**ARQUIVOS BASE E SISTEMA DE ROTAS**

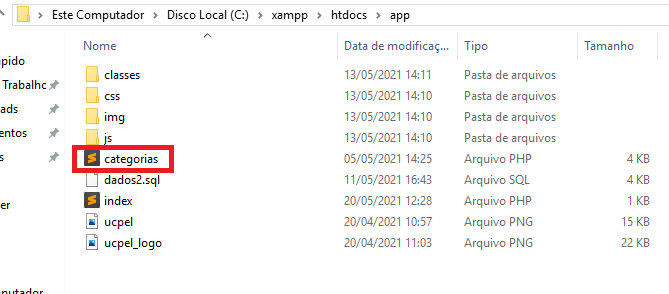
Pessoal, como eu havia comentado, eu tinha feito um projeto que tinha sistemas de rotas há um tempo atrás. Eu peguei aquele projeto e limpei tudo deixando só o essencial para um projeto novo. Acredito que com esses arquivos dá para fazer um projeto bem organizado e elegante de se utilizar. Vou tentar deixar tudo explicado para vocês.

**SISTEMA DE ROTAS**

Antes de seguir para a explicação dos arquivos, vou falar sobre o que é um sistema de rotas. Para começar, se executarmos os arquivos da aula do professor, vamos notar o seguinte no link do navegador:



Como pode ser observado, o que aparece ali é o nome do arquivo que está na pasta.



Se entramos em um site qualquer, como o site da Católica, por exemplo, vamos ver o seguinte:



Pode-se notar que o que aparece no link é uma informação que não necessariamente tem alguma relação com algum arquivo. Basicamente o que está ali é o que eles quiseram colocar.

Isso é o que o sistema de rotas faz, ele nos permite personalizar o link como quisermos, não dependendo mais do nome do arquivo.

Há muitas vantagens em fazer isso. Vou citar aqui algumas que me lembro:

\*Os links ficam elegantes;

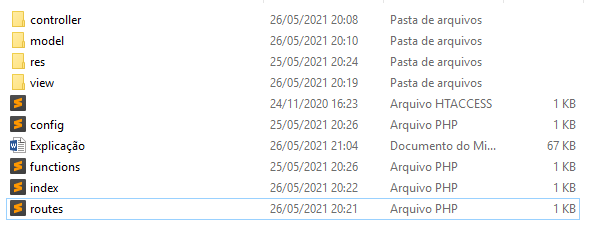
\*A estrutura de pasta interna não influencia na forma do link.

\*Melhor controle nos acessos e nas páginas.

Do sistema de rotas é basicamente isso.

**EXPLICAÇÃO DOS ARQUIVOS**

**VISÃO GERAL**



A estrutura que estou mandando para vocês é essa acima.

A pasta “controller” vai ser para os arquivos que vão realizar a lógica do sistema.

A pasta “model” será para as classes que tem de manipular o banco de dados (semelhante às classes Categorias, Crud e DB dos arquivos do professor).

A pasta “res” será para recursos gerais, como arquivos css, javascript, imagens e outras coisas semelhantes.

A pasta “view” será para os arquivos de visualização (html) (bem semelhante ao arquivo “categorias.php” que o professor abre para carregar a página).

O arquivo sem nome do tipo “HTACCESS” vou explicar mais à frente.

O arquivo “config.php” será para configurações gerais.

O arquivo “functions.php” será para funções gerais que resolvermos utilizar ao redor do projeto.

O arquivo “index.php” é o arquivo que é carregado ao acessar a página. Vou explicar mais para frente também.

O arquivo “Routes.php” é onde estarão registradas as rotas (links) que queremos utilizar.

**COMO UTILIZAR O SISTEMA**

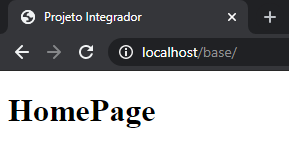
****

A base do sistema é o arquivo routes.php. É aqui que iremos registrar os nomes dos links que quisermos utilizar.

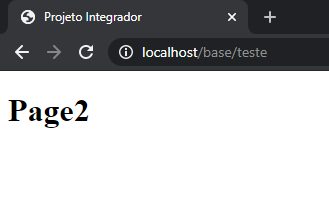
Não se assustem com todas essas classes sendo utilizadas, vou explicar elas mais para frente. Por enquanto entendam o geral de como utilizar.

Basicamente o ‘index.php’ e o ‘teste’ são as rotas que estão registradas no arquivo nesse momento.

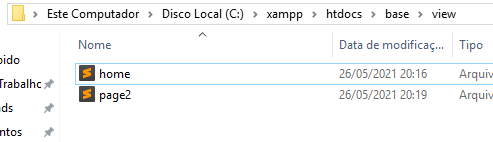
Ao entrarmos no projeto, é carregada a rota index.php. O index.php é equivalente a uma “/” no navegador. Então o link é o nome do projeto (“base”, por enquanto) + “/”. E isso carrega a view “home”. Conforme pode ser observado abaixo:



Ao entramos na rota “base/teste”, isto é, a pasta do projeto + a rota “teste”, chamaremos a view “page2”. (Entenda-se view como um arquivo html). Apresentando o seguinte resultado:



Esses arquivos que estão aparecendo estão na pasta view:







Então, resumindo, para criar uma rota é só adicionar o seguinte no arquivo routes:



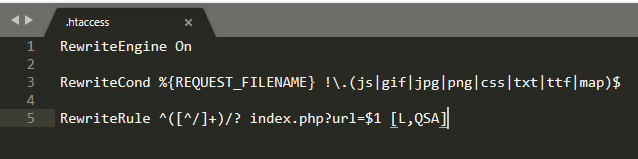
O básico do funcionamento é isto. Entendam bem essa parte antes de irem para as próximas

**ENTENDO O QUE OCORRE ATRÁS DAS CORTINAS**

A partir daqui vou explicar como o sistema faz isso. Para utilizar ele só é necessário saber a parte de cima. Aqui é mais para entender o que ele faz por trás dos panos.

O que torna isso possível são 4 arquivos que interagem entre si: o arquivo .htaccess, o index.php, a classe Route.php e a classe Controller.php (ambas presentes na pasta controller).

**O ARQUIVO .HTACCESS**

****

Isso aqui já não é PHP, mas acredito que algo relacionado com o servidor Apache. Não entendo muito dessa parte, mas o que ocorre é o seguinte, esse arquivo diz para o servidor redirecionar qualquer coisa que colocarmos na barra de navegação para o arquivo index.php. Então é como se independente do que escrevermos sempre estaremos abrindo o arquivo index.php.

Ele tem algumas exceções ali, como arquivos js, gif, jpg, png, css, txt, ttf e map. Estas exceções servem para que ele não redirecione quando formos abrir esses tipos de arquivos. Isso acontece porque caso fossemos tentar abrir uma imagem ou script seriamos redirecionados para o index.php. não abrindo os arquivos. Essas exceções servem, portanto, para podermos ter acesso a esses tipos de arquivos.

Enfim, daqui deve-se entender que isso faz com que o arquivo index.php seja sempre carregado, independente do link (com excessão dos arquivos com aquelas extensões).

**O ARQUIVO INDEX.PHP**

****

Esse arquivo basicamente faz o seguinte:

\*Ele vai varrer a pasta model, dando require\_once em todos os arquivos que estão lá dentro;

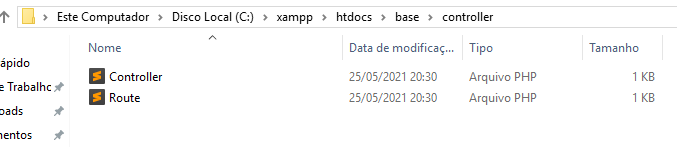
\*Vai varrer, também, a pasta controller, dando require\_once em todos os arquivos que estão lá dentro;

\*Vai requerer o config.php;

\*Vai requerer o arquivo functions.php;

\*Vai requerer o arquivo routes.php.

O importante aqui é entender que ele vai requerer estes 2 arquivos que estão na pasta controller e que também chama o routes.php. Isso é importante porque essas classes são as que são utilizadas pelo routes.php.



Isso me leva, por sua vez, a explicar essas classes:

**A CLASSE ROUTE.PHP**

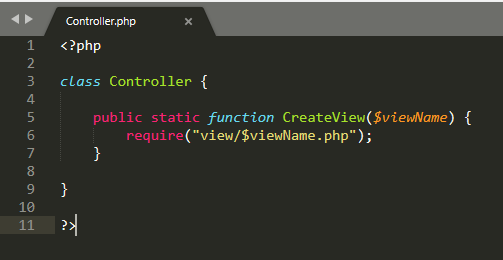
****

Essa classe tem um array (vetor) estático onde ficarão salvas as rotas válidas. Também tem o método estático set (aquele que utilizamos para registrar as rotas).

Quando realizamos o método set lá no arquivo routes.php basicamente estamos registrando uma rota nesse array e verificando o link em que estamos no navegador. Isso ocorre toda vez que acessamos uma página. Se o link em que estamos é igual a rota que registramos, é invocada a função que passamos como parâmetro.

Ela não é uma classe longa, mas tem vários detalhezinhos, recomendo tentarem entender bem ela e pesquisar tudo o que foi utilizado aqui.

**A CLASSE CONTROLLER.PHP**

****

Essa provavelmente é a classe mais simples de entender no momento. Ela contém apenas o método estático CreateView. A única coisa que esse método faz é dar require na view que passamos no parâmetro $viewname.

**CONCLUSÃO**

Pessoal, tentei dar o meu melhor nessa explicação, sendo o mais didático que eu consegui. Ainda assim, qualquer dúvida é só me perguntar. Recomendo analisarem bem tudo o que está aqui. Se não conseguirem entender tudo, tentem pelo menos entender como utilizar. Eu pessoalmente acredito que será de grande utilidade usar esses arquivos como base para o projeto, principalmente pela organização e clareza que é possível ter conforme as coisas vão crescendo.

Bons estudos!

Juliano Duarte Seus