

APLICAÇÃO DE AGENDAMENTO DE TAREFAS COM CAKEPHP

Guilherme Morigi 1

Ivan da Silva 2

Nathan Henrique Riffel 3

Thiago Emanuel Correia Heckler 4

RESUMO

Este artigo detalha o processo de desenvolvimento de uma aplicação web para agendamento e gerenciamento de tarefas, concebida como projeto para a disciplina de Programação 3. A plataforma foi construída utilizando o framework CakePHP 5, com PHP 8.1 e o sistema de gerenciamento de banco de dados PostgreSQL. O sistema implementa funcionalidades essenciais, incluindo um robusto sistema de autenticação de usuários, permitindo que cada indivíduo se cadastre, faça login e gerencie sua própria lista de tarefas de forma privada e segura. O projeto segue o padrão arquitetural MVC (Model-View-Controller), fazendo uso de ferramentas como Composer para gerenciamento de dependências e Migrations para o versionamento do banco de dados. O objetivo é apresentar uma solução funcional e didática para o problema de organização de atividades diárias, demonstrando a aplicação prática de conceitos de engenharia de software em um projeto real.

Palavras-chave: *CakePHP, MVC, Agendador de Tarefas, Desenvolvimento Web, PostgreSQL*

ABSTRACT

This article details the development process of a web application for scheduling and managing tasks, designed as a project for the Programming 3 course. The platform was built using the CakePHP 5 framework, with PHP 8.1 and the PostgreSQL database management system. The system implements essential functionalities, including a robust user authentication system, allowing individuals to register, log in, and privately and securely manage their own task lists. The project follows the MVC (Model-View-Controller) architectural pattern, utilizing tools such as Composer for dependency management and Migrations for database versioning. The goal is to present a functional and didactic solution for daily activity organization, demonstrating the practical application of software engineering concepts in a real-world project.

Keywords: CakePHP, MVC, Task Scheduler, Web Development, PostgreSQL.

1 INTRODUÇÃO

A gestão de tarefas diárias é um desafio comum na vida moderna, onde a sobrecarga de informações e responsabilidades pode levar à desorganização e à perda de produtividade. Ferramentas digitais de gerenciamento de tarefas surgem como soluções eficazes para mitigar esse problema, oferecendo um meio centralizado e acessível para organizar, priorizar e acompanhar atividades. No entanto, muitas soluções disponíveis podem ser excessivamente complexas ou onerosas para o usuário comum.

Diante desse cenário, este projeto aborda o desenvolvimento de uma aplicação web de agendamento de tarefas, denominada "Agendador de Tarefas". O sistema foi concebido com o objetivo de oferecer uma plataforma simples, intuitiva e segura para o gerenciamento de tarefas pessoais. O problema central que se busca resolver é a necessidade de uma ferramenta minimalista que permita a usuários cadastrados criar, visualizar, editar e remover suas próprias tarefas, garantindo a privacidade e a integridade de seus dados.

Desenvolvido como uma atividade acadêmica para a disciplina de Programação 3, o projeto não apenas visa criar um produto funcional, mas também aplicar e aprofundar os conhecimentos em tecnologias e conceitos fundamentais do desenvolvimento de software, como a arquitetura MVC, programação orientada a objetos, sistemas de autenticação e versionamento de código e banco de dados.

2 DESENVOLVIMENTO

O processo de construção do "Agendador de Tarefas" foi estruturado em etapas que abrangeram desde a fundamentação teórica até a implementação e os testes, utilizando uma combinação de tecnologias modernas e metodologias de desenvolvimento consolidadas.

Fundamentação Técnica

O projeto foi solidamente baseado em princípios e padrões de arquitetura de software para garantir manutenibilidade, escalabilidade e segurança.

Programação Orientada a Objetos (POO): O PHP, sendo uma linguagem com forte suporte à POO, foi a base do desenvolvimento. O framework CakePHP potencializa esse paradigma, onde cada componente da aplicação — como controladores, modelos (Tables e Entities) e views — é uma classe com responsabilidades bem definidas, promovendo a reutilização de código e a encapsulação.

Arquitetura MVC (Model-View-Controller): A aplicação foi estruturada seguindo o padrão MVC, uma diretriz central do CakePHP, que separa as responsabilidades da aplicação em três camadas:

- o **Model:** Responsável pela lógica de negócio e interação com o banco de dados. Implementado através das classes Table (ex: TasksTable.php, UsersTable.php), que gerenciam as consultas e a validação dos dados, e das classes Entity (ex: Task.php, User.php), que representam os registros individuais do banco de dados.

- o **View:** Camada de apresentação responsável por exibir os dados ao usuário. Localizada no diretório templates/, consiste em arquivos que renderizam a interface, como index.php para listagem e add.php para formulários de adição.

- o **Controller:** Atua como intermediário, recebendo as requisições do usuário, acionando as operações no Model e selecionando a View apropriada para a resposta. Os arquivos TasksController.php e UsersController.php contêm as ações (actions) que implementam as funcionalidades de CRUD de tarefas e autenticação de usuários, respectivamente.

Autenticação e Segurança: Para a gestão de acesso, foi utilizado o plugin oficial cakephp/authentication. Ele foi configurado em src/Application.php para gerenciar o login, logout e as sessões de usuário. A segurança das senhas é garantida pela função setPassword na entidade User.php, que utiliza o DefaultPasswordHasher para criptografar as senhas antes de salvá-las no banco. Além disso, o framework oferece proteção nativa contra ataques como Cross-Site Request

Forgery (CSRF), que foi mantida ativa.

- **Versionamento:** O controle de versão do código-fonte foi realizado com o Git, e o projeto foi hospedado no GitHub. O repositório contém um arquivo .gitignore configurado para excluir arquivos sensíveis e desnecessários, como config/app_local.php e o diretório vendor. Para o versionamento da estrutura do banco de dados, utilizou-se o recurso de Migrations do CakePHP.

Metodologia de Desenvolvimento

A equipe, composta por três membros, adotou uma abordagem de desenvolvimento colaborativa e iterativa. As tarefas foram divididas com base nas funcionalidades principais da aplicação (autenticação de usuário e CRUD de tarefas). O GitHub foi utilizado como plataforma central para colaboração, permitindo que os desenvolvedores trabalhassem em paralelo e integrassem suas contribuições de forma organizada. A comunicação constante e a revisão de código foram práticas adotadas para garantir a qualidade e a coesão do projeto final.

Descrição das Tecnologias e Frameworks Utilizados

- **Backend:** A aplicação foi desenvolvida com **PHP 8.1+** e o **CakePHP 5.x**, um framework de desenvolvimento rápido que oferece uma base robusta e ferramentas integradas para acelerar o processo construtivo.

- **Banco de Dados:** Foi escolhido o **PostgreSQL 16**, um sistema de gerenciamento de banco de dados relacional objeto, conhecido por sua robustez e conformidade com os padrões SQL. A configuração da conexão está definida nos arquivos config/app.php e config/app_local.php.

- **Frontend:** A interface do usuário foi construída com **HTML5** e estilizada com o framework **Milligram**, um framework CSS minimalista, complementado por estilos customizados definidos em webroot/css/custom.css para criar um tema escuro e uma identidade visual para a aplicação.

- **Dependências:** O **Composer** foi utilizado para gerenciar as dependências do projeto, incluindo o próprio CakePHP e seus plugins.

- **Servidor de Desenvolvimento:** Para os testes locais, foi utilizado o servidor web embutido no CakePHP, iniciado pelo comando bin/cake server.

Etapas do Processo Construtivo

· **Modelagem de Dados:** A estrutura do banco de dados foi definida e implementada utilizando o recurso de **Migrations** do CakePHP. Foram criados três arquivos de migração principais:

1. 20250617220000_CreateUsers.php: Cria a tabela users com os campos id, email, password e created/modified.
2. 20250610005848_CreateTasksTable.php: Cria a tabela tasks com os campos id, title, description, completed, data_agendada e created/modified.
3. 20250617220100_AddUserIdToTasks.php: Adiciona a coluna user_id à tabela tasks e cria uma chave estrangeira que a relaciona com a tabela users, estabelecendo o relacionamento de que um usuário pode ter várias tarefas.

· **Implementação:**

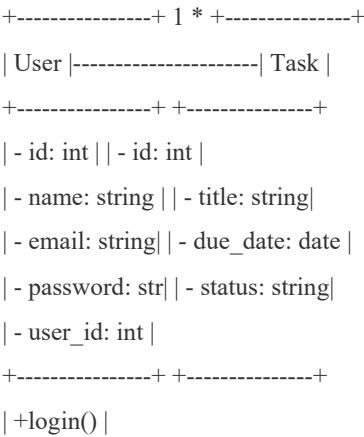
1. **Autenticação:** Foi criado o UsersController com as ações add, login e logout. A view login.php fornece o formulário de acesso e a add.php o de registro. O AppController foi configurado para carregar o componente de autenticação em toda a aplicação.
2. **CRUD de Tarefas:** O TasksController foi implementado com as ações index, view, add, edit e delete. As consultas foram ajustadas para garantir que um usuário só possa visualizar e manipular suas próprias tarefas, utilizando o id do usuário logado como filtro nas buscas.
3. **Views e Layout:** Foi desenvolvido um layout padrão (templates/layout/default.php) que inclui um menu de navegação e a área de conteúdo. O layout também verifica se um usuário está logado para exibir o link "Sair". Foram criados templates para cada ação dos controladores.

· **Testes:** A estrutura do CakePHP já fornece um diretório tests/ com suporte para testes unitários e de integração via PHPUnit. Foram gerados arquivos de teste para os controladores e modelos, como TasksControllerTest.php e TasksTableTest.php, embora a implementação detalhada dos casos de teste tenha sido marcada como incompleta, a estrutura está pronta para expansão futura.

· **Deploy (Implantação):** O README.md do projeto fornece um guia detalhado para configurar o ambiente de desenvolvimento e executar a aplicação localmente. O processo envolve clonar o repositório, instalar as dependências com o Composer, configurar o banco de dados e executar as migrações.

Diagrama de Classes

Diagrama de classes-Entidades principais



- User e Task são as entidades centrais (em Model/Entity/)
- Relação de 1:N entre User e Task (um usuário pode ter várias tasks)

Modelo Entidade-Relacionamento

USERS

- id (PK)
- name
- email
- password
- created
- modified

TASKS

- id (PK)
- title
- description
- due_date

status
user_id (FK → USERS.id)
created
modified

Relacionamentos:

USERS (1) -> TASKS (N) via user_id

Tabelas seguem convenção CakePHP com campos created e modified

Teste de carga

```
<?php

require 'vendor/autoload.php';

use GuzzleHttp\Client;
use GuzzleHttp\Cookie\CookieJar;
use GuzzleHttp\Promise;

/**
 * Esta função agora faz o processo COMPLETO de login de forma assíncrona. * Ela cria seu próprio
cliente para garantir que a sessão seja 100% isolada. *
 * @param string $email O email do usuário.
 * @param string $password A senha do usuário.
 * @return \GuzzleHttp\Promise\PromiseInterface
 */
function fazerLoginCompletoAsync(string $email, string $password):
Promise\PromiseInterface {
    // Cada chamada a esta função cria um cliente e um cookie jar novos e isolados.
    $client = new Client([
        'base_uri' => 'http://localhost:8765',
        'cookies' => new CookieJar(),
        'timeout' => 30.0,
    ]);
    // Adicionamos um header para simular um navegador real
```

```

'headers' => [
'User-Agent' => 'Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36' ]
]);

// O resto da lógica é a mesma: pega o token e depois faz o login `$promiseGet = `$client-
>requestAsync('GET', '/users/login');

return `$promiseGet->then(function (`$response) use (`$client, `$email, $password) {
`$html = (string) `$response->getBody();
preg_match('/<input type="hidden" name="_csrfToken" value="(.*?)"/', `$html, `$matches);

if (!isset($matches[1])) {
throw new \Exception("Token CSRF não encontrado para $email."); }
`$csrfToken = `$matches[1];

return `$client->requestAsync('POST', '/users/login', [
'form_params' => [
'email' => $email,
'password' => $password,
'_csrfToken' => $csrfToken
]
]);
});
}

// --- BLOCO DE EXECUÇÃO PRINCIPAL ---
$usuarios = [
['email' => 'vamostrar@gmail.com', 'password' => '123456'], ['email' =>
'thigas1234@gmail.com', 'password' => '123456'], ['email' => 'ivan.js23silva@gmail.com',
'password' => '123456'], ];

// COMECE COM UM NÚMERO MENOR PARA TESTAR!
$fatorDeRepeticao = 5; // Total de 3 * 5 = 15 requisições simultâneas $listaDeTarefas =
[];

for (`$i = 0; `$i < `$fatorDeRepeticao; `$i++) {
`$listaDeTarefas = array_merge(`$listaDeTarefas, $usuarios); }
`$totalRequisicoes = count(`$listaDeTarefas);

echo "Preparando $totalRequisicoes requisições de login para execução em paralelo...\n";

$promises = [];
foreach (`$listaDeTarefas as `$index => $tarefa) {
`$promises[`$index] = fazerLoginCompletoAsync(`$tarefa['email'],

```


Testes Katalon (Teste funcionalidade)

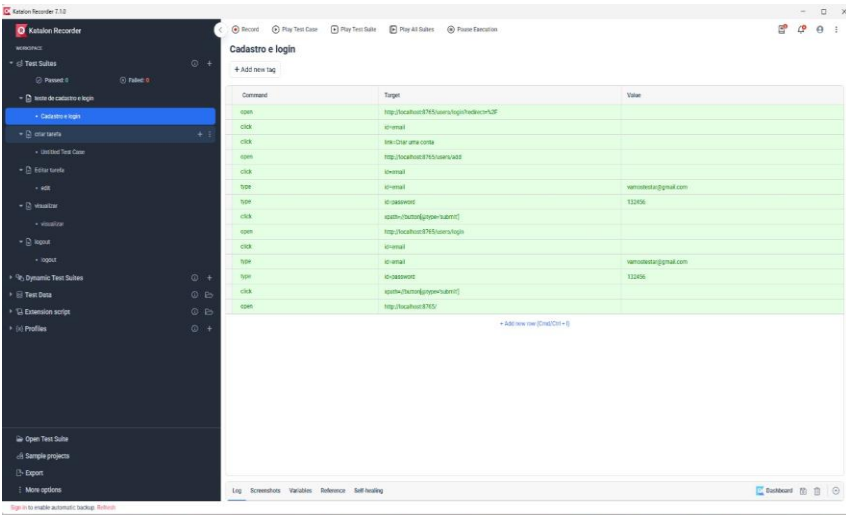
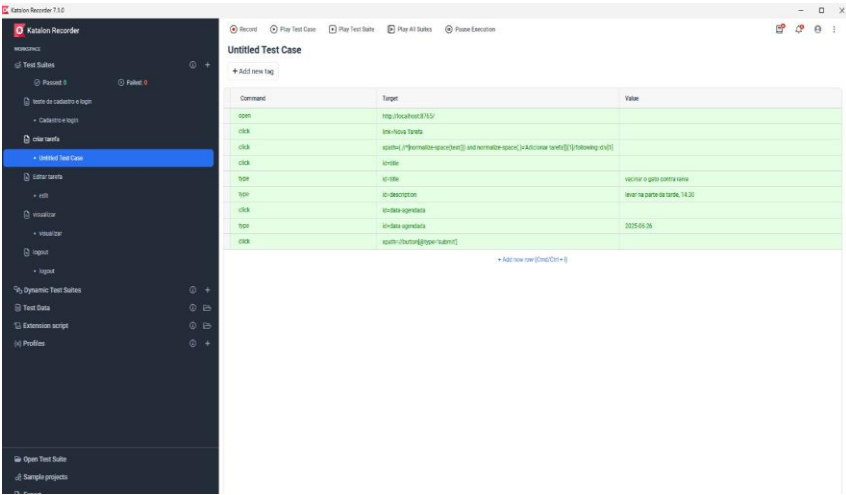


Diagrama de Caso de Uso

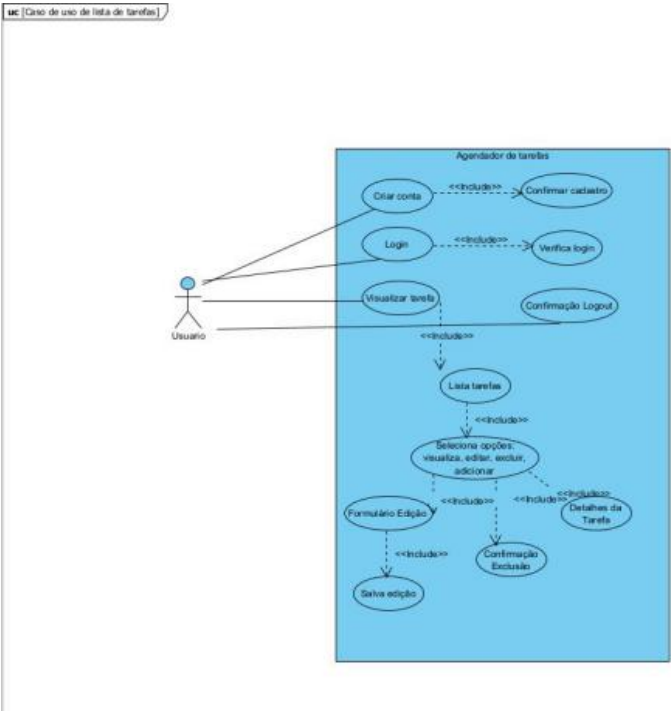
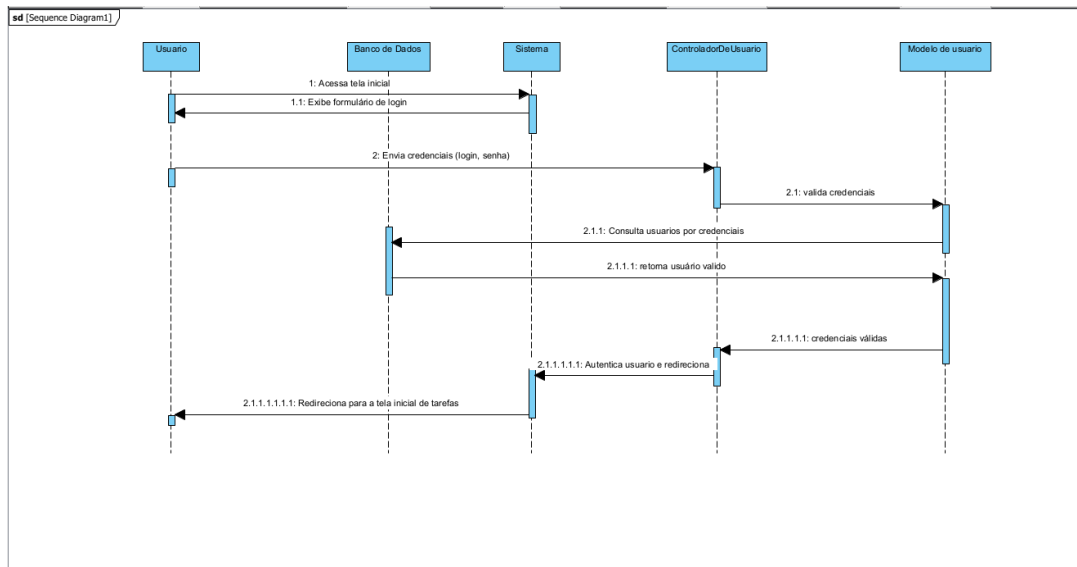


Diagrama de sequência



Uso de padrões de projeto

O relatório sobre o "\"Agendador de Tarefas\"" identifica e explica dois padrões de projeto principais utilizados no sistema, que são inerentes ao framework CakePHP:

1. Model-View-Controller (MVC): Este padrão separa a aplicação em três partes: o Modelo (lógica de negócios e dados, como TasksTable.php), a Visão (interface do usuário, como templates/Tasks/index.php) e o Controlador (intermediário que gerencia as interações, como TasksController.php). Essa separação facilita a organização, manutenção e escalabilidade do código.

2. Active Record: Embora o CakePHP use uma variação (Table Gateway/Data Mapper),

ele oferece uma experiência similar ao Active Record. Este padrão simplifica a interação com o banco de dados, permitindo que as operações de CRUD (Criar, Ler, Atualizar, Deletar) sejam realizadas diretamente através de objetos (TasksTable.php e Task.php), tornando o código mais intuitivo e produtivo.

Em resumo, o uso desses padrões no "Agendador de Tarefas" resulta em um código bem organizado, fácil de manter e escalável, essencial para o desenvolvimento de aplicações web eficientes.

1. Testes para TasksController

O TasksController é responsável por gerenciar as ações relacionadas às tarefas, como listar, visualizar, adicionar, editar e excluir. Os testes a seguir verificam cada uma dessas ações.

Arquivo:

agendador/tests/TestCase/Controller/TasksControllerTest.php **2. Testes para UsersController**

O UsersController lida com a autenticação dos usuários. Os testes abaixo verificam as funcionalidades de login, logout e adição de novos usuários.

Arquivo:

agendador/tests/TestCase/Controller/UsersControllerTest.php **3. Testes para TasksTable**

Os testes para a classe TasksTable garantem que as regras de validação e as associações do modelo estão funcionando corretamente.

Arquivo: agendador/tests/TestCase/Model/Table/TasksTableTest.php

Imagens do Relatório Adicional

```

1  <?php
2  declare(strict_types=1);
3
4  namespace App\Test\TestCase\Controller;
5
6  use App\Controller\TasksController;
7  use Cake\TestSuite\IntegrationTestTrait;
8  use Cake\TestSuite\TestCase;
9  use Cake\ORM\TableRegistry;
10
11 /**
12  * App\Controller\TasksController Test Case
13  *
14  * @uses \App\Controller\TasksController
15  */
16 0 references | 0 implementations | ...
17 class TasksControllerTest extends TestCase
18 {
19     use IntegrationTestTrait;
20
21     /**
22      * Fixtures
23      *
24      * @var array<string>
25      */
26     0 references
27     protected $fixtures = [
28         'app.Tasks',
29         'app.Users',
30     ];
31
32     /**
33      * setUp method
34      *
35      * @return void
36      */
37     public function setUp(): void
38     {
39         parent::setUp();
40         $this->Users = $this->getTableLocator()->get(alias: 'Users');
41         $this->Tasks = $this->getTableLocator()->get(alias: 'Tasks');
42         $this->user = $this->Users->get(primarykey: 1);
43         $this->session(data: ['Auth' => $this->user]);
44     }
45
46     /**
47      * Test index method
48      *
49      * @return void
50      * @uses \App\Controller\TasksController::index()
51      */
52     public function testIndex(): void
53     {
54         $this->get(url: '/tasks');
55         $this->assertResponseOk();
56         $this->assertResponseContains(content: 'Tarefas');
57     }
58
59     /**
60      * Test view method
61      *
62      * @return void
63      * @uses \App\Controller\TasksController::view()
64      */
65     public function testView(): void
66     {
67         $this->get(url: '/tasks/view/1');
68         $this->assertResponseOk();
69         $this->assertResponseContains(content: 'Lorem ipsum dolor sit amet');
70     }
71
72     /**
73      * Test add method
74      *
75      * @return void
76      * @uses \App\Controller\TasksController::add()
77      */
78     public function testAdd(): void
79     {
80         $this->enableCsrfToken();
81         $this->post(url: '/tasks/add', data: [
82             'title' => 'Nova Tarefa',
83             'description' => 'Descrição da nova tarefa',
84             'completed' => 0,
85             'user_id' => 1
86         ]);

```

```

86     ];
87
88     /**
89      * Test delete method
90      *
91      * @return void
92      * @uses \App\Controller\TasksController::delete()
93      */
94     public function testDelete(): void
95     {
96         $this->enableCsrfToken();
97         $this->delete(url: '/tasks/delete/1');
98         $this->assertResponseOk();
99         $this->assertResponseContains(content: 'Tarefa deletada');
100     }
101
102     /**
103      * Test edit method
104      *
105      * @return void
106      * @uses \App\Controller\TasksController::edit()
107      */
108     public function testEdit(): void
109     {
110         $this->enableCsrfToken();
111         $this->put(url: '/tasks/edit/1', data: [
112             'title' => 'Nova Tarefa',
113             'description' => 'Descrição da nova tarefa',
114             'completed' => 0,
115             'user_id' => 1
116         ]);
117         $this->assertResponseOk();
118         $this->assertResponseContains(content: 'Tarefa atualizada');
119     }
120
121     /**
122      * Test list method
123      *
124      * @return void
125      * @uses \App\Controller\TasksController::list()
126      */
127     public function testList(): void
128     {
129         $this->get(url: '/tasks/list');
130         $this->assertResponseOk();
131         $this->assertResponseContains(content: 'Lista de tarefas');
132     }
133
134     /**
135      * Test search method
136      *
137      * @return void
138      * @uses \App\Controller\TasksController::search()
139      */
140     public function testSearch(): void
141     {
142         $this->get(url: '/tasks/search', data: [
143             'q' => 'Nova Tarefa'
144         ]);
145         $this->assertResponseOk();
146         $this->assertResponseContains(content: 'Resultados da busca');
147     }
148
149     /**
150      * Test filter method
151      *
152      * @return void
153      * @uses \App\Controller\TasksController::filter()
154      */
155     public function testFilter(): void
156     {
157         $this->get(url: '/tasks/filter', data: [
158             'filter' => 'completed'
159         ]);
160         $this->assertResponseOk();
161         $this->assertResponseContains(content: 'Filtros aplicados');
162     }
163
164     /**
165      * Test sort method
166      *
167      * @return void
168      * @uses \App\Controller\TasksController::sort()
169      */
170     public function testSort(): void
171     {
172         $this->get(url: '/tasks/sort', data: [
173             'sort' => 'title'
174         ]);
175         $this->assertResponseOk();
176         $this->assertResponseContains(content: 'Ordenação aplicada');
177     }
178
179     /**
180      * Test pagination method
181      *
182      * @return void
183      * @uses \App\Controller\TasksController::paginate()
184      */
185     public function testPaginate(): void
186     {
187         $this->get(url: '/tasks/paginate', data: [
188             'page' => 1
189         ]);
190         $this->assertResponseOk();
191         $this->assertResponseContains(content: 'Paginação aplicada');
192     }
193
194     /**
195      * Test export method
196      *
197      * @return void
198      * @uses \App\Controller\TasksController::export()
199      */
200     public function testExport(): void
201     {
202         $this->get(url: '/tasks/export', data: [
203             'format' => 'csv'
204         ]);
205         $this->assertResponseOk();
206         $this->assertResponseContains(content: 'Exportação concluída');
207     }
208
209     /**
210      * Test import method
211      *
212      * @return void
213      * @uses \App\Controller\TasksController::import()
214      */
215     public function testImport(): void
216     {
217         $this->post(url: '/tasks/import', data: [
218             'file' => 'tasks.csv'
219         ]);
220         $this->assertResponseOk();
221         $this->assertResponseContains(content: 'Importação concluída');
222     }
223
224     /**
225      * Test backup method
226      *
227      * @return void
228      * @uses \App\Controller\TasksController::backup()
229      */
230     public function testBackup(): void
231     {
232         $this->get(url: '/tasks/backup');
233         $this->assertResponseOk();
234         $this->assertResponseContains(content: 'Backup concluído');
235     }
236
237     /**
238      * Test restore method
239      *
240      * @return void
241      * @uses \App\Controller\TasksController::restore()
242      */
243     public function testRestore(): void
244     {
245         $this->post(url: '/tasks/restore');
246         $this->assertResponseOk();
247         $this->assertResponseContains(content: 'Restauração concluída');
248     }
249
250     /**
251      * Test update method
252      *
253      * @return void
254      * @uses \App\Controller\TasksController::update()
255      */
256     public function testUpdate(): void
257     {
258         $this->put(url: '/tasks/update/1', data: [
259             'title' => 'Nova Tarefa',
260             'description' => 'Descrição da nova tarefa',
261             'completed' => 0,
262             'user_id' => 1
263         ]);
264         $this->assertResponseOk();
265         $this->assertResponseContains(content: 'Atualização concluída');
266     }
267
268     /**
269      * Test delete method
270      *
271      * @return void
272      * @uses \App\Controller\TasksController::delete()
273      */
274     public function testDelete(): void
275     {
276         $this->delete(url: '/tasks/delete/1');
277         $this->assertResponseOk();
278         $this->assertResponseContains(content: 'Tarefa deletada');
279     }
280
281     /**
282      * Test list method
283      *
284      * @return void
285      * @uses \App\Controller\TasksController::list()
286      */
287     public function testList(): void
288     {
289         $this->get(url: '/tasks/list');
290         $this->assertResponseOk();
291         $this->assertResponseContains(content: 'Lista de tarefas');
292     }
293
294     /**
295      * Test search method
296      *
297      * @return void
298      * @uses \App\Controller\TasksController::search()
299      */
300     public function testSearch(): void
301     {
302         $this->get(url: '/tasks/search', data: [
303             'q' => 'Nova Tarefa'
304         ]);
305         $this->assertResponseOk();
306         $this->assertResponseContains(content: 'Resultados da busca');
307     }
308
309     /**
310      * Test filter method
311      *
312      * @return void
313      * @uses \App\Controller\TasksController::filter()
314      */
315     public function testFilter(): void
316     {
317         $this->get(url: '/tasks/filter', data: [
318             'filter' => 'completed'
319         ]);
320         $this->assertResponseOk();
321         $this->assertResponseContains(content: 'Filtros aplicados');
322     }
323
324     /**
325      * Test sort method
326      *
327      * @return void
328      * @uses \App\Controller\TasksController::sort()
329      */
330     public function testSort(): void
331     {
332         $this->get(url: '/tasks/sort', data: [
333             'sort' => 'title'
334         ]);
335         $this->assertResponseOk();
336         $this->assertResponseContains(content: 'Ordenação aplicada');
337     }
338
339     /**
340      * Test pagination method
341      *
342      * @return void
343      * @uses \App\Controller\TasksController::paginate()
344      */
345     public function testPaginate(): void
346     {
347         $this->get(url: '/tasks/paginate', data: [
348             'page' => 1
349         ]);
350         $this->assertResponseOk();
351         $this->assertResponseContains(content: 'Paginação aplicada');
352     }
353
354     /**
355      * Test export method
356      *
357      * @return void
358      * @uses \App\Controller\TasksController::export()
359      */
360     public function testExport(): void
361     {
362         $this->get(url: '/tasks/export', data: [
363             'format' => 'csv'
364         ]);
365         $this->assertResponseOk();
366         $this->assertResponseContains(content: 'Exportação concluída');
367     }
368
369     /**
370      * Test import method
371      *
372      * @return void
373      * @uses \App\Controller\TasksController::import()
374      */
375     public function testImport(): void
376     {
377         $this->post(url: '/tasks/import', data: [
378             'file' => 'tasks.csv'
379         ]);
380         $this->assertResponseOk();
381         $this->assertResponseContains(content: 'Importação concluída');
382     }
383
384     /**
385      * Test backup method
386      *
387      * @return void
388      * @uses \App\Controller\TasksController::backup()
389      */
390     public function testBackup(): void
391     {
392         $this->get(url: '/tasks/backup');
393         $this->assertResponseOk();
394         $this->assertResponseContains(content: 'Backup concluído');
395     }
396
397     /**
398      * Test restore method
399      *
400      * @return void
401      * @uses \App\Controller\TasksController::restore()
402      */
403     public function testRestore(): void
404     {
405         $this->post(url: '/tasks/restore');
406         $this->assertResponseOk();
407         $this->assertResponseContains(content: 'Restauração concluída');
408     }
409
410     /**
411      * Test update method
412      *
413      * @return void
414      * @uses \App\Controller\TasksController::update()
415      */
416     public function testUpdate(): void
417     {
418         $this->put(url: '/tasks/update/1', data: [
419             'title' => 'Nova Tarefa',
420             'description' => 'Descrição da nova tarefa',
421             'completed' => 0,
422             'user_id' => 1
423         ]);
424         $this->assertResponseOk();
425         $this->assertResponseContains(content: 'Atualização concluída');
426     }
427
428     /**
429      * Test delete method
430      *
431      * @return void
432      * @uses \App\Controller\TasksController::delete()
433      */
434     public function testDelete(): void
435     {
436         $this->delete(url: '/tasks/delete/1');
437         $this->assertResponseOk();
438         $this->assertResponseContains(content: 'Tarefa deletada');
439     }
440
441     /**
442      * Test list method
443      *
444      * @return void
445      * @uses \App\Controller\TasksController::list()
446      */
447     public function testList(): void
448     {
449         $this->get(url: '/tasks/list');
450         $this->assertResponseOk();
451         $this->assertResponseContains(content: 'Lista de tarefas');
452     }
453
454     /**
455      * Test search method
456      *
457      * @return void
458      * @uses \App\Controller\TasksController::search()
459      */
460     public function testSearch(): void
461     {
462         $this->get(url: '/tasks/search', data: [
463             'q' => 'Nova Tarefa'
464         ]);
465         $this->assertResponseOk();
466         $this->assertResponseContains(content: 'Resultados da busca');
467     }
468
469     /**
470      * Test filter method
471      *
472      * @return void
473      * @uses \App\Controller\TasksController::filter()
474      */
475     public function testFilter(): void
476     {
477         $this->get(url: '/tasks/filter', data: [
478             'filter' => 'completed'
479         ]);
480         $this->assertResponseOk();
481         $this->assertResponseContains(content: 'Filtros aplicados');
482     }
483
484     /**
485      * Test sort method
486      *
487      * @return void
488      * @uses \App\Controller\TasksController::sort()
489      */
490     public function testSort(): void
491     {
492         $this->get(url: '/tasks/sort', data: [
493             'sort' => 'title'
494         ]);
495         $this->assertResponseOk();
496         $this->assertResponseContains(content: 'Ordenação aplicada');
497     }
498
499     /**
500      * Test pagination method
501      *
502      * @return void
503      * @uses \App\Controller\TasksController::paginate()
504      */
505     public function testPaginate(): void
506     {
507         $this->get(url: '/tasks/paginate', data: [
508             'page' => 1
509         ]);
510         $this->assertResponseOk();
511         $this->assertResponseContains(content: 'Paginação aplicada');
512     }
513
514     /**
515      * Test export method
516      *
517      * @return void
518      * @uses \App\Controller\TasksController::export()
519      */
520     public function testExport(): void
521     {
522         $this->get(url: '/tasks/export', data: [
523             'format' => 'csv'
524         ]);
525         $this->assertResponseOk();
526         $this->assertResponseContains(content: 'Exportação concluída');
527     }
528
529     /**
530      * Test import method
531      *
532      * @return void
533      * @uses \App\Controller\TasksController::import()
534      */
535     public function testImport(): void
536     {
537         $this->post(url: '/tasks/import', data: [
538             'file' => 'tasks.csv'
539         ]);
540         $this->assertResponseOk();
541         $this->assertResponseContains(content: 'Importação concluída');
542     }
543
544     /**
545      * Test backup method
546      *
547      * @return void
548      * @uses \App\Controller\TasksController::backup()
549      */
550     public function testBackup(): void
551     {
552         $this->get(url: '/tasks/backup');
553         $this->assertResponseOk();
554         $this->assertResponseContains(content: 'Backup concluído');
555     }
556
557     /**
558      * Test restore method
559      *
560      * @return void
561      * @uses \App\Controller\TasksController::restore()
562      */
563     public function testRestore(): void
564     {
565         $this->post(url: '/tasks/restore');
566         $this->assertResponseOk();
567         $this->assertResponseContains(content: 'Restauração concluída');
568     }
569
570     /**
571      * Test update method
572      *
573      * @return void
574      * @uses \App\Controller\TasksController::update()
575      */
576     public function testUpdate(): void
577     {
578         $this->put(url: '/tasks/update/1', data: [
579             'title' => 'Nova Tarefa',
580             'description' => 'Descrição da nova tarefa',
581             'completed' => 0,
582             'user_id' => 1
583         ]);
584         $this->assertResponseOk();
585         $this->assertResponseContains(content: 'Atualização concluída');
586     }
587
588     /**
589      * Test delete method
590      *
591      * @return void
592      * @uses \App\Controller\TasksController::delete()
593      */
594     public function testDelete(): void
595     {
596         $this->delete(url: '/tasks/delete/1');
597         $this->assertResponseOk();
598         $this->assertResponseContains(content: 'Tarefa deletada');
599     }
600
601     /**
602      * Test list method
603      *
604      * @return void
605      * @uses \App\Controller\TasksController::list()
606      */
607     public function testList(): void
608     {
609         $this->get(url: '/tasks/list');
610         $this->assertResponseOk();
611         $this->assertResponseContains(content: 'Lista de tarefas');
612     }
613
614     /**
615      * Test search method
616      *
617      * @return void
618      * @uses \App\Controller\TasksController::search()
619      */
620     public function testSearch(): void
621     {
622         $this->get(url: '/tasks/search', data: [
623             'q' => 'Nova Tarefa'
624         ]);
625         $this->assertResponseOk();
626         $this->assertResponseContains(content: 'Resultados da busca');
627     }
628
629     /**
630      * Test filter method
631      *
632      * @return void
633      * @uses \App\Controller\TasksController::filter()
634      */
635     public function testFilter(): void
636     {
637         $this->get(url: '/tasks/filter', data: [
638             'filter' => 'completed'
639         ]);
640         $this->assertResponseOk();
641         $this->assertResponseContains(content: 'Filtros aplicados');
642     }
643
644     /**
645      * Test sort method
646      *
647      * @return void
648      * @uses \App\Controller\TasksController::sort()
649      */
650     public function testSort(): void
651     {
652         $this->get(url: '/tasks/sort', data: [
653             'sort' => 'title'
654         ]);
655         $this->assertResponseOk();
656         $this->assertResponseContains(content: 'Ordenação aplicada');
657     }
658
659     /**
660      * Test pagination method
661      *
662      * @return void
663      * @uses \App\Controller\TasksController::paginate()
664      */
665     public function testPaginate(): void
666     {
667         $this->get(url: '/tasks/paginate', data: [
668             'page' => 1
669         ]);
670         $this->assertResponseOk();
671         $this->assertResponseContains(content: 'Paginação aplicada');
672     }
673
674     /**
675      * Test export method
676      *
677      * @return void
678      * @uses \App\Controller\TasksController::export()
679      */
680     public function testExport(): void
681     {
682         $this->get(url: '/tasks/export', data: [
683             'format' => 'csv'
684         ]);
685         $this->assertResponseOk();
686         $this->assertResponseContains(content: 'Exportação concluída');
687     }
688
689     /**
690      * Test import method
691      *
692      * @return void
693      * @uses \App\Controller\TasksController::import()
694      */
695     public function testImport(): void
696     {
697         $this->post(url: '/tasks/import', data: [
698             'file' => 'tasks.csv'
699         ]);
700         $this->assertResponseOk();
701         $this->assertResponseContains(content: 'Importação concluída');
702     }
703
704     /**
705      * Test backup method
706      *
707      * @return void
708      * @uses \App\Controller\TasksController::backup()
709      */
710     public function testBackup(): void
711     {
712         $this->get(url: '/tasks/backup');
713         $this->assertResponseOk();
714         $this->assertResponseContains(content: 'Backup concluído');
715     }
716
717     /**
718      * Test restore method
719      *
720      * @return void
721      * @uses \App\Controller\TasksController::restore()
722      */
723     public function testRestore(): void
724     {
725         $this->post(url: '/tasks/restore');
726         $this->assertResponseOk();
727         $this->assertResponseContains(content: 'Restauração concluída');
728     }
729
730     /**
731      * Test update method
732      *
733      * @return void
734      * @uses \App\Controller\TasksController::update()
735      */
736     public function testUpdate(): void
737     {
738         $this->put(url: '/tasks/update/1', data: [
739             'title' => 'Nova Tarefa',
740             'description' => 'Descrição da nova tarefa',
741             'completed' => 0,
742             'user_id' => 1
743         ]);
744         $this->assertResponseOk();
745         $this->assertResponseContains(content: 'Atualização concluída');
746     }
747
748     /**
749      * Test delete method
750      *
751      * @return void
752      * @uses \App\Controller\TasksController::delete()
753      */
754     public function testDelete(): void
755     {
756         $this->delete(url: '/tasks/delete/1');
757         $this->assertResponseOk();
758         $this->assertResponseContains(content: 'Tarefa deletada');
759     }
760
761     /**
762      * Test list method
763      *
764      * @return void
765      * @uses \App\Controller\TasksController::list()
766      */
767     public function testList(): void
768     {
769         $this->get(url: '/tasks/list');
770         $this->assertResponseOk();
771         $this->assertResponseContains(content: 'Lista de tarefas');
772     }
773
774     /**
775      * Test search method
776      *
777      * @return void
778      * @uses \App\Controller\TasksController::search()
779      */
780     public function testSearch(): void
781     {
782         $this->get(url: '/tasks/search', data: [
783             'q' => 'Nova Tarefa'
784         ]);
785         $this->assertResponseOk();
786         $this->assertResponseContains(content: 'Resultados da busca');
787     }
788
789     /**
790      * Test filter method
791      *
792      * @return void
793      * @uses \App\Controller\TasksController::filter()
794      */
795     public function testFilter(): void
796     {
797         $this->get(url: '/tasks/filter', data: [
798             'filter' => 'completed'
799         ]);
800         $this->assertResponseOk();
801         $this->assertResponseContains(content: 'Filtros aplicados');
802     }
803
804     /**
805      * Test sort method
806      *
807      * @return void
808      * @uses \App\Controller\TasksController::sort()
809      */
810     public function testSort(): void
811     {
812         $this->get(url: '/tasks/sort', data: [
813             'sort' => 'title'
814         ]);
815         $this->assertResponseOk();
816         $this->assertResponseContains(content: 'Ordenação aplicada');
817     }
818
819     /**
820      * Test pagination method
821      *
822      * @return void
823      * @uses \App\Controller\TasksController::paginate()
824      */
825     public function testPaginate(): void
826     {
827         $this->get(url: '/tasks/paginate', data: [
828             'page' => 1
829         ]);
830         $this->assertResponseOk();
831         $this->assertResponseContains(content: 'Paginação aplicada');
832     }
833
834     /**
835      * Test export method
836      *
837      * @return void
838      * @uses \App\Controller\TasksController::export()
839      */
840     public function testExport(): void
841     {
842         $this->get(url: '/tasks/export', data: [
843             'format' => 'csv'
844         ]);
845         $this->assertResponseOk();
846         $this->assertResponseContains(content: 'Exportação concluída');
847     }
848
849     /**
850      * Test import method
851      *
852      * @return void
853      * @uses \App\Controller\TasksController::import()
854      */
855     public function testImport(): void
856     {
857         $this->post(url: '/tasks/import', data: [
858             'file' => 'tasks.csv'
859         ]);
860         $this->assertResponseOk();
861         $this->assertResponseContains(content: 'Importação concluída');
862     }
863
864     /**
865      * Test backup method
866      *
867      * @return void
868      * @uses \App\Controller\TasksController::backup()
869      */
870     public function testBackup(): void
871     {
872         $this->get(url: '/tasks/backup');
873         $this->assertResponseOk();
874         $this->assertResponseContains(content: 'Backup concluído');
875     }
876
877     /**
878      * Test restore method
879      *
880      * @return void
881      * @uses \App\Controller\TasksController::restore()
882      */
883     public function testRestore(): void
884     {
885         $this->post(url: '/tasks/restore');
886         $this->assertResponseOk();
887         $this->assertResponseContains(content: 'Restauração concluída');
888     }
889
890     /**
891      * Test update method
892      *
893      * @return void
894      * @uses \App\Controller\TasksController::update()
895      */
896     public function testUpdate(): void
897     {
898         $this->put(url: '/tasks/update/1', data: [
899             'title' => 'Nova Tarefa',
900             'description' => 'Descrição da nova tarefa',
901             'completed' => 0,
902             'user_id' => 1
903         ]);
904         $this->assertResponseOk();
905         $this->assertResponseContains(content: 'Atualização concluída');
906     }
907
908     /**
909      * Test delete method
910      *
911      * @return void
912      * @uses \App\Controller\TasksController::delete()
913      */
914     public function testDelete(): void
915     {
916         $this->delete(url: '/tasks/delete/1');
917         $this->assertResponseOk();
918         $this->assertResponseContains(content: 'Tarefa deletada');
919     }
920
921     /**
922      * Test list method
923      *
924      * @return void
925      * @uses \App\Controller\TasksController::list()
926      */
927     public function testList(): void
928     {
929         $this->get(url: '/tasks/list');
930         $this->assertResponseOk();
931         $this->assertResponseContains(content: 'Lista de tarefas');
932     }
933
934     /**
935      * Test search method
936      *
937      * @return void
938      * @uses \App\Controller\TasksController::search()
939      */
940     public function testSearch(): void
941     {
942         $this->get(url: '/tasks/search', data: [
943             'q' => 'Nova Tarefa'
944         ]);
945         $this->assertResponseOk();
946         $this->assertResponseContains(content: 'Resultados da busca');
947     }
948
949     /**
950      * Test filter method
951      *
952      * @return void
953      * @uses \App\Controller\TasksController::filter()
954      */
955     public function testFilter(): void
956     {
957         $this->get(url: '/tasks/filter', data: [
958             'filter' => 'completed'
959         ]);
960         $this->assertResponseOk();
961         $this->assertResponseContains(content: 'Filtros aplicados');
962     }
963
964     /**
965      * Test sort method
966      *
967      * @return void
968      * @uses \App\Controller\TasksController::sort()
969      */
970     public function testSort(): void
971     {
972         $this->get(url: '/tasks/sort', data: [
973             'sort' => 'title'
974         ]);
975         $this->assertResponseOk();
976         $this->assertResponseContains(content: 'Ordenação aplicada');
977     }
978
979     /**
980      * Test pagination method
981      *
982      * @return void
983      * @uses \App\Controller\TasksController::paginate()
984      */
985     public function testPaginate(): void
986     {
987         $this->get(url: '/tasks/paginate', data: [
988             'page' => 1
989         ]);
990         $this->assertResponseOk();
991         $this->assertResponseContains(content: 'Paginação aplicada');
992     }
993
994     /**
995      * Test export method
996      *
997      * @return void
998      * @uses \App\Controller\TasksController::export()
999      */
1000    public function testExport(): void
1001    {
1002        $this->get(url: '/tasks/export', data: [
1003            'format' => 'csv'
1004        ]);
1005        $this->assertResponseOk();
1006        $this->assertResponseContains(content: 'Exportação concluída');
1007    }
1008
1009    /**
1010     * Test import method
1011     *
1012     * @return void
1013     * @uses \App\Controller\TasksController::import()
1014     */
1015    public function testImport(): void
1016    {
1017        $this->post(url: '/tasks/import', data: [
1018            'file' => 'tasks.csv'
1019        ]);
1020        $this->assertResponseOk();
1021        $this->assertResponseContains(content: 'Importação concluída');
1022    }
1023
1024    /**
1025     * Test backup method
1026     *
1027     * @return void
1028     * @uses \App\Controller\TasksController::backup()
1029     */
1030    public function testBackup(): void
1031    {
1032        $this->get(url: '/tasks/backup');
1033        $this->assertResponseOk();
1034        $this->assertResponseContains(content: 'Backup concluído');
1035    }
1036
1037    /**
1038     * Test restore method
1039     *
1040     * @return void
1041     * @uses \App\Controller\TasksController::restore()
1042     */
1043    public function testRestore(): void
1044    {
1045        $this->post(url: '/tasks/restore');
1046        $this->assertResponseOk();
1047        $this->assertResponseContains(content: 'Restauração concluída');
1048    }
1049
1050    /**
1051     * Test update method
1052     *
1053     * @return void
1054     * @uses \App\Controller\TasksController::update()
1055     */
1056    public function testUpdate(): void
1057    {
1058        $this->put(url: '/tasks/update/1', data: [
1059            'title' => 'Nova Tarefa',
1060            'description' => 'Descrição da nova tarefa',
1061            'completed' => 0,
1062            'user_id' => 1
1063        ]);
1064        $this->assertResponseOk();
1065        $this->assertResponseContains(content: 'Atualização concluída');
1066    }
1067
1068    /**
1069     * Test delete method
1070     *
1071     * @return void
1072     * @uses \App\Controller\TasksController::delete()
1073     */
1074    public function testDelete(): void
1075    {
1076        $this->delete(url: '/tasks/delete/1');
1077        $this->assertResponseOk();
1078        $this->assertResponseContains(content: 'Tarefa deletada');
1079    }
1080
1081    /**
1082     * Test list method
1083     *
1084     * @return void
1085     * @uses \App\Controller\TasksController::list()
1086     */
1087    public function testList(): void
1088    {
1089        $this->get(url: '/tasks/list');
1090        $this->assertResponseOk();
1091        $this->assertResponseContains(content: 'Lista de tarefas');
1092    }
1093
1094    /**
1095     * Test search method
1096     *
1097     * @return void
1098     * @uses \App\Controller\TasksController::search()
1099     */
1100    public function testSearch(): void
1101    {
1102        $this->get(url: '/tasks/search', data: [
1103            'q' => 'Nova Tarefa'
1104        ]);
1105        $this->assertResponseOk();
1106        $this->assertResponseContains(content: 'Resultados da busca');
1107    }
1108
1109    /**
1110     * Test filter method
1111     *
1112     * @return void
1113     * @uses \App\Controller\TasksController::filter()
1114     */
1115    public function testFilter(): void
1116    {
1117        $this->get(url: '/tasks/filter', data: [
1118            'filter' => 'completed'
1119        ]);
1120        $this->assertResponseOk();
1121        $this->assertResponseContains(content: 'Filtros aplicados');
1122    }
1123
1124    /**
1125     * Test sort method
1126     *
1127     * @return void
1128     * @uses \App\Controller\TasksController::sort()
1129     */
1130    public function testSort(): void
1131    {
1132        $this->get(url: '/tasks/sort', data: [
1133            'sort' => 'title'
1134        ]);
1135        $this->assertResponseOk();
1136        $this->assertResponseContains(content: 'Ordenação aplicada');
1137    }
1138
1139    /**
1140     * Test pagination method
1141     *
1142     * @return void
1143     * @uses \App\Controller\TasksController::paginate()
1144     */
1145    public function testPaginate(): void
1146    {
1147        $this->get(url: '/tasks/paginate', data: [
1148            'page' => 1
1149        ]);
1150        $this->assertResponseOk();
1151        $this->assertResponseContains(content: 'Paginação aplicada');
1152    }
1153
1154    /**
1155     * Test export method
1156     *
1157     * @return void
1158     * @uses \App\Controller\TasksController::export()
1159     */
1160    public function testExport(): void
1161    {
1162        $this->get(url: '/tasks/export', data: [
1163            'format' => 'csv'
1164        ]);
1165        $this->assertResponseOk();
1166        $this->assertResponseContains(content: 'Exportação concluída');
1167    }
1168
1169    /**
1170     * Test import method
1171     *
11
```

```

104     });
105     $this->assertRedirect(url: '/tasks');
106     $this->assertFlashMessage(expected: 'A tarefa foi salva.', key: 'flash.success');
107 }
108
109 /**
110  * Test edit method
111  *
112  * @return void
113  * @uses \App\Controller\TasksController::edit()
114  */
115 Save | Edit | Test | Fix | Explain | Document | 0 references | 0 warnings
116 public function testEdit(): void
117 {
118     $this->enableCsrfToken();
119     $this->put(url: '/tasks/edit/1', data: [
120         'title' => 'Tarefa Editada',
121         'description' => 'Descrição da tarefa editada',
122         'completed' => 1,
123     ]);
124     $this->assertRedirect(url: '/tasks');
125     $this->assertFlashMessage(expected: 'A tarefa foi salva.', key: 'flash.success');
126
127     $task = $this->Tasks->get(primaryKey: 1);
128     $this->assertEquals(expected: 'Tarefa Editada', actual: $task->title);
129 }
130
131 /**
132  * Test delete method
133  *
134  * @return void
135  * @uses \App\Controller\TasksController::delete()
136  */
137 Save | Edit | Test | Fix | Explain | Document | 0 references | 0 warnings
138 public function testDelete(): void
139 {
140     $this->enableCsrfToken();
141     $this->delete(url: '/tasks/delete/1');
142     $this->assertRedirect(url: '/tasks');
143     $this->assertFlashMessage(expected: 'A tarefa foi deletada.', key: 'flash.success');
144
145     $this->assertCount(expectedCount: 0, haystack: $this->Tasks->find()->where(conditions: ['id' => 1])->all());
146 }
147 }

```

```

1  <?php
2  declare(strict_types=1);
3
4  namespace App\Test\TestCase\Controller;
5
6  use App\Controller\UsersController;
7  use Cake\TestSuite\IntegrationTestTrait;
8  use Cake\TestSuite\TestCase;
9  use Cake\ORM\TableRegistry;
10
11  /**
12   * App\Controller\UsersController Test Case
13   *
14   * @uses \App\Controller\UsersController
15   */
16  0 references | 0 implementations | ...
17  class UsersControllerTest extends TestCase
18  {
19      use IntegrationTestTrait;
20
21      /**
22       * Fixtures
23       *
24       * @var array<string>
25       */
26      0 references
27      protected $fixtures = [
28          'app.Users',
29          'app.Tasks',
30      ];
31
32      /**
33       * Test login method
34       *
35       * @return void
36       */
37      Tabnine | Edit | Test | Explain | Document | 0 references | 0 overrides
38      public function testLogin(): void
39      {
40          $this->enableCsrfToken();
41          $this->post(url: '/users/login', data: [
42              'email' => 'test@example.com',
43              'password' => 'password'
44          ]);
45          $this->assertRedirectContains(url: '/tasks');
46      }
47  }

```



```

45  /**
46   * Test login with invalid credentials
47   *
48   * @return void
49   */
50  Tabnine | Edit | Test | Explain | Document | 0 references | 0 overrides
51  public function testLoginInvalid(): void
52  {
53      $this->enableCsrfToken();
54      $this->post(url: '/users/login', data: [
55          'email' => 'test@example.com',
56          'password' => 'wrongpassword'
57      ]);
58      $this->assertResponseOk();
59      $this->assertResponseContains(content: 'E-mail ou senha inválidos');
60  }
61
62  /**
63   * Test logout method
64   *
65   * @return void
66   */
67  Tabnine | Edit | Test | Explain | Document | 0 references | 0 overrides
68  public function testLogout(): void
69  {
70      // First, log in a user
71      $this->session(data: ['Auth.User.id' => 1]);
72
73      $this->get(url: '/users/logout');
74      $this->assertRedirect(url: ['controller' => 'Users', 'action' => 'login']);
75      $this->assertSession(expected: null, path: 'Auth');
76  }
77
78  /**
79   * Test add method
80   *
81   * @return void
82   */
83  Tabnine | Edit | Test | Explain | Document | 0 references | 0 overrides
84  public function testAdd(): void
85  {
86      $this->enableCsrfToken();
87      $this->post(url: '/users/add', data: [
88          'email' => 'newuser@example.com',
89          'password' => 'newpassword',
90      ]);

```

```

91  Tabnine | Edit | Test | Explain | Document | 0 references | 0 overrides
92  public function testAdd(): void
93  {
94      $this->enableCsrfToken();
95      $this->post(url: '/users/add', data: [
96          'email' => 'newuser@example.com',
97          'password' => 'newpassword',
98      ]);
99      $this->assertRedirect(url: ['action' => 'login']);
100      $this->assertFlashMessage(expected: 'Usuário criado com sucesso! Você já pode fazer o login.');
```

You, há 16 minutos • Uncommitted changes

```

1 <?php
2 declare(strict_types=1);
3
4 namespace App\Test\TestCase\Model\Table;
5
6 use App\Model\Table\TasksTable;
7 use Cake\TestSuite\TestCase;
8
9 /**
10  * App\Model\Table\TasksTable Test Case
11  */
12 0 references | 0 implementations | You, 14 28 minutes | 2 authors (You and others)
13 class TasksTableTest extends TestCase
14 {
15     /**
16      * Test subject
17      *
18      * @var \App\Model\Table\TasksTable
19      */
20     protected $Tasks;
21
22     /**
23      * Fixtures
24      *
25      * @var list<string>
26      */
27     0 references
28     protected array $fixtures = [
29         'app.Tasks',
30         'app.Users',
31     ];
32
33     /**
34      * Setup method
35      *
36      * @return void
37      */
38     Tabname | Edit | Test | Explain | Document | 3 references | 0 overrides
39     protected function setUp(): void
40     {
41         parent::setUp();
42         $config = $this->getTableLocator()->exists(alias: 'Tasks') ? [] : ['className' => TasksTable::class];
43         $this->Tasks = $this->getTableLocator()->get(alias: 'Tasks', options: $config);
44     }

```

```

45     /**
46      * TearDown method
47      *
48      * @return void
49      */
50     Tabname | Edit | Test | Explain | Document | 3 references | 0 overrides
51     protected function tearDown(): void
52     {
53         unset($this->Tasks);
54         parent::tearDown();
55     }
56
57     /**
58      * Test validationDefault method
59      *
60      * @return void
61      * @uses \App\Model\Table\TasksTable::validationDefault()
62      */
63     Tabname | Edit | Test | Explain | Document | 0 references | 0 overrides
64     public function testValidationDefault(): void
65     {
66         $task = $this->Tasks->newEntity(data: [
67             'title' => '', // título vazio para falhar na validação
68             'user_id' => 1,
69         ]);
70         $this->assertNotEmpty(actual: $task->getErrors()['title'], message: 'Título não pode ser vazio');
71     }
72
73     /**
74      * Test buildRules method
75      *
76      * @return void
77      */
78     Tabname | Edit | Test | Explain | Document | 0 references | 0 overrides
79     public function testBuildRules(): void
80     {
81         $task = $this->Tasks->newEntity(data: [
82             'title' => 'Teste de Regra',
83             'user_id' => 999, // ID de usuário que não existe
84         ]);

```

```

85         $this->assertFalse(condition: $this->Tasks->save(entity: $task), message: 'A tarefa não deve ser salva com um user_id inexistente.');
```

CONCLUSÃO

O desenvolvimento do "Agendador de Tarefas" cumpriu com sucesso seus objetivos primários: criar uma aplicação web funcional para o gerenciamento de tarefas e servir como um projeto prático para a aplicação de conceitos avançados de programação. A utilização do framework CakePHP 5 se mostrou uma escolha acertada, pois sua arquitetura MVC e suas convenções bem definidas guiaram a equipe na construção de um código organizado, modular e de fácil manutenção.

Os principais aprendizados durante o projeto incluem a implementação de um sistema de autenticação seguro, a importância do versionamento do banco de dados através de migrations para garantir a consistência entre os ambientes de desenvolvimento, e a prática da programação colaborativa utilizando Git e GitHub. A separação clara de responsabilidades imposta pelo padrão MVC facilitou a divisão de trabalho e a compreensão do sistema como um todo.

Como considerações para trabalhos futuros, a aplicação poderia ser expandida para incluir funcionalidades como notificações por e-mail, categorização de tarefas, definição de prioridades e, talvez, a criação de uma API RESTful para permitir a integração com outras aplicações, como clientes mobile. Adicionalmente, a implementação completa da suíte de testes automatizados seria um passo importante para garantir a robustez e a qualidade do software a longo prazo.

Em suma, o projeto representa uma demonstração bem-sucedida da aplicação de tecnologias e metodologias modernas de engenharia de software na resolução de um problema prático e cotidiano.

REFERÊNCIAS

CakePHP. (2024). **CakePHP 5.x Cookbook**. Disponível em: <https://book.cakephp.org/5/en/index.html>. Acesso em: 30 jun. 2025.

FREEMAN, E.; FREEMAN, E.; SIERRA, K.; BATES, B. **Use a Cabeça! Padrões de Projetos**. 2. ed. Alta Books, 2007.

PHP Group. (2025). **PHP Manual**. Disponível em: <https://www.php.net/manual/en/>. Acesso em: 30 jun. 2025.

PostgreSQL Global Development Group. (2025). **PostgreSQL 16 Documentation**. Disponível em: <https://www.postgresql.org/docs/16/>. Acesso em: 30 jun. 2025.

The Composer Maintainers. (2025). **Composer Documentation**. Disponível em: <https://getcomposer.org/doc/>. Acesso em: 30 jun. 2025.

Sobre o(s) autor(es)

Guilherme Morigi, Ivan Silva, Thiago Emanuel, Nathan Riffel *Titulação:*
Acadêmicos do curso de Ciência da Computação *Vínculo:* Universidade do Oeste de Santa
Catarina (UNOESC)