DOCUMENTO PADRÃO PARA DESENVOLVIMENTO DE SITES

SOBRE OS SITES

- Os sites desenvolvidos pela empresa utilizam a linguagem de programação **PHP** orientado a objetos (versão 5.3.0 ou maior), banco de dados **MySQL**, *framework* JavaScript **jQuery** (versão 1.8.2 ou maior) e estrutura de URLs amigáveis (*mod_rewrite*).
- É utilizado o *framework*/CMS interno da empresa, o **Osiris**.
- Internamente, os sites são configurados no Apache através de *virtual hosts* com endereço "*http://site.loc*" **(tutorial em anexo no fim do documento)**.

SOBRE A ESTRUTURA DO SISTEMA

O sistema **Osiris** é organizado, minimamente, na seguinte estrutura de pastas:

- /admin
  - /assets
    - /css
    - /js
    - /themes
  - /conf
  - /images
  - /inc
  - /modules
    - /site
    - /system
  - /pages
  - /reports
    - /site
    - /system
- /app
  - /ajax
  - /api
  - /assets
    - /compress
      - /admin
      - /site
    - /css
    - /gallery
    - /images
    - /js
      - /jquery
        - /plugins
        - /ui
  - /cache
  - /conf
    - /settings
  - /core
    - /class
    - /inc
      - /ckeditor
      - /ckfinder
      - /facebook
      - /image-upload
      - /pagseguro
      - /phpmailer
      - /securimage
      - /twitter
      - /uploadify
    - /util
  - /lang
  - /rss
- /class
  - /dao
- /database
- /doc

- /site
    - /assets
        - /css
        - /js
    - /conf
    - /inc
    - /media
        - /flash
        - /images
    - /pages
- /temp
- /uploads
    - /editor
    - /images

Detalhamento das pastas e arquivos do sistema

/:
- Raiz do sistema, contendo 2 arquivos:
  - **.htaccess** – Contém as *rewrite rules* para a utilização de URLs amigáveis no site (não são necessárias alterações, pois todo esse controle é realizado através do site).
  - **robots.txt** – Contém as instruções aos robôs de buscadores.

/admin:
- Contém a área administrativa do sistema.

/admin/assets:
- Contém os recursos utilizados pela área administrativa, como CSS e JavaScript.

/admin/assets/css:
- Contém os arquivos de CSS da área administrativa:
  - **bar.css** – Estilo da barra administrativa que é exibida no site quando se está logado na área administrativa.
  - **form.css** – Estilo dos formulários.
  - **pagination.css** – Estilo da paginação de registros.
  - **styles.css** – Estilos customizados.
  - **system.css** – Estilo da estrutura da área administrativa, etc.
  - **table.css** – Estilo das tabelas.

/admin/assets/js:
- Contém os arquivos de JavaScript da área administrativa:
  - **ready.js** – Script a ser executado assim que o sistema é carregado.

/admin/assets/themes:
- Contém os temas da área administrativa, dentre eles azul, verde, rosa, roxo, vermelho e amarelo.

/admin/conf:
- Contém os arquivos de configuração da área administrativa:
  - **routes.ini** – Arquivo de configuração que contém as rotas customizadas da área administrativa.

/admin/images:
- Contém as imagens utilizadas na área administrativa.

/admin/inc:

- Contém os arquivos de estrutura da área administrativa para inclusão:
    - **bar.php** – Barra administrativa exibida no site quando se está logado na área administrativa.
    - **footer.php** – Rodapé da área administrativa.
    - **header.php** – Cabeçalho da área administrativa.
    - **menu.php** – Menu exibido da lateral esquerda da área administrativa.
    - **page-title.php** – Título do módulo/pacote atual.
    - **sidebar.php** – Barra lateral direita com a lista de registros do módulo atual.

/admin/modules:

- Contém os módulos da área administrativa divididos entre as pastas "*site*", que contém os módulos que estão relacionados ao site; e "*system*", que contém os módulos que estão relacionados ao sistema.
- Os módulos são divididos em pacotes, que são pastas no padrão *"_package.**"*, onde o asterisco (**) indica o nome do pacote. Os pacotes contém o arquivo:
    - **lang.ini** – Arquivo que contém o nome do pacote em diferentes idiomas.

- As pastas dos módulos dentro dos pacotes possuem, em geral, os seguintes arquivos e pastas:
    - **/lang** – Pasta que contém os arquivos de configuração dos idiomas disponíveis para o módulo.
    - **icon-large.png** – Ícone no formato 32px x 32px que representa o módulo.
    - **Icon-small.png** – Ícone no formato 16px x 16px que representa o módulo.
    - **list.php** – Arquivo que lista os registros do módulo em forma de tabela na coluna lateral.
    - **main.php** – Arquivo que contém o formulário principal para edição de registros do módulo.

/admin/pages:

- Contém as páginas da área administrativa:
    - **404.php** – Página exibida ao encontrar um erro 404 (página não encontrada).
    - **home.php** – Página inicial.
    - **login.php** – Página que contém o formulário de login.
    - **logout.php** – Página que realiza o logout.

/admin/reports:

- Contém os relatórios do sistema divididos entre as pastas "*site*", que contém os relatórios que estão relacionados ao site; e "*system*", que contém os relatórios que estão relacionados ao sistema.

/app:

- Contém o sistema Osiris propriamente dito.

/app/ajax:
- Contém os arquivos utilizados para chamadas AJAX realizadas pelo sistema:
  - **ajax.js** – JavaScript que contém funções úteis para chamadas AJAX.
  - **load-more.php** – Realiza o carregamento de mais registros em uma lista.
  - **load-options.php** – Realiza o carregamento de opções utilizadas em um elemento SELECT.
  - **post-data.php** – Realiza um processamento de dados no banco de dados e retorna uma mensagem.

/app/api:
- Contém o arquivo **control.php**, utilizado para realizar a chamada correta à API.

/app/assets:
- Contém os recursos utilizados pelo sistema, como CSS e JavaScript.

/app/assets/compress:
- Contém os recursos do sistema comprimidos em um só arquivo.

/app/assets/compress/admin:
- Recursos comprimidos da área administrativa:

/app/assets/compress/site:
- Recursos comprimidos do site:
  - **compress.css** – Conteúdo dos arquivos CSS.
  - **compress.js** – Conteúdo dos arquivos JavaScript.
  - **css_files** – Contém as informações de cada arquivo CSS comprimido, como caminho e data de modificação.
  - **js_files** – Contém as informações de cada arquivo JavaScript comprimido, como caminho e data de modificação.

/app/assets/css:
- Contém os arquivos CSS padrão do sistema:
  - **common.css** – Regras comuns à área administrativa e o site.
  - **dialog-container.css** – Regras das caixas de diálogo do sistema.
  - **ie-warning.css** – Regras da barra de alerta exibida quando o sistema é acessado através do navegador Internet Explorer de versão 8 ou menor.
  - **reset.css** – Regras que limpam a formatação padrão dos navegadores.
  - **show-more.css** – Regras para o botão que exibe mais registros em uma lista através de AJAX.

/app/assets/gallery:
- Contém os arquivos de CSS e imagens da galeria do sistema.

/app/assets/images:
- Contém as imagens utilizadas pelo sistema.

/app/assets/js:
- Contém os arquivos JavaScript utilizados pelo sistema:
  - **browser-warning.js** – Exibe a barra de alerta de acesso ao sistema através de versões antigas do navegador Internet Explorer.
  - **common.js** – Procedimentos padrão comuns à área administrativa e o site.

/app/assets/js/jquery:
- Contém os arquivos relacionados à biblioteca jQuery:
  - **jquery-1.8.2.min.js** – Biblioteca jQuery versão 1.8.2.

/app/assets/js/jquery/plugins:
- Contém os plugins jQuery utilizados pelo sistema:
  - **/autocomplete** – Autopreenchimento de campos de um formulário.
  - **/fancybox** – Exibição de conteúdo em caixa flutuante.
  - **jquery.bxSlider.pack.js** – Slider para deslizar conteúdos de uma caixa.
  - **jquery.color.pack.js** – Animação com cores.
  - **jquery.cookie.pack.js** – Manipulação de *cookies*.
  - **jquery.cursor.js** – Manipulação do cursor em campos de texto.
  - **jquery.cycle.all.pack.js** – Alterna conteúdos de uma caixa utilizando diversos efeitos visuais.
  - **jquery.highlight.min.js** – Destaca (*highlight*) palavras em um texto.
  - **jquery.hoverIntent.min.js** – Controla tempo de ação do evento *hover*.
  - **jquery.limit.min.js** – Limita a quantidade de caracteres em um campo do tipo TEXTAREA.
  - **jquery.livequery.min.js** – Realiza funções em elementos dinâmicos.
  - **jquery.maskedinput.min.js** – Exibe máscaras customizadas em campos de um formulário.
  - **jquery.maskMoney.pack.js** – Exibe máscara monetária em campos de um formulário (estilo internet banking).
  - **jquery.numeric.pack.js** – Permite somente números em campos de um formulário.
  - **jquery.placeholder.min.js** – Exibe textos descritivos em campos de um formulário (desaparecem quando um texto é escrito).

- **jquery.scrollTo.min.js** – Realiza *scroll* na página utilizando efeitos visuais.

/app/assets/js/jquery/ui:
- Contém a biblioteca e os plugins jQueryUI:
  - **/datepicker** – Calendário.
  - **/sortable** – Re-ordenação de elementos de uma lista.

/app/assets/js/password-strength-meter:
- Contém um modulo de cálculo de força de senha:
  - **script.js** – Script que realiza o cálculo.
  - **styles.css** – Estilos CSS da barra de progresso da força de senha.

/app/cache:
- Contém os arquivos temporários de cache gerados pela classe **Cache**.

/app/conf:
- Contém os arquivos de configuração do sistema:
  - **bootstrap.php** – Rotinas de inicialização do sistema.
  - **connection.php** – Realiza a conexão com o banco de dados.
  - **security.php** – Realiza o tratamento de segurança do sistema.
  - **session.php** – Controla o tempo de expiração das sessões.

/app/conf/settings:
- Contém os arquivos com as definições de dados do sistema:
  - **database.php** – Dados de conexão com o banco de dados.
  - **dynamic.php** – Dados dinâmicos definidos no módulo de configurações do sistema na área administrativa.
  - **static.php** – Definições estáticas do sistema.

/app/core:
- Contém o núcleo do sistema.

/app/core/class:
- Contém as classes necessárias para o funcionamento do sistema.
- Os nomes dos arquivos são iguais aos nomes das classes no código, sempre começando com letra maiúscula:
  - **Admin.php** – Manipula sessões, login e controle de acesso de administradores do sistema.
  - **AJAX.php** – Realiza operações via AJAX.
  - **API.php** – Provê uma API para as classes DAO.
  - **ArrayUtil.php** – Manipulação de vetores.
  - **Assets.php** – Controla os componentes CSS e JavaScript do sistema.

- **Bitly.php** – Encurta URLs utilizando o serviço bit.ly.
- **Browser.php** – Detecta o navegador utilizado pelo usuário e exibe avisos de segurança.
- **Cache.php** – Manipula *cache* de dados.
- **Compressor.php** – Realiza compressão/descompressão de arquivos em formato ZIP.
- **Config.php** – Manipula configurações do sistema.
- **Control.php** – Realiza o controle de URLs do sistema.
- **Cookie.php** – Manipula *cookies*.
- **Database.php** – Realiza o gerenciamento de banco de dados.
- **DatabaseObject.php** – Controla objetos DAO.
- **DateFormat.php** – Contém métodos diversos para formatação de data/hora.
- **Download.php** – Controla o download de arquivos.
- **Email.php** – Responsável pelo envio de e-mails do site, contendo estrutura padrão. Utiliza a classe PHPMailer como base.
- **Facebook.php** – Realiza a integração do sistema com o Facebook.
- **Files.php** – Manipula arquivos.
- **Flash.php** – Exibe conteúdos Flash.
- **Flickr.php** – Utiliza a API do Flickr para manipulação de fotos do serviço.
- **Folders.php** – Manipula pastas.
- **Form.php** – Responsável pela criação de formulários, contendo diversos tipos de campos e de validação.
- **Format.php** – Contém métodos diversos para formatação de textos e valores.
- **Gallery.php** – Utilizada para exibição de galerias de fotos com *zoom* e legenda.
- **Googl.php** – Encurta URLs utilizando o serviço Goo.gl.
- **GoogleAnalytics.php** – Manipula dados do Google Analytics.
- **GoogleChart.php** – Utiliza a API do Google Chart para geração de gráficos.
- **GoogleMaps.php** – Carrega imagens do mapa através da API do Google Maps.
- **GooglePlus.php** – Realiza a integração do sistema com o Google Plus.
- **HTML.php** – Manipula HTML.
- **Image.php** – Manipula imagens.
- **Language.php** – Controla multilinguagem no sistema.
- **Message.php** – Manipula as mensagens (sucesso, erro, informação) do sistema.
- **Paginator.php** – Realiza a paginação de registros.
- **PagSeguro.php** – Controle de pagamentos através do serviço PagSeguro.
- **PayPal.php** – Controle de pagamentos através do serviço PayPal.

- **Regex.php** – Contém expressões regulares diversas.
- **Report.php** – Responsável pela geração de relatórios.
- **Request.php** – Manipula parâmetros GET e POST.
- **Router.php** – Controla as rotas da área administrativa e do site.
- **RSS.php** – Realiza o controle e exibição de RSS.
- **Security.php** – Controla a segurança do sistema.
- **Server.php** – Manipula informações sobre o servidor.
- **Session.php** – Manipula sessões.
- **Sitemap.php** – Gera mapa do site.
- **SitemapNews.php** – Gera *sitemap* de notícias para o Google News.
- **Social.php** – Métodos diversos de redes sociais.
- **System.php** – Manipula os módulos do sistema administrativo.
- **Table.php** – Responsável pela geração de tabelas de registros com métodos diversos, como formatação de dados, ordenação, etc.
- **Tools.php** – Métodos diversos.
- **Twitter.php** – Realiza a integração do sistema com o Twitter.
- **URL.php** – Manipula URLs.
- **User.php** – Manipula sessões e login de usuários.
- **Validator.php** – Realiza validação de dados.
- **YouTube.php** – Métodos para utilização de *widgets* do YouTube.

/app/core/inc:

- Contém os módulos de inclusão do sistema:

- **/ckeditor** – CKEditor, editor WYSIWYG utilizado no site.
- **/ckfinder** – CKFinder, integrado ao CKEditor, que realiza o *upload* de imagens e arquivos no CKEditor.
- **/facebook** – SDK do Facebook.
- **/image-upload** – Upload e recorte de imagens utilizado na classe **Form**.
- **/pagseguro** – Biblioteca de pagamentos do PagSeguro, utilizada pela classe **PagSeguro**.
- **/phpmailer** – Biblioteca PHPMailer, utilizada pela classe **Email** e responsável pelo envio de e-mails.
- **/securimage** – Biblioteca Securimage, responsável pela geração de *captcha* utilizado pela classe **Form**.
- **/twitter** – API do Twitter.
- **/uploadify** – Biblioteca Uploadify, responsável pelo *upload* de arquivos utilizado pela classe **Form**.

/app/core/util:

- Contém arquivos úteis utilizados pelo sistema:

- **download.php** – Realiza o download de arquivos.
- **option-add.php** – Caixa de diálogo que insere novas opções em um elemento SELECT.

- **password-change.php** – Caixa de diálogo que altera uma senha.
- **terms-of-use.php** – Caixa de diálogo que exibe os termos de uso do sistema cadastrados no módulo de configurações da área administrativa.
- **thumb.php** – Exibe imagens redimensionadas *on the fly*.

/app/lang:
- Contém os arquivos INI que contém os idiomas disponíveis no sistema:
  - **en.ini** – Idioma inglês.
  - **pt-br.ini** – Idioma português (Brasil).

/app/rss:
- Contém o arquivo **control.php**, utilizado para controlar e gerar os *feeds* RSS do site.

/class:
- Contém as classes relativas ao site:
  - **SiteSearch.php** – Realiza busca geral nas classes DAO do sistema.

/class/dao:
- Contém classes do tipo Database Access Object que extendem da classe **DatabaseObject**, que manipulam registros de tabelas do banco de dados:
  - **City.php** – Registro de cidade.
  - **Country.php** – Registro de país.
  - **Page.php** – Registro de páginas dinâmicas do site.
  - **State.php** – Registro de estados de país.

/database:
- Contém arquivos referentes ao banco de dados do sistema, dentre eles:
  - **model.mwb** – Modelo entidade-relacionamento do banco de dados do aplicativo MySQL Workbench.
  - **script.sql** – Script SQL de geração do banco de dados.

/doc:
- Contém documentos referentes ao sistema, dentre eles:
  - **osiris.chm** – Arquivo de ajuda contendo a documentação de todas as classes utilizadas pelo sistema.
  - **readme.pdf** – Este próprio arquivo, contendo as informações, regras e padrões do sistema.

/site:
- Contém o site propriamente dito e possui, no mínimo, os seguintes arquivos:
  - **index.php** – Página principal do site, onde são configuradas as outras páginas, inclusos os arquivos CSS e JavaScript e definido o cabeçalho HTML das páginas.

- **sitemap.php** – Página que monta o mapa do site em formato XML (acessada através do endereço "/sitemap.xml").

/site/assets:
- Contém os recursos utilizados pelo site, como CSS e JavaScript.

/site/assets/css:
- Contém os arquivos de CSS do site:
  - **form.css** – Estilo dos formulários.
  - **pagination.css** – Estilo da paginação de registros.
  - **styles.css** – Estilo da estrutura do site e seu conteúdo.

/site/assets/js:
- Contém os arquivos JavaScript utilizados exclusivamente no site:
  - **ready.js** – Contém os procedimentos que devem ser executados após o carregamento da página.

/site/conf:
- Contém os arquivos de configuração do site:
  - **routes.ini** – Arquivo de configuração que contém as rotas customizadas do site.

/site/inc:
- Contém os arquivos de estrutura do site para inclusão:
  - **footer.php** – Rodapé do site.
  - **header.php** – Cabeçalho do site.

/site/media:
- Contém os arquivos de mídia do site.

/site/media/flash:
- Contém os filmes Flash utilizados no site.

/site/media/images:
- Contém as imagens utilizadas no site.
  - **/email/header.jpg** – Imagem de 600px x 120px utilizada como cabeçalho dos e-mails enviados pelo site com o corpo padrão que é configurado na classe **Email**.
  - **/rss/logo.png** – Imagem 150px x 150px contendo a logo do site utilizada para ilustrar o *feed* RSS.
  - **favicon.ico** – Ícone 16px x 16px utilizado como ícone do site.

/site/pages:

- Contém as páginas do site:
    - **404.php** – Página exibida ao encontrar um erro 404 (página não encontrada).
    - **db.php** – Página que exibe o conteúdo de uma página dinâmica criada no módulo de páginas da área administrativa.
    - **home.php** – Página inicial.
    - **search.php** – Página que realiza a pesquisa em todo o site através da classe **SiteSearch**.

/temp:

- Pasta onde são armazenados arquivos temporários do sistema.

/uploads:

- Pasta onde é armazenado todo o conteúdo enviado para o site através de campos de *upload*, que devem ser divididos em pastas de acordo com a categoria.

/uploads/editor:

- Pasta onde são armazenados os arquivos enviados pelo editor *CKEditor*.

/uploads/images:

- Pasta onde são armazenadas todas as imagens enviadas para o site através de campos de *upload*, que devem ser divididas em pastas de acordo com a categoria.

SOBRE O CÓDIGO

- O código deve ser limpo, indentado de forma correta (com TABs, sendo o TAB com tamanho de 4 espaços) e documentado.
- Utilizar orientação a objetos sempre que possível.
- Sempre pensar na melhor maneira de escrever um trecho de código ou função, economizando linhas e pensando no reaproveitamento do mesmo.
- Comentar as classes, métodos e funções no código PHP utilizando os padrões do PHPDoc.
- Em uma consulta SQL, utilizar as palavras reservadas em letras maiúsculas e o restante em letras minúsculas:
  - `SELECT u.id, u.name FROM user u WHERE u.email = "user@mail.com" ORDER BY u.birth_date LIMIT 0,5;`

- Não dar espaços entre parênteses e seu conteúdo e chaves:
  - `if ( $b > 5 ) {` **(errado)**
  - `if($b > 5){` **(certo)**

- Abrir chaves na mesma linha da instrução:
  - `if($b > 5)` **(errado)**
    ```
    {
        echo 'B maior que 5';
    }
    ```
  - `if($b > 5){` **(certo)**
    ```
        echo 'B maior que 5';
    }
    ```

- Dar espaços entre variáveis/valores e operadores/vírgulas:
  - `array(0=>'Valor 0',1=>'Valor 1',2=>'Valor 2');` **(errado)**
  - `array(0 => 'Valor 0', 1 => 'Valor 1', 2 => 'Valor 2');` **(certo)**

- Utilizar termos em **inglês** para nomes de variáveis, funções, classes e métodos do PHP e em classes e IDs do CSS. A mesma regra deve ser seguida para os nomes de todos os arquivos do site, como imagens, flash, scripts, etc.
- Indentar código dentro do bloco PHP:
  - ```
    <?php
        echo 'Olá mundo!';
    ?>
    ```

- Os nomes de variáveis, funções, classes e métodos do PHP, e classes e IDs do CSS devem ser completamente minúsculos e com espaços substituídos por "_". A mesma regra deve ser seguida para os nomes de todos os arquivos do site, como imagens, flash, scripts, etc; porém os espaços devem ser substituídos por "-":
  - `$default_value = 'Valor da variável';`
  - `header-background.jpg`
    ```

- Os nomes de constantes devem ser completamente maiúsculos:
  - `define('CONSTANT', 'Valor da constante');`

- Utilizar aspas simples (') externamente e aspas duplas (") internamente em *strings*:
  - `echo '<div id="content"></div>'`

- Não abrir e fechar chaves { } quando houver apenas uma linha de instrução dentro da condição:
  - ```
    if($b > 5)
        $a = 'B maior que 5';
    ```
- Utilizar o operador ternário sempre que possível:
  - `$a = ($b > 5) ? 'B maior que 5' : 'B menor ou igual a 5';`

- Não utilizar *short tags* do PHP:
  - `<?=$var?>` **(errado)**
  - `<?php echo $var; ?>` **(certo)**

- Não dar espaço entre a concatenação de *strings* no PHP e dar espaço entre a concatenação de *strings* no JavaScript:
  - `echo 'Total de '.$total.' registros';`
  - `$('#' + id).css('display', 'block');`

- As chamadas `echo`, `include` e `require` não são funções, por isso não devem ser utilizados parênteses em sua utilização, mas espaço:
  - `include('page.php');` **(errado)**
  - `include 'page.php';` **(certo)**

- Indentar o conteúdo dos *cases* dos blocos *switch-case*, além de quebrar uma linha entre os *cases*:
  - ```
    switch($i){
        case 1:
            echo 'I igual a 1';
            break;

        case 2:
            echo 'I igual a 2';
            break;
    }
    ```

- Estar sempre atento ao SEO (Search Engine Optimization), no mínimo, colocando o atributo *alt* em imagens e *title* sempre que necessário, definindo as meta tags HTML *keywords* (palavras-chave) e *description* (descrição) do site, cadastrar no Google Analytics, etc.
- Evitar a utilização de estilos CSS *inline* (atributo *style*).

- Indentar o código CSS de acordo com a hierarquia dos elementos:
  - ```
    .myclass{
          font-size: 12px;
          color: #000;
    }
          .myclass span{
                font-weight: bold;
                color: #FF0000;
          }
    ```
- Comentar e agrupar o código CSS de acordo com cada área do site:
  - ```
    /*-- Notícias --*/

    ul.news-list{
          list-style: none;
    }
          ul.news-list li{
                color: #333;
          }

    /*-- Blog --*/

    .post{
          font-family: Arial, sans-serif;
    }
    ```

- Fechar os elementos HTML únicos, dando espaço entre o último atributo e a barra "/":
  - ```
    <img src="image.jpg" />
    ```

- Inserir 1 linha em branco para separar os blocos de código PHP dos blocos de código HTML:
  - ```
    <?php
          echo 'Olá mundo!';
    ?>

    <div id="content">
          Conteúdo
    </div>
    ```

- Inserir 1 linha em branco para separar blocos de código HTML:
  - ```

          Cabeçalho

          Conteúdo

          Rodapé

    ```
    ```

SOBRE O BANCO DE DADOS

- Os nomes dos bancos de dados, das tabelas e dos campos das tabelas devem ser completamente minúsculos, sem acento e com espaços substituídos por "_".
- Os nomes das tabelas do site devem possuir um prefixo de, no máximo, 4 letras seguido de "_" para distingui-las das tabelas do sistema (que possuem o prefixo "sys") e de configuração (que possuem o prefixo "conf"):
    - blog **(errado)**
    - ecad_blog **(certo)**

- Os nomes das tabelas devem estar sempre no singular:
    - sys_users **(errado)**
    - sys_user **(certo)**

- Utilizar a *engine* **InnoDB** para armazenamento das tabelas.
- Utilizar o conjunto de caracteres (*collation*) **utf8_general_ci** nas tabelas.

SOBRE A ESTRUTURA DE URLs

Tomando como exemplo o endereço do site como http://www.site.com, se deseja criar uma página de URL estática (ex.: http://www.site.com/noticias) deve-se utilizar o arquivo de configuração **routes.ini** do site e acrescentá-la como uma nova entrada, utilizando os atributos especificados.

Porém, se deseja que o sistema reconheça as URLs dinâmicas das notícias como válidas (ex.: http://www.site.com/noticias/slug-da-noticia), deve-se criar uma classe DAO para notícia (ex.: News.php) com, no mínimo, o seguinte código:

```
<?php
	class News extends DatabaseObject{
		const TABLE_NAME = 'sys_news';
		const BASE_PATH = '/noticias/';
		const PATH_SIZE = 2;

		protected $id;
		protected $title;
		protected $text;
		protected $slug;
		protected $url;

		public function load($id){
			if($record = $this->load_data($id)){
				$this->url = self::BASE_PATH.$record->slug;

				return true;
			}

			return false;
		}

		public static function check_url($url){
			global $db;

			$url_pieces = parent::get_current_url_pieces($url);
			$slug = $url_pieces[0];

			$db->query('SELECT id, title FROM self::TABLE_NAME
WHERE slug = "'.$slug.'"');
			if($db->row_count()){
				$news = $db->result(0);
				return array('title' => $news->title, 'file' =>
'news-details.php', 'reg_id' => $news->id);
			}

			return false;
		}
	}
?>
```

Detalhamento do código

- A cada URL que é acessada no sistema e que não está contida nas rotas estáticas da classe **Router**, a classe **Control** faz uma chamada ao método `check_url` de todas as classes DAO (que extendem **DatabaseObject**) de acordo com os valores das constantes `BASE_PATH` e `PATH_SIZE` das mesmas:
    - Ex.: Ao acessar o endereço http://www.site.com/noticias/slug-da-noticia são verificadas as classes DAO que possuem `PATH_SIZE` = 2 e `BASE_PATH` = "/noticias/" ou `BASE_PATH` = "/".

- A constante `TABLE_NAME` define o nome da tabela do banco de dados cuja classe DAO manipula.
- A constante `BASE_PATH` define o caminho base fixo, a partir do endereço do site, até o *slug* da notícia.
- A constante `PATH_SIZE` indica o tamanho da URL a partir do endereço base do site até o *slug* da notícia:
    - Ex.: http://www.site.com/noticias/slug-da-noticia.

- O método `check_url` é chamado automaticamente pela classe **Control** e recebe como parâmetro a URL acessada para verificar se a mesma é uma notícia válida, ou seja, se o endereço acessado é http://www.site.com/noticias/slug-da-noticia, o parâmetro `$url` recebido contém o valor "/noticias/slug-da-noticia".
- Ao invocar o método `get_current_url_pieces` da classe pai **DatabaseObject**, a variável `$url_pieces` recebe um vetor com os pedaços da URL a partir da `BASE_PATH` definida na classe, ou seja, nesse caso recebe o valor `array(0 => 'slug-da-noticia')`.
- Assim, tendo em mãos o *slug* da notícia acessada, é possível verificar no banco de dados através de uma consulta SQL se realmente existe um registro de notícia na tabela definida por `TABLE_NAME`. Se existir, o método retorna um vetor de página no mesmo padrão definido pela classe **Router** e a classe **Control** inclui a página. Se não existir, o método retorna `false` e a classe **Control** exibe página não encontrada.

ANEXOS

Retirando mensagens de *notices* do PHP

- Adicionar "& ~E_NOTICE" na seguinte linha do **php.ini**

```
error_reporting = E_ALL & ~E_NOTICE
```

Ativando o módulo de *mod_rewrite* no Apache:

- Descomentar a seguinte linha no arquivo **httpd.conf**:

```
LoadModule rewrite_module modules/mod_rewrite.so
```

Inserindo de *Virtual Hosts* no Apache:

- Inserir o seguinte trecho no final do arquivo **httpd.conf**:

```
NameVirtualHost *
<VirtualHost *>
      ServerName localhost
      DocumentRoot "C:\wamp\www"
</VirtualHost>

<Directory "C:/wamp/www">
      Options FollowSymLinks
      AllowOverride All
      Order allow,deny
      Allow from all
      Options +Indexes
</Directory>
```

- Inserir e configurar, para cada site, o seguinte trecho no final do arquivo **httpd.conf**:

```
<VirtualHost *>
      ServerName site.loc
      DocumentRoot "C:\wamp\www\site"
</VirtualHost>
```

- Inserir e configurar, para cada site, a seguinte linha no final do arquivo **hosts** do Windows (*C:\Windows\System32\drivers\etc\hosts*):

```
127.0.0.1   site  site.loc    site
```