobre o sistema operacional, inclusive os padrões que são: Cliente de e-mail, cliente de SMS e MMS, calendário, jogos, navegador, calculadora, bússola, contatos, entre outros.

Todos utilizam a linguagem de programação Java, porém o *layout* das aplicações é feito com a linguagem XML.

Com isso, entende-se que as aplicações são compostas de layout XML com os comandos em JAVA.

### 3.2.2 CAMADA DE FRAMEWORK DE APLICAÇÕES

Parte em que os desenvolvedores tem todo o acesso à API (Application Programming Interface ou Interface de programação de aplicativos) utilizada na camada de aplicações.

A vantagem de se utilizar uma API é que o conjunto de rotinas, padrões de utilização e funções dos aplicativos já estão estabelecidas por padrão nos arquivos XML. Cada linha representa uma função diferente. Com isso, usuário não se envolve em como o software foi implementado. Apenas tem acesso aos seus serviços.

No caso do Android, isto simplifica a reutilização de componentes, pois qualquer aplicação pode ter seus recursos publicados e outra aplicação pode fazer uso deles livremente.

Os principais serviços e sistemas inclusos nesta camada são:

* Conjunto de views que é composto de: Listas, grades, caixas de texto, botões, dentre outros, enfim, tudo necessário para se desevolver uma aplicação de forma gráfica.
* Fornecedores de conteúdos que tem como função facilitarem o acesso aos dados de outros aplicativos, como por exemplo, a lista de contatos, ou os seus próprios dados cadastrados.
* Gerenciador de recursos visuais que é responsável pelos gráficos, imagens e layouts da tela.
* Gerenciador de notificações que exibe os alertas de um aplicativo na tela. (Conceito semelhante à barra de notificações empregada no Microsoft Windows).
* Gerenciador de atividades que é responsável por gerenciar todo o ciclo de vida de uma aplicação e aperfeiçoar a navegação.

### 3.2.3 CAMADA DE BIBLIOTECAS

O Android possui um conjunto de bibliotecas C e C++ que são utilizadas por vários componentes do sistema. Algumas das principais são as seguintes:

* Biblioteca C: conjunto de *scripts* em C (libc), por ser um sistema baseado em Linux, ele implementa as mesmas funções desta biblioteca que são utilizadas em todos as distribuições Linux.
* Bibliotecas de mídia (Media Librarys): são bibliotecas baseadas na PacketVideo da OpenCore. Consiste em ferramentas para reproduzir e gravar os formatos mais populares deáudio e vídeo (Os formatos suportados, já foram citados no capítulo 3.1 em que falamos dos recursos).
* Surface Manager: responsável pelo acesso aos recursos gráficos 2D e 3D a partir de camadas de diversas aplicações.
* Lib WebCore: Navegador integrado ao sistema.
* SGL: Motor de gráficos 2D.
* Bibliotecas 3D:Baseada na OpenGL ES 1.0, trabalham com a aceleração 3D do dispositivo.
* Biblioteca FREETYPE: Contém as fontes do sistema.
* SQLite: Gerenciador de Banco de Dados relacional.

### 3.2.4 CAMADA RUNTIME DO SISTEMA

Runtime significa tempo de execução, ou seja, nesta camada está tudo o que é responsável pela execução do sistema. O Android implementa quase todas as funcionalidades das bibliotecas da linguagem de programação JAVA.

De forma semelhante à linguagem JAVA, toda a aplicação roda em uma instância de máquina virtual, que no caso é a Dalvik (Já citada no capítulo 3.1).

A Dalvik VM permite que várias VMs sejam executadas de forma eficente, isoladas em processos diferentes.

Os arquivos executáveis são os de extensão .dex, que foram otimizados para um menor e mais eficiente consumo de memória física do hardware, evitando assim também desperdício de bateria do dispositivo.

A máquina virtual executa classes compiladas por um compilador JAVA transformadas em .dex por meio de uma ferramenta chamada “dx”.

Dependendo da necessidade o Dalvik VM também pode invocar o kernel, caso seja necessário utilizar-se de funções extras ligadas diretamente ao hardware.

### 3.2.5 CAMADA KERNEL LINUX

O Androidé baseado no KernelLinux. O utilizando para integração e abstração com o hardware e utilização de recursos que o envolvam diretamente, tais como: Segurança, Gerenciamento de memória, Gestão de processos, Gestão de rede e drivers para componentes.

Até a versão 3.0 (Honeycomb) era baseado na versão 2.6 do Kernel Linux, agora, a partir da versão 4.0 (Ice Cream Sandwich), passou a ser baseado na versão 3.x (3 e superiores).

## 3.3 O DESENVOLVIMENTO ANDROID

O Android utiliza-se do conceito de SDK (Software Devolopment Kit ou Kit de Desenvolvimento de Software) que contém as ferramentas necessárias para o desenvolvimento facilitado e prático das aplicações.

A IDE mais utilizada para o desenvolvimento é a Eclipse, pois possui plugin oficial do Google, que permite utilizar-se do conceito RAD (Rapid Application Devolopment ou Desenvolvimento Rápido de Aplicação), que foi popularizada com a IDE e Linguagem de programação Delphi, desenvolvida inicialmente pela Borland e atualmente pela Embarcadero.

A vantagem de utilizar este padrão é que para o desenvolvimento podem ser utilizadas APIs e frameworks já pré-definidos e o desenvolvedor focar-se essencialmente na análise e na parte do design com a devida participação do usuário final. Isso evita possíveis transtornos e mudanças repentinas.

Outra semelhança com o Delphi está na orientação a eventos. Os eventos podem ser chamados por métodos em classes escritas em Java.

Para ter várias telas, o correto é criarmos mais de um layout, ou seja, mais de um arquivo XML.

No Android todas as subclasses e métodos devem ser derivados da classe principal denomiada *Activity* que serve para gerenciar a interface com o usuário. Um bom programa com várias telas tem mais de uma Activity que são chamadas por comandos Java.

As *Activitys* também podem estar contidas em uma *ListActivity*, que nada mais é que uma lista de *Activitys*.

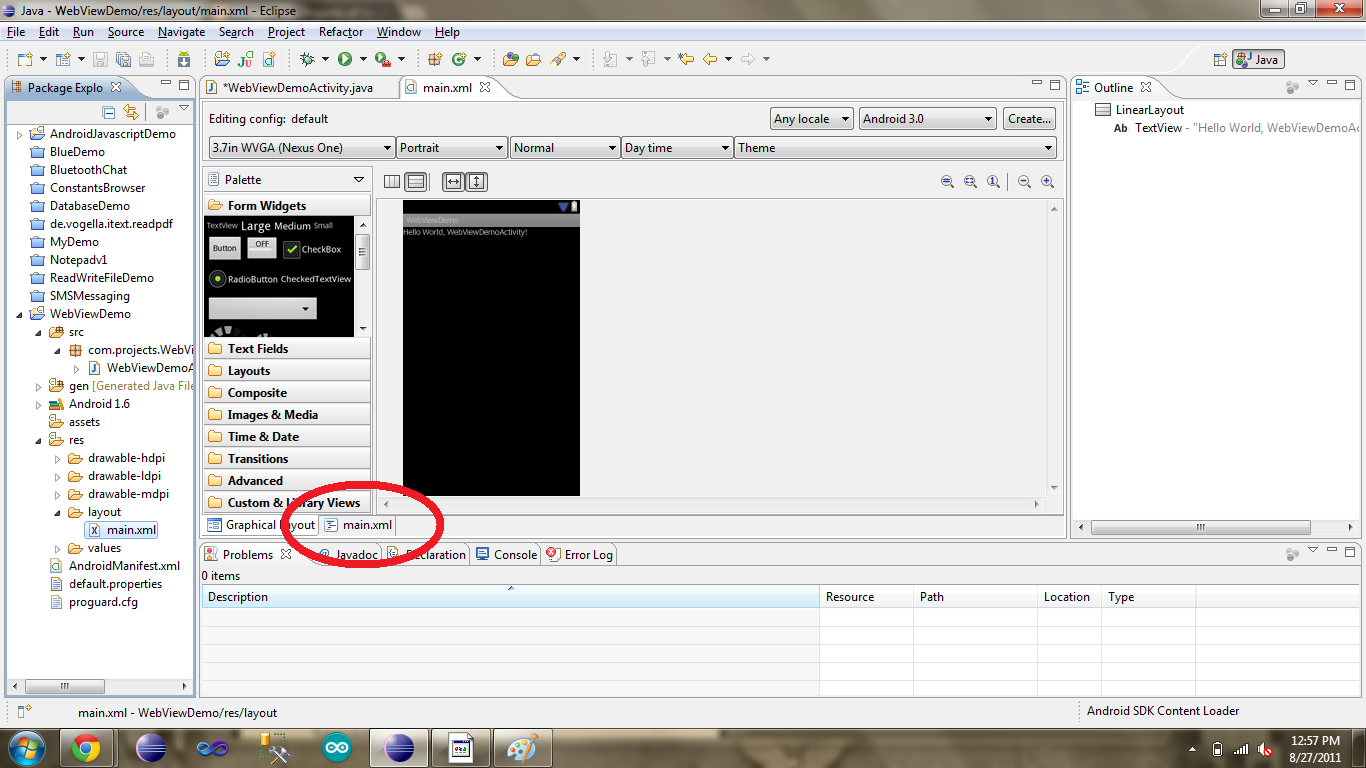


Figura 2 – Ambiente de Desenvolvimento Eclipse com Plugin ADT (AndroidDevelopment).

Link da imagem com o texto

### 3.3.1 ANDROID VIRTUAL DEVICE (AVD)

Para fazer o teste do aplicativo, não é necessário que você tenha um dispositivo, no SDK já vem incluso a possibilidade de você criar dispositivos virtuais para fazer o teste de sua aplicação. Podem ser criados dispositivos de qualquer versão a partir da versão 1.5 e de várias resoluções de tela.

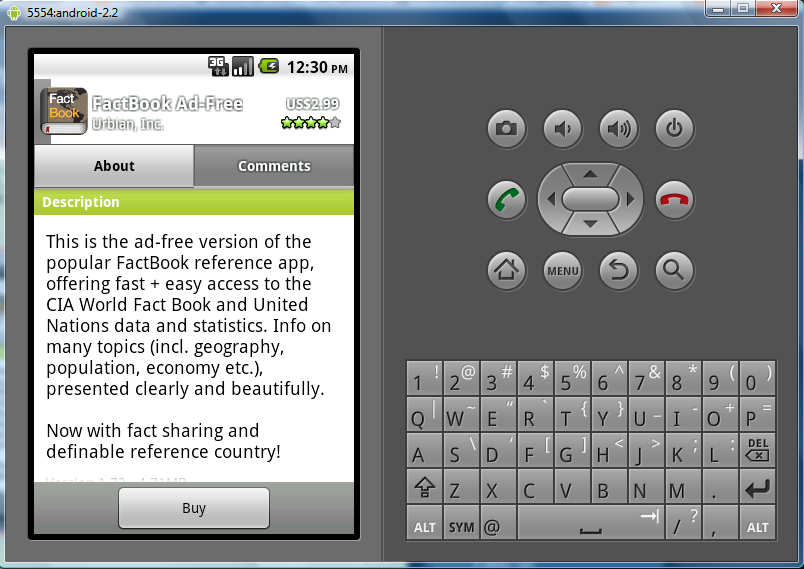


Figura 3 – Android Virtual Device (AVD). Link da imagem com o texto

# O SERVIDOR

Utilizaremos um servidor web Apache com um banco de dados integrado para armazenar os pedidos que virão diretamente dos dispositivos dos vendedores.

## 4.1 APACHE

O Apache é um servidor web de código-aberto desenvolvido pela Fundação Apache que possui, como principais características:

* Suporte a scripts cgi, tendo suporte a linguagens, como: PHP, Perl, Shell Script e ASP.
* Suporte para autorização de acesso, ou seja, é possível definir quem pode e quando deve acessar o servidor, aumentando assim a segurança.
* Autenticação com criptografia *Crypto* ou MD5.
* Negociação de conteúdo, que permite que seja escolhido o idioma que o navegador requisitar.
* Suporte a tipos mime.
* Logs personalizáveis.
* Notificações através de mensagens de erro.
* Virtualização de *hostings* (Abrir duas ou mais páginas com endereços diferentes e/ou portas diferentes a partir do mesmo processo ou usar mais de um processo para controlar o mesmo endereço.
* Suporte a proxy e redirecionamentos com URLs alocadas para endereços internos.
* Criptografia SSL e demais certificados digitais.
* Módulos DSO (*DynamicSharedObjects*), permitindo assim, adicionar ou remover recursos sem precisar recompilar o programa.

## 4.2 PHP

Do inglês PHP *HiperText Processor* é uma linguagem de programação de código-aberto com foco para utilização de *scripts* ao lado de um servidor. Bastando para isso, ter um interpretador instalado e um browser de internet para acesso.

Dentre as principais aplicações estão: a coleta de dados e armazenamento em uma base de dados, a geração de conteúdo dinâmico e o envio e recebimento de cookies.

Em nosso projeto coletaremos os dados e armazenaremos em uma base de dados SQL, gerenciada pelo SGBD MySQL.

## 4.3 MYSQL

O MySQL é um SGBD (Sistema de Gerenciamento de Banco de Dados) de código aberto desenvolvido inicialmente por David Axmark, Allan Largsson e Michael “Monty” Widenius que compunham a MySQL AB que hoje pertence a Oracle Corporation.

As principais vantagens de utilizá-lo são:

* Facilidade de ser portável para qualquer plataforma atual.
* Forte compatibilidade para *drivers* ODBC, JDBC e .NET e módulos das linguagens: *Delphi*, Java, C/C++, C#, *Visual Basic*, *Python*, *Perl*, PHP, ASP e *Ruby*.
* Estabilidade e desempenho excelentes.
* Exige poucos recursos.
* Fácil uso.
* Código-aberto.
* Controle de transações.
* Suporte a *Triggers*.
* Suporte a *Cursors*.
* *Stored Procedures* e *Functions*.
* Replicação facilmente configurável.
* Diversas interfaces gráficas fornecidas pelos desenvolvedores, como por exemplo, a Workbench, já fornecida.

## 4.4 XAMPP

Visando facilitar o desenvolvimento, optamos por utilizar o XAMPP, por ser um pacote bem completo no que se refere a servidores *web*. Tendo incluso os já mencionados PHP, Apache e MySQL. Seus recursos facilitam, pois além de já ter vir praticamente tudo embutido, tudo pode ser utilizado via browser e graficamente, o que economiza tempo, tornando assim o processo mais ágil.

## 4.5 A INTEGRAÇÃO COM O ANDROID

Para que o aplicativo acesse o servidor HTTP Apache, é necessário utilizar-se de uma classe escrita em Java com diversos métodos das bibliotecas Apache e Java que ficam responsáveis, pela conexão, armazenamento, lidar com as entidades, dentre outras atividades. Nesses métodos é que implementados os parâmetros já escritos em PHP anteriormente.

# CONCLUSÃO

Com esta nova era de dispositivos móveis e portáteis foi possível substituir métodos antigos, o que garantiram a segurança e maior agilidade. Este projeto buscou contribuir com a melhoria dos sistemas que já são adotados, pois utiliza-se de tecnologias de código-aberto e facilmente implementadas e adaptadas. Pode-se ver a enorme integração do *Android* com a *Web* e os servidores HTTP em qualquer dispositivo que o suporte. Isso gerou uma plataforma sólida, confiável e robusta. A integração de todas essas tecnologias, não só aumenta a credibilidade do fornecedor e /ou distribuidor, como também aumenta o reconhecimento dele, frente ao mercado. Visto que é exatamente isto tudo que o mercado busca: Soluções fáceis, práticas, duradouras, seguras e facilmente adaptáveis às novas tendências de nosso século XXI.

# REFERÊNCIAS

PISKE, RODOLFO OTÁVIO, SEIDEL, ANDRÉ FÁBIO. **RapidApplicationDevelopment.** Centro Universitário Positivo – UnicenP– Curitiba – PR. Disponível em:<http://www.angusyoung.org/arquivos/artigos/rad.pdf>. Acesso em: 19 de Novembro de 2012.

SILVEIRA, FELIPE. **Desenvolvendo para Android**. Disponível em: <http://www.felipesilveira.com.br/desenvolvendo-para-android/>. Acesso em 29 de Outubro de 2012.

JOBSTRAIBIZER, FLÁVIA. **Criação de Aplicativos para celulares com Google Android.**– São Paulo – SP – Digerati Books, 2009.

ALECRIM, EMERSON. **Conhecendo o Servidor Apache (HTTP Server Project).** Disponível em: <http://www.infowester.com/servapach.php>. Acesso em: 20 de novembro de 2012.

FURLAN PAULO, MARCOS. **PHP.** Disponível em: <http://serverapostilando.com/tutorials2/3300_php.zip>. Acesso em: 19 de novembro de 2012.