Normas de Codificação PHP & Laravel

Detalhamento de normas e padrões para o desenvolvimento de aplicações PHP e Laravel.

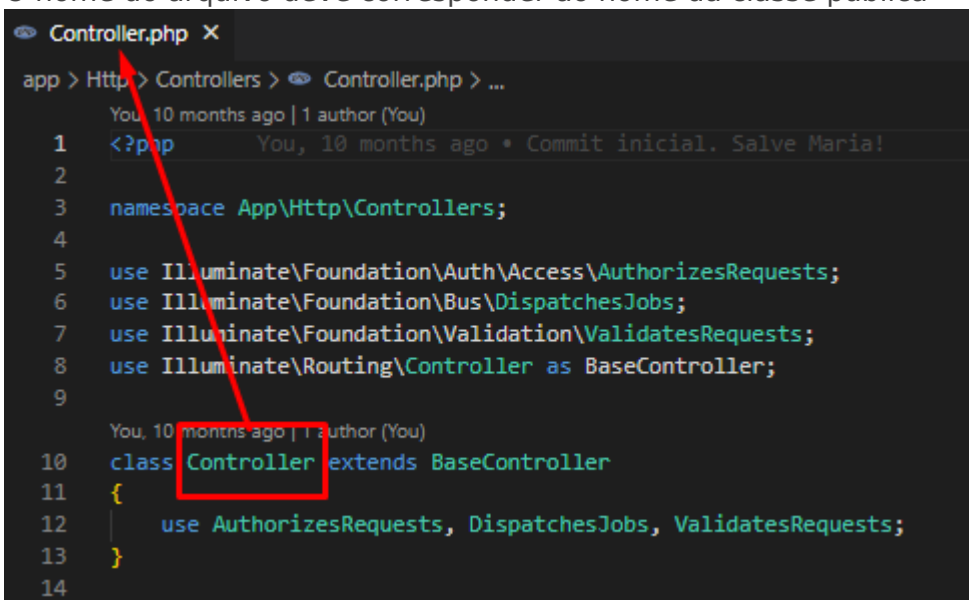
Arquivos

Nomenclatura

- **Sobre o Nome de Arquivos**
  Os arquivos devem ser nomeados com palavras iniciando com letra maiúscula. Se o nome for composto por várias palavras, cada palavra também inicia com letra maiúscula.
  Note exemplo abaixo:

```
class NomeDaClasse {
...
}
```

- **Arquivos de Classe**
  O nome do arquivo deve corresponder ao nome da classe publica



Note a imagem acima que o nome da classe é **Controller** e o nome do arquivo é **Controller.php**

Não defina mais de uma classe dentro do mesmo arquivo .php!

Arquivos que não contém classe devem ser nomeados de forma que representem a estrutura contida!

- **Arquivos de Views**
  - Arquivos de formulários devem iniciar com o prefixo "**frm_**" seguido do nome do formulário com letra minuscula. Se o nome for composto por várias palavras, cada

palavra também deve ser antecedida com *underscore* _.
    - Exemplos:
      frm_create_user.blade.php
      frm_login.blade.php
      frm_edit_role.blade.php
- Arquivos de listagem de registros devem iniciar com o prefixo "**lst_**" seguido do nome da lista com letra minuscula. Se o nome for composto por várias palavras, cada palavra também deve ser antecedida com *underscore* _.
    - Exemplos:
      lst_user.blade.php
      lst_material.blade.php
      lst_item_user.blade.php
- Arquivos de *index* devem ser nomeados como ***index.blade.php***

Diretórios

Nomenclatura

A criação de diretórios definem a organização e estrutura da aplicação e devem ser criados de forma que representem da melhor forma possível os conceitos que o agrupam.

O nome dos diretórios deve ser em letra **minuscula** e para casos de nomes compostos deve-se utilizar o ***underline*** para separação das palavras.

Por exemplo, poderiam ser definidos os diretórios "**gestao/financeiro/contas_receber**" e "**gestao/contabilidade**" para agrupar, respectivamente, as classes do subsistema de contas a receber do sistema financeiro e as classes da contabilidade.

Classes

Model

Models devem ficar de dentro do diretório **../app/Models/..**

O nome de uma *Model* deve ser igual à entidade que representa no banco de dados.

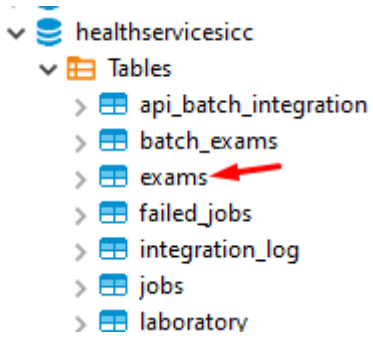
Por exemplo:

```
namespace App\Models\Laboratorio;


use Illuminate\Database\Eloquent\Model;


class Exams extends Model
{
...
}
```

No banco de dados devemos ter uma tabela com o mesmo nome.



> *Models* herdadas de bibliotecas externas devem ficar no diretório padrão de instalação da biblioteca, normalmente em **../app/**

Classes

Migration

O nome de uma *migration* deve seguir as seguintes regras:

Regra	**Nome Esperado**	**Comando artisan**
Criação de Tabelas	**CreateTable**NomeTabelaBanco...	`php artisan make:migration create_table_nomeTabela`
Inserir Registro	**Insert**NomeTabelaBanco...	`php artisan make:migration insert_nomeTabela_....`
Alteração Tabela	**Change**NomeTabelaBanco...	`php artisan make:migration change_nomeTabela_....`

Exemplo de *migration* para criação de tabela:

```
class CreateTableMaterial extends Migration
{
...
}
```

Exemplo de *migration* para inserir registros:

```
class InsertMaterial extends Migration
{
...
}
```

Exemplo de *migration* para alteração de tabela:

```
class ChangeMaterialAddColumnExternalCode extends Migration
{
...
}
```

> Todos os métodos Down devem ficar vázios.

Controller

Controllers deve ficar no diretório **App\Http\Controllers**

É importante organizar a localização das *controllers* obedecendo às Normas de criação de Diretórios no diretório supracitado.

O nome de uma *Controller* deve terminar com a palavra Controller. Note exemplo abaixo.

```
class MaterialExamController extends Controller
{
  ...
}
```

Request

Requests deve ficar no diretório **App\Http\Requests**

É importante organizar a localização das *Requests* obedecendo às Normas de criação de Diretórios no diretório supracitado.

O nome de uma Request deve iniciar com "Request"

Exemplo de classe Request

```
class RequestExamCreate extends FormRequest
{
  ...
}
```

Job

Jobs deve ficar no diretório **App\Jobs**

É importante organizar a localização das *Jobs* obedecendo às Normas de criação de Diretórios no diretório supracitado.

Um classe Job deve iniciar com "Job"

Exemplo:

```
class JobIntegrateBatchTasy implements ShouldQueue
{
  ...
}
```

Helper

Helpers deve ficar no diretório **App\Helpers**

É importante organizar a localização das *Helpers* obedecendo às Normas de criação de Diretórios no diretório supracitado.

Uma classe Helper deve inicar com "Helper"

```
class Helper
{
...
}
```

Rule

Rules deve ficar no diretório **App\Rules**

É importante organizar a localização das *Rules* obedecendo às Normas de criação de Diretórios no diretório supracitado.

Uma classe Rule deve iniciar com "Rule"

Exemplo

```
class RuleCpfPatient implements Rule
{
...
}
```

Rotas

Grupo de Rotas

Prioritariamente devemos utilizar o agrupamento de rotas conforme exemplo de código abaixo;

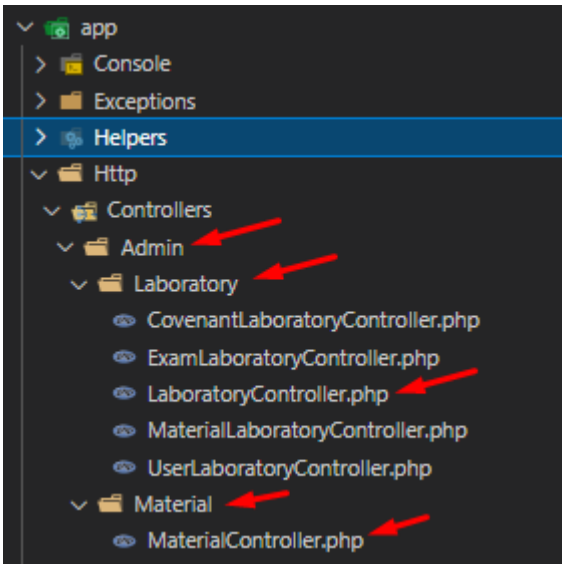
```
Route::group(['middleware' => 'auth'], function () {

  Route::prefix('admin')->name('admin.')->namespace('Admin\Laboratory')->group(function(){
    Route::resource('/laboratory','LaboratoryController');
  }

  Route::prefix('admin')->name('admin.')->namespace('Admin\Material')->group(function(){
        Route::resource('material','MaterialController');
    Route::post('/material/sync/{laboratory}','MaterialController@sync')->name('sync');
  });
}
```

Note na linha 9 uma rota para chamar o metódo `sync` na *controller* `MaterialCotroller`.

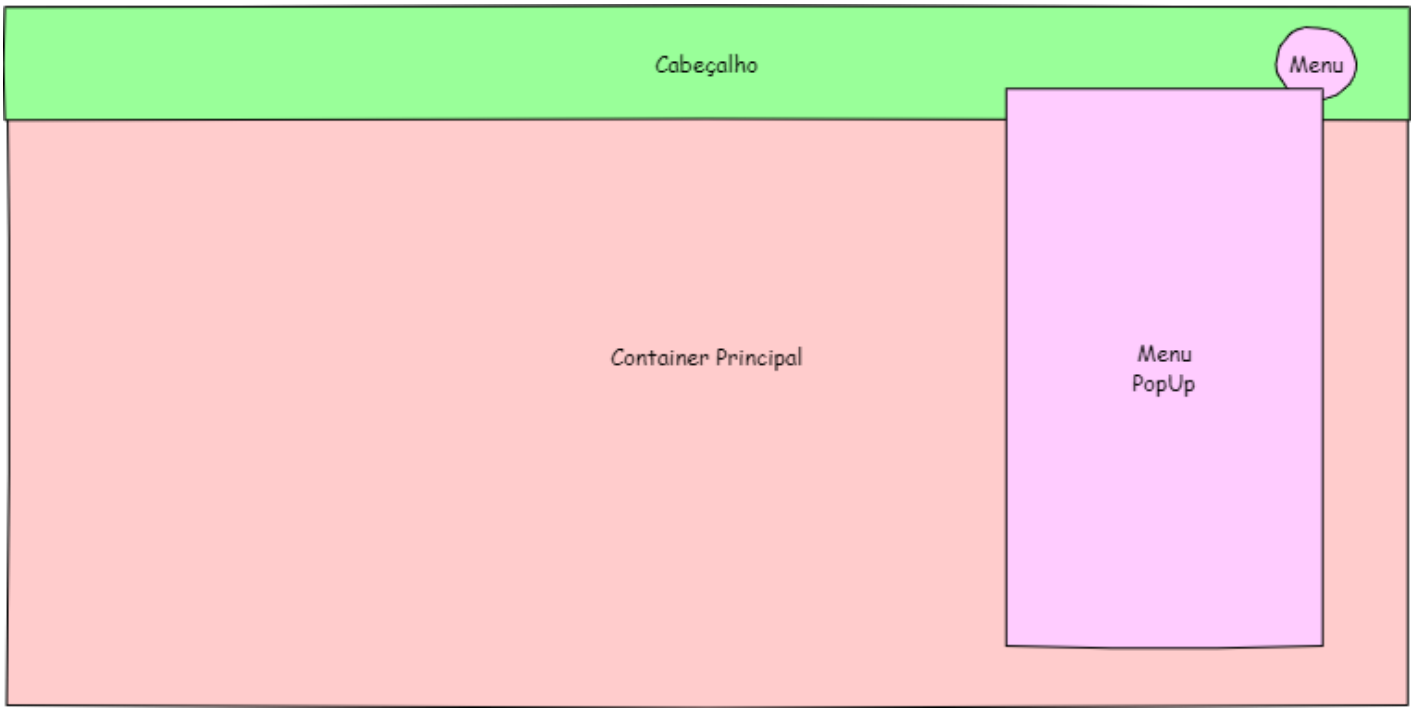
Note que nas linhas 3 e 7 do código acima é feita referência aos diretórios das *controllers* utilizadas. Observe na imagem abaixo como está organizada a estrutura do projeto.



Views

Estrutura - Principal

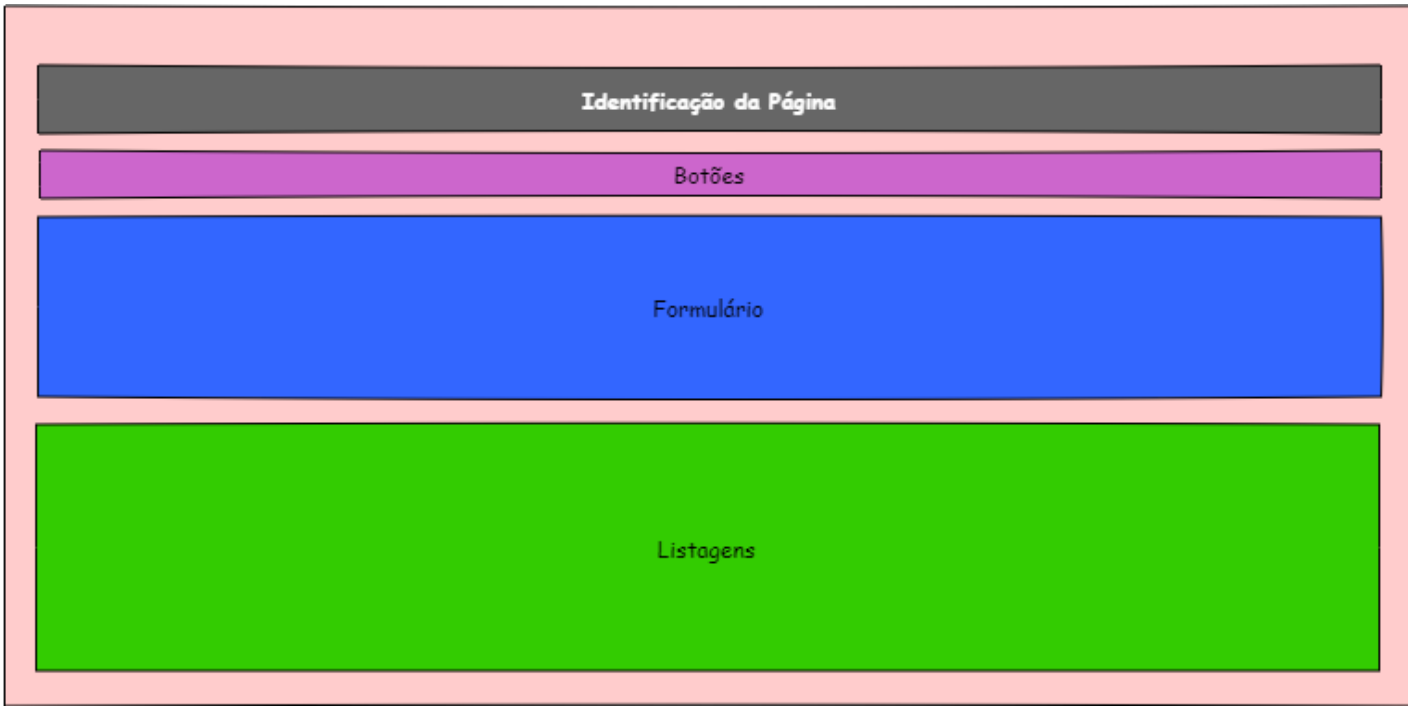
A estrutura principal das páginas index devem seguir o layout conforme imagem abaixo.



Estrutura - Container Principal

O Container Principal as páginas devem ser organizadas conforme imagem abaixo.

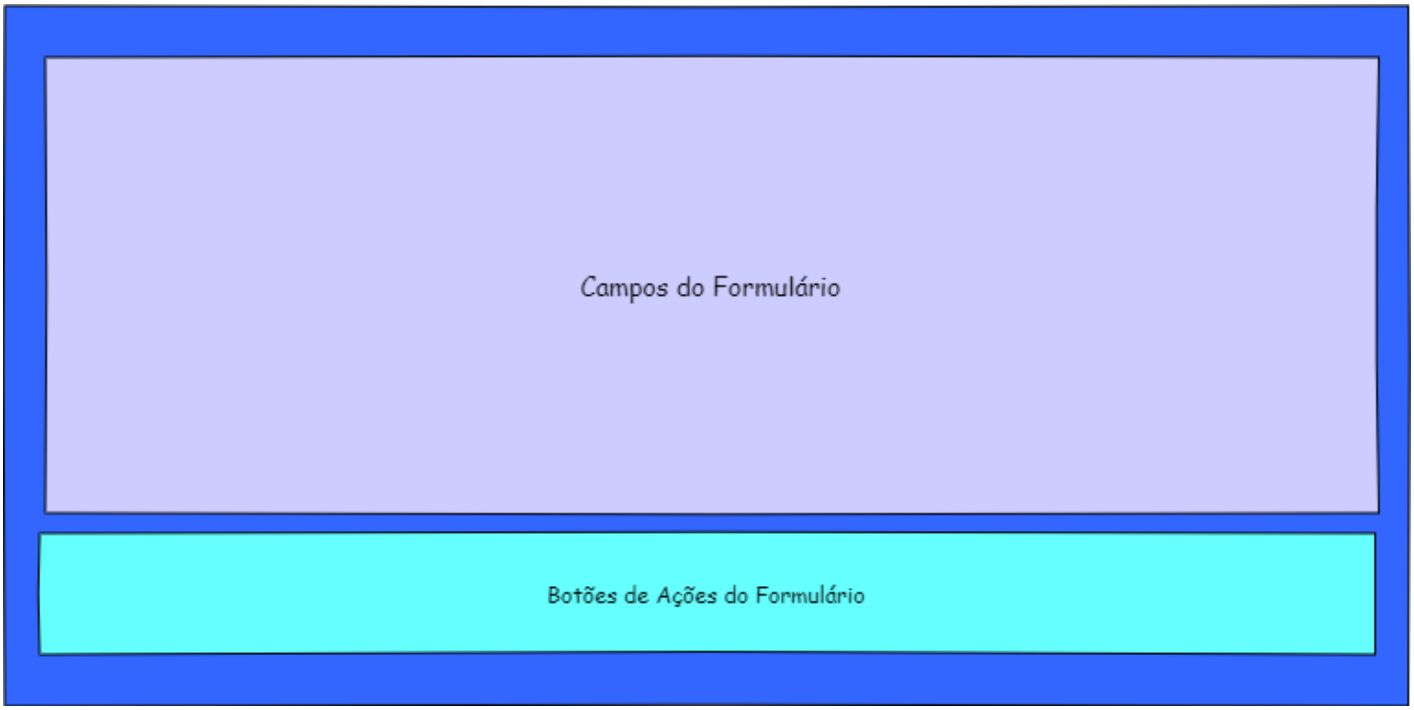
> Para navegação padrão do sistema um Container Principal deve estar **sempre com Cabeçalho e Menu**, conforme Estrutura - Principal



> Caso seja necessário abrir novas janelas/abas (`target="_blank"`) no fluxo da funcionalidade do sistema, então o Container Principal **NÃO deve estar** junto de Cabeçalho e Menu.
>
> Na página Pop Up - Abrindo nova Janela (aba) temos sugestão de código para esse tipo de situação.

Estrutura - Formulários

A estrutura para formulários deve seguir conforme a imagem abaixo.



Estrutura - Listagem

A estrutura para listagens (tabelas) de registros do sistema deve seguir a conforme imagem abaixo.

> Se for necessário a aplicação de formulários de filtros na lista deve-se seguir a Estrutura - Formulários

> A **Coluna de Opções** deve ser utilizada **apenas** para disponibilizar as ações do sistema referentes ao respctivo item da listagem.

Pop Up - Abrindo nova Janela (aba)

Note abaixo sugestão de código para abrir novas janela/aba da aplicação.

HTML

```
<a href="#" onclick="openwindow(); return false;">
  Anrindo Nova Janela
</a>
```

JavaScript

```
<script type="text/javascript">
  function openwindow() {

window.open("{{route('sua/rota/aqui')}}","nomeDaJanela","menubar=no,resizable=no,width=550,heigh
  }
</script>
```

Não é recomendado usar esse tipo de recurso. O ideal é manter o fluxo de uso da aplicação dentro do mesmo ambiente, seguindo a Estrutura - Principal

Variáveis

Nomenclatura

A nomenclatura das variáveis de classes, de instâncias atende aos requisitos abaixo:

A primeira letra das palavras, exceto da primeira palavra é maiúscula.

Exemplo:

```
$umaVariavelDeclarada;
```

A nomenclatura das variáveis locais e parâmetros atende aos requisitos abaixo:

A primeira letra das palavras é maiúscula.
A variável deve conter um prefixo que define seu tipo, conforme as tabelas abaixo:
Tipos primitivos:

Tipo de Dados	**Prefixo**
boolean	bl
int	it
string	st
datetime	dt
float	ft

Exemplo:

```
$stNome = 'Fulano da Silva' // string
$itIdade = 28; // int
$flPeso = 78.500; //float
$blCasado = true; //bool
```

Tipos Complexos:

Tipo de Dados	**Prefixo**
Array	lst
Objetos	obj

Exemplo:

```
$lstCursos = [1, 2, 3, 4]; // array

$objStdClasse = new stdClass;
$objStdClasse->propriedade = 'valor'; //objeto

$objPessoa = new Pessoa();
$objPessoa->stNome = "Fulano";
```

Banco de Dados

Tabela

As Tabelas da Aplicação devem seguir as seguintes definições:

Estrutura	tbXxxxxXxxxxxx
Formatação	
tb	Caractere utilizado para identificar que é um objeto do tipo Tabela da aplicação
XxxxxXxxxxxx	Descrição ou mnemônico da tabela com no máximo 30 posições onde a primeira letra de cada palavra é maiúscula e o restante minúsculo.
Exemplo	tbUserPermissions

> Tabelas herdadas de bibliotecas importadas no projeto devem permanecer com o nome original.

Colunas

As colunas das tabelas da aplicação devem seguir as seguintes definições:

Estrutura	cccXxxxxXxxxxxx
Formatação	
ccc	Caractere utilizado para identificar a classe conforme tabela de classe descrita abaixo
XxxxxXxxxxxx	Descrição ou mnemônico da coluna com no máximo 30 posições onde a primeira letra de cada palavra é maiúscula e o restante minúsculo.
Exemplo	codTypeExam

Tabelas herdadas de bibliotecas importadas no projeto devem permanecer como original.

Tabela de Classes

Classe	Descrição	Exemplo
id	Campo que identifica a chave única da tabela.	id
id **(índice)**	Utiliza-se o nome ou mnemônico da tabela de referência após a definição da classe **id**.	idUser idExam idPatient
cod	Representa um dado codificado que tem seu significado em uma tabela de domínio do sistema, ou um código de controle.	codBank codTypePerson
num	Identifica entidades específicas que não representam quantidades ou valores.	numCpf numCep
dsc	Identifica um campo que contém uma descrição de um código ou um elemento.	dscItem
nom	Campo alfanumérico que identifica entidades específicas.	nomClient nomEmployee

dt	Data ou partes de uma data do calendário.	dtBirth
dth	Data e Hora Exceto para Atributos **created_at** e **updated_at** (padrão Laravel).	dthShipping
hr	Hora	hrShipping
flg	Campo identificador	flgVip flgUrgent
vlr	Quantidade relativa a dinheiro ou medidas comumente usadas como valor.	vlrWage vlrRate
qtd	Número quantitativo de qualquer coisa excluindo valor monetário.	qtdItem qtdMax
lob	Campos para grande volume de informação, como por exemplo, imagem, arquivos.	lobPhoto lobJsonData

Índice

Estrutura	**id**XxxxxXxxxxxx
Formatação	
id	Caractere utilizado para identificar que é um objeto do tipo Índice.
XxxxxXxxxxxx	Descrição ou mnemônico do índice que deve obrigatoriamente expressar o mesmo nome da coluna a qual ele pertence com no máximo 20 posições onde a primeira letra de cada palavra é maiúscula e o restante minúsculo.
Exemplo	idUser idPatient