



# 1º Trabalho avaliação psi

## Trabalho

Trabalho realizado por:

Gonçalo gomes Nº7

### Instalação do CakePHP

Para instalar o cakephp no seu computador é usado o método composer. Apartir da cmd digita esta link na cmd que ira instalar

```
curl -s https://getcomposer.org/installer | php
```

Em seguida, basta digitar a seguinte linha de comando no seu terminal a partir do diretório onde se localiza o arquivo `composer.phar` para instalar o esqueleto da aplicação do CakePHP no diretório `[nome_do_app]`.

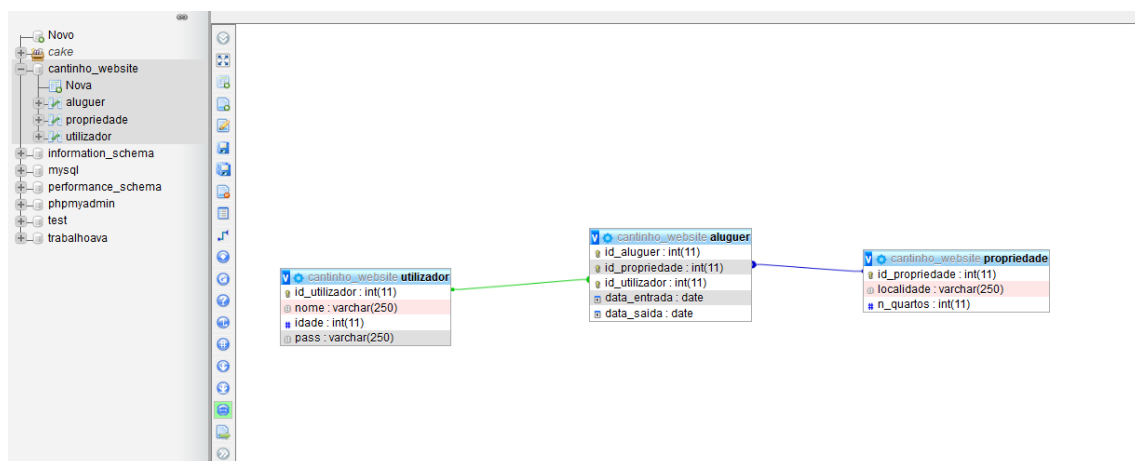
```
php composer.phar create-project --prefer-dist cakephp/app:^3.8  
[nome_do_app]
```

Quando a instalação for concluída devera ter a mesma estrutura de pastas como a que aparece em baixo

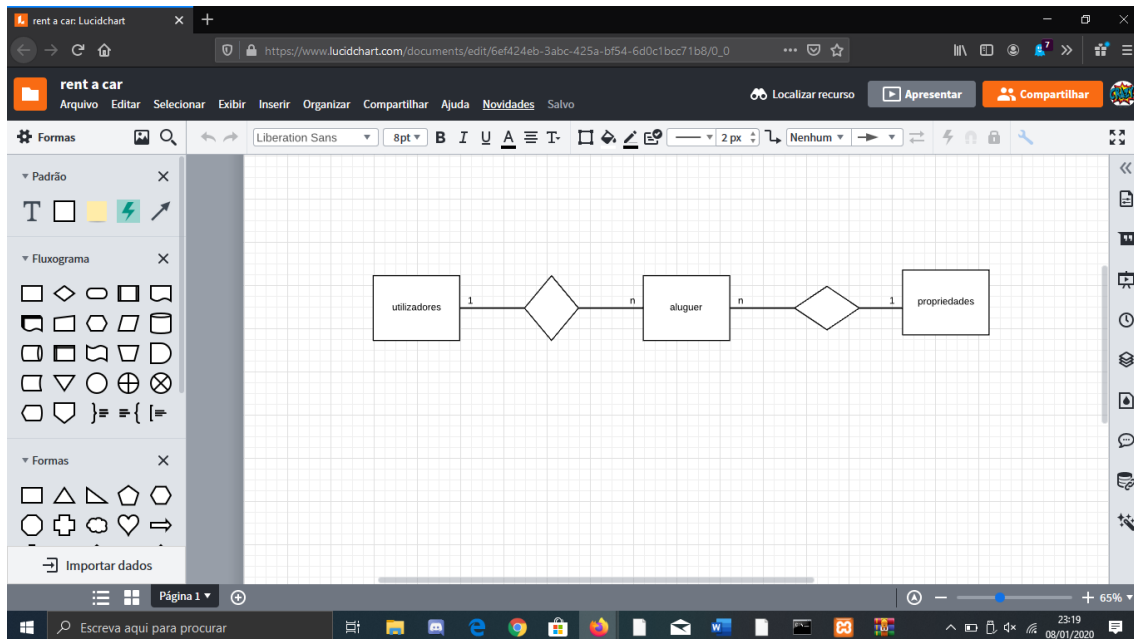
Nome	Data de modificação	Tipo	Tamanho
.github	09/01/2020 14:44	Pasta de ficheiros	
bin	09/01/2020 14:44	Pasta de ficheiros	
config	09/01/2020 14:46	Pasta de ficheiros	
logs	09/01/2020 15:13	Pasta de ficheiros	
plugins	09/01/2020 14:44	Pasta de ficheiros	
src	09/01/2020 14:44	Pasta de ficheiros	
tests	09/01/2020 14:44	Pasta de ficheiros	
tmp	10/01/2020 00:01	Pasta de ficheiros	
vendor	09/01/2020 14:46	Pasta de ficheiros	
webroot	09/01/2020 14:44	Pasta de ficheiros	
.editorconfig	09/01/2020 14:44	Ficheiro EDITORC...	1 KB
.gitattributes	09/01/2020 14:44	Ficheiro GITATTRIB...	1 KB
.gitignore	09/01/2020 14:44	Ficheiro GITIGNORE	1 KB
.htaccess	09/01/2020 14:44	Ficheiro HTACCESS	1 KB
.travis.yml	09/01/2020 14:44	Ficheiro YML	1 KB
composer.json	09/01/2020 14:44	Ficheiro JSON	2 KB
composer.lock	09/01/2020 14:46	Ficheiro LOCK	151 KB
index	09/01/2020 14:44	Ficheiro PHP	1 KB
phpunit.xml.dist	09/01/2020 14:44	Ficheiro DIST	2 KB
README.md	09/01/2020 14:44	Ficheiro MD	2 KB

Criando a base de dados:

Create DataBase cantinho\_website;



# DER:



## Criando ligação há base de dados com cakephp:

Em seguida, vamos dizer ao CakePHP onde se encontra a base de dados e como se ligar a ela.

A configuração é simples e objetiva, basta alterar os valores no array `Datasources.default` localizado no arquivo **config/app.php**,

```
return [
    'Datasources' => [
        'default' => [
            'className' => 'Cake\Database\Connection',
            'driver' => 'Cake\Database\Driver\Mysql',
            'persistent' => false,
            'host' => 'localhost',
            'username' => 'cakephp',
            'password' => 'AngelF00dC4k3~',
            'database' => 'cake_blog',
            'encoding' => 'utf8',
            'timezone' => 'UTC',
            'cacheMetadata' => true,
        ],
    ],
];
```


```

* E.g set it to 'utf8mb4' in MariaDB and MySQL and 'utf8' for any
* other RBMS.
*/
'Datasources' => [
    'default' => [
        'className' => Connection::class,
        'driver' => Mysql::class,
        'persistent' => false,
        'host' => 'localhost',
        /*
        * CakePHP will use the default DB port based on the driver selected
        * MySQL on MAMP uses port 8889, MAMP users will want to uncomment
        * the following line and set the port accordingly
        */
        //'port' => 'non_standard_port_number',
        'username' => 'pentium',
        'password' => 'pentium',
        'database' => 'cantinho_website',
        /*
        * You do not need to set this flag to use full utf-8 encoding (internal default since CakePHP 3.6).
        */
        //'encoding' => 'utf8mb4',
        'timezone' => 'UTC',
        'flags' => [],
        'cacheMetadata' => true,
        'log' => false,
    ],
],
/**

```

Depois de guardar o documento **config/app.php**, se for á pagina do cake php onde diz base de dados, deverá ficar verde.

## Database

 CakePHP is able to connect to the database.

Caso a mesma se mantenha a vermelho deverá verificar se efectuou bem os passos e a conexão esta bem feita.

## Criando o model

Após criar um model no CakePHP, teremos a base necessária para interagirmos com a base de dados e executar operações.

O ficheiro que criaremos deverá ficar guardado em **src/Model/Table/ArticlesTable.php**:

```

// src/Model/Table/ArticlesTable.php

namespace App\Model\Table;

use Cake\ORM\Table;

class ArticlesTable extends Table
{
    public function initialize(array $config)
    {
        $this->addBehavior('Timestamp');
    }
}

```

## Criando o controller

A seguir, criaremos um controller para controlar nossos artigos. O controller é o lugar onde se utilizam as regras contidas nos models e se executam as tarefas relacionadas com os artigos. Criaremos um arquivo chamado **ArticlesController.php** no diretório **src/Controller**:

```
// src/Controller/ArticlesController.php

namespace App\Controller;

class ArticlesController extends ApplicationController
{
    public function index()
    {
        $articles = $this->Articles->find('all');
        $this->set(compact('articles'));
    }
}
```

```
<?php

namespace App\Controller;

class AluguerController extends ApplicationController
{
    public function index()
    {
        $aluguer = $this->Aluguer->find('all');
        $this->set(compact('aluguer'));
    }
    public function view($id = null)
    {
        $aluguer = $this->Aluguer->get($id);
        $this->set(compact('aluguer'));
    }
}
```

## Criando as views

Um layout é um conjunto de códigos encontrado ao redor das views. Múltiplos layouts podem ser definidos, e você pode alternar entre eles, mas agora, vamos usar o default, localizado em `src/Template/Layout/default.ctp`.

```
<!-- File: src/Template/Articles/index.ctp -->

<h1>Blog articles</h1>
<table>
    <tr>
        <th>Id</th>
        <th>Title</th>
        <th>Created</th>
    </tr>

    <!-- Aqui é onde iremos iterar nosso objeto de solicitação
    $articles, exibindo informações de artigos -->

    <?php foreach ($articles as $article): ?>
    <tr>
        <td><?= $article->id ?></td>
        <td>
            <?= $this->Html->link($article->title, ['action' =>
'view', $article->id]) ?>
        </td>
        <td>
            <?= $article->created->format (DATE_RFC850) ?>
        </td>
    </tr>
    <?php endforeach; ?>
</table>
```

```
1  <h1>Aluguer</h1>
2  <p><?= $this->Html->link("Adicionar Aluguer", ['action' => 'add']) ?></p>
3  <table>
4  <tr>
5      <th>Id Contrato</th>
6      <th>Id Propriedade</th>
7      <th>Id Utilizador</th>
8      <th>Data de Entrada</th>
9      <th>Data de Saída</th>
10 </tr>
11 <?php foreach ($aluguer as $alugado): //Lista todos os utilizadores existentes na tabela?>
12 <tr>
13     <td><?= $alugado->id_aluguer ?></td>
14     <td>
15         <?= $alugado->id_propriedade ?>
16     </td>
17     <td>
18         <?= $alugado->id_utilizador ?>
19     </td>
20     <td>
21         <?= $alugado->data_entrada ?>
22     </td>
23     <td>
24         <?= $alugado->data_saida ?>
25     </td>
26     <td>
27         <?= $this->Html->link('Editar', ['action' => 'edit', $alugado->id_aluguer]) //Onde iremos editar o registo ?>
28     </td>
29     <td>
30         <?= $this->Form->postLink('Apagar', ['action' => 'delete', $alugado->id_aluguer], ['confirm' => 'Deseja apagar o utilizador selecio
31     </td>
32 </tr>
```

Com este código iremos criar as action que não foram criadas no código anterior:

```
// src/Controller/ArticlesController.php

namespace App\Controller;

class ArticlesController extends ApplicationController
{
    public function index()
    {
        $this->set('articles', $this->Articles->find('all'));
    }

    public function view($id = null)
    {
        $article = $this->Articles->get($id);
        $this->set(compact('article'));
    }
}
```

Ao usar a função `get()`, fazemos também algumas verificações para garantir que o usuário realmente está a ter acesso.

Agora vamos criar a view para nossa action em **src/Template/Articles/view.ctp**

```
<!-- File: src/Template/Articles/view.ctp -->

<h1><?= h($article->title) ?></h1>
<p><?= h($article->body) ?></p>
<p><small>Criado: <?= $article->created->format(DATE_RFC850)
?></small></p>
```

Verifique se está tudo a funcionar acedendo às pastas em `/articles/index` ou manualmente. Solicite a visualização de um artigo acedendo a `articles/view/{id}`.

## Adicionando artigos

Primeiro, comece por criar a action `add()` no `ArticlesController`:

```
// src/Controller/ArticlesController.php

namespace App\Controller;

use App\Controller\AppController;

class ArticlesController extends ApplicationController
{
    public function initialize()
    {
        parent::initialize();
    }
}
```

```

        $this->loadComponent('Flash'); // Inclui o FlashComponent
    }

    public function index()
    {
        $this->set('articles', $this->Articles->find('all'));
    }

    public function view($id)
    {
        $article = $this->Articles->get($id);
        $this->set(compact('article'));
    }

    public function add()
    {
        $article = $this->Articles->newEntity();
        if ($this->request->is('post')) {
            $article = $this->Articles->patchEntity($article, $this->request->getData());
            if ($this->Articles->save($article)) {
                $this->Flash->success(__('Seu artigo foi salvo.'));
                return $this->redirect(['action' => 'index']);
            }
            $this->Flash->error(__('Não é possível adicionar o seu artigo.'));
        }
        $this->set('article', $article);
    }
}

```

```

C:\> xampp > htdocs > trabalhoM13 > src > Controller > AluguerController.php
1  <?php
2  namespace App\Controller;
3  class AluguerController extends ApplicationController
4  {
5      public function index()
6      {
7          $aluguer = $this->Aluguer->find('all');
8          $this->set(compact('aluguer'));
9      }
10     public function view($id = null)
11     {
12         $aluguer = $this->Aluguer->get($id);
13         $this->set(compact('aluguer'));
14     }
15
16     public function add()
17     {
18         $aluguer = $this->Aluguer->newEmptyEntity();
19         if ($this->request->is('post')) {
20             $aluguer = $this->Aluguer->patchEntity($aluguer, $this->request->getData());
21
22             if ($this->Aluguer->save($aluguer)) {
23                 $this->Flash->success(__('Reserva feita com sucesso.'));
24                 return $this->redirect(['action' => 'index']);
25             }
26         }
27     }
28     public function delete($id)
29     {
30         $this->request->allowMethod(['post', 'delete']);
31
32         $aluguer = $this->Aluguer->get($id);
33         if ($this->Aluguer->delete($aluguer)) {

```



```

        if ($this->Aluguer->delete($aluguer)) {
            $this->Flash->success(__('O propriedades com id: {0} foi apagado.', h($id)));
            return $this->redirect(['action' => 'index']);
        }
    }
    public function edit($id = null)
    {
        $aluguer = $this->Aluguer->get($id);
        if ($this->request->is(['post', 'put'])) {
            $this->Aluguer->patchEntity($aluguer, $this->request->getData());
            if ($this->Aluguer->save($aluguer)) {
                $this->Flash->success(__('A propriedades foi atualizado.'));
                return $this->redirect(['action' => 'index']);
            }
            $this->Flash->error(__('A propriedades não pôde ser atualizado.'));
        }

        $this->set('aluguer', $aluguer);
    }
}

```

## Validação de artigos

O CakePHP torna mais prática e menos monótona a validação de dados de formulário.

Para usufruir dos recursos de validação, precisamos usar o [Form](#) helper nas views. O [Cake\View\Helper\FormHelper](#) está disponível por padrão em todas as views através do uso do `$this->Form`.

Segue a view correspondente a action add:

```

<!-- File: src/Template/Articles/add.ctp -->

<h1>Add Article</h1>
<?php
    echo $this->Form->create($article);
    echo $this->Form->input('title');
    echo $this->Form->input('body', ['rows' => '3']);
    echo $this->Form->button(__('Salvar artigo'));
    echo $this->Form->end();
?>

```

```

C: > xampp > htdocs > trabalhoM13 > src > Template > Aluguer > add.ctp
1  |<!-- File: src/Template/Articles/add.ctp -->
2
3  <h1>Adicionar aluguer</h1>
4  <?php
5      echo $this->Form->create($aluguer);
6      echo $this->Form->input('title');
7      echo $this->Form->input('body', ['rows' => '3']);
8      echo $this->Form->button(__('Salvar artigo'));
9      echo $this->Form->end();
10 ?>

```

```

// src/Model/Table/ArticlesTable.php

namespace App\Model\Table;

use Cake\ORM\Table;
use Cake\Validation\Validator;

class ArticlesTable extends Table
{
    public function initialize(array $config)
    {
        $this->addBehavior('Timestamp');
    }

    public function validationDefault(Validator $validator)
    {
        $validator
            ->notEmpty('title')
            ->notEmpty('body');

        return $validator;
    }
}

```

## Edição de artigos

Cria-se a action e de seguida a view. A action `edit()` deverá ser inserida no `ArticlesController`:

```

// src/Controller/ArticlesController.php

public function edit($id = null)
{
    $article = $this->Articles->get($id);
    if ($this->request->is(['post', 'put'])) {

```

```

        $this->Articles->patchEntity($article, $this->request-
>getData());
        if ($this->Articles->save($article)) {
            $this->Flash->success(__('Seu artigo foi atualizado.'));
            return $this->redirect(['action' => 'index']);
        }
        $this->Flash->error(__('Seu artigo não pôde ser atualizado.'));
    }

    $this->set('article', $article);
}

```

Essa “action” em primeiro lugar, certifica-se que o registo existe. Se o parâmetro `$id` não foi passado ou se o registo é inexistente, uma *NotFoundException* é emitida pelo *ErrorHandler* do CakePHP.

Em seguida, a action verifica se a requisição é POST ou PUT e caso seja, os dados são usados para atualizar a entidade de artigo em questão ao usar o método `patchEntity()`. Então usamos o `ArticlesTable` para guardar a entidade.

Segue a view correspondente a action edit:

```

<!-- File: src/Template/Articles/edit.ctp -->

<h1>Edit Article</h1>
<?php
    echo $this->Form->create($article);
    echo $this->Form->input('title');
    echo $this->Form->input('body', ['rows' => '3']);
    echo $this->Form->button(__('Salvar artigo'));
    echo $this->Form->end();
?>

```

O CakePHP irá determinar se o `save()` vai inserir ou atualizar um registo baseado nos dados da entidade.

Pode-se atualizar a sua view index com os links para editar artigos:

```

<!-- File: src/Template/Articles/index.ctp (edit links added) -->

<h1>Blog articles</h1>
<p><?= $this->Html->link("Adicionar artigo", ['action' => 'add'])
?></p>
<table>
    <tr>
        <th>Id</th>
        <th>Título</th>
        <th>Criado</th>
        <th>Ações</th>
    </tr>

```

```

<!-- Aqui é onde iremos iterar nosso objeto de solicitação $articles,
exibindo informações de artigos -->

<?php foreach ($articles as $article): ?>
    <tr>
        <td><?= $article->id ?></td>
        <td>
            <?= $this->Html->link($article->title, ['action' =>
'view', $article->id]) ?>
        </td>
        <td>
            <?= $article->created->format(DATE_RFC850) ?>
        </td>
        <td>
            <?= $this->Html->link('Editar', ['action' => 'edit',
$article->id]) ?>
        </td>
    </tr>
<?php endforeach; ?>

</table>

```

## Apagar artigos

A seguir, vamos criar uma forma de apagar artigos.

```

// src/Controller/ArticlesController.php

public function delete($id)
{
    $this->request->allowMethod(['post', 'delete']);

    $article = $this->Articles->get($id);
    if ($this->Articles->delete($article)) {
        $this->Flash->success(__('O artigo com id: {0} foi deletado.',
h($id)));
        return $this->redirect(['action' => 'index']);
    }
}

```

```

public function delete($id)
{
    $this->request->allowMethod(['post', 'delete']);

    $aluguer = $this->Aluguer->get($id);
    if ($this->Aluguer->delete($aluguer)) {
        $this->Flash->success(__('O propriedades com id: {0} foi apagado.', h($id)));
        return $this->redirect(['action' => 'index']);
    }
}

```

Com este código podemos apagar artigos:

```
<!-- File: src/Template/Articles/index.ctp (delete links added) -->

<h1>Blog articles</h1>
<p><?= $this->Html->link('Adicionar artigo', ['action' => 'add']) ?></p>
<table>
    <tr>
        <th>Id</th>
        <th>Titulo</th>
        <th>Criado</th>
        <th>Ações</th>
    </tr>

    <!-- Aqui é onde iremos iterar nosso objeto de solicitação $articles,
    exibindo informações de artigos -->

    <?php foreach ($articles as $article): ?>
    <tr>
        <td><?= $article->id ?></td>
        <td>
            <?= $this->Html->link($article->title, ['action' => 'view',
            $article->id]) ?>
        </td>
        <td>
            <?= $article->created->format(DATE_RFC850) ?>
        </td>
        <td>
            <?= $this->Form->postLink(
                'Deletar',
                ['action' => 'delete', $article->id],
                ['confirm' => 'Tem certeza?'])
            ?>
            <?= $this->Html->link('Edit', ['action' => 'edit', $article-
            >id]) ?>
        </td>
    </tr>
    <?php endforeach; ?> </table>
```

C:\> xampp > htdocs > trabalhoM13 > src > Template > Aluguer > index.ctp

```
1 <h1>Aluguer</h1>
2 <p><?= $this->Html->link("Adicionar Aluguer", ['action' => 'add']) ?></p>
3 <table>
4     <tr>
5         <th>Id Contrato</th>
6         <th>Id Propriedade</th>
7         <th>Id Utilizador</th>
8         <th>Data de Entrada</th>
9         <th>Data de Saída</th>
10    </tr>
11    <?php foreach ($aluguer as $alugado): //Lista todos os utilizadores existentes na tabela?>
12        <tr>
13            <td><?= $alugado->id_aluguer ?></td>
14            <td>
15                <?= $alugado->id_propriedade ?>
16            </td>
17            <td>
18                <?= $alugado->id_utilizador ?>
19            </td>
20            <td>
21                <?= $alugado->data_entrada ?>
22            </td>
23            <td>
24                <?= $alugado->data_saida ?>
25            </td>
26            <td>
27                <?= $this->Html->link('Editar', ['action' => 'edit', $alugado->id_aluguer]) //Onde iremos editar
28            </td>
29            <td>
30                <?= $this->Form->postLink('Apagar', ['action' => 'delete', $alugado->id_aluguer], ['confirm' => 'D
31            </td>
32        </tr>
33    <?php endforeach: ?>
```

## **WEBGRAFIA**

<https://book.cakephp.org/4/en/index.html>