



Пловдивски университет „Паисий Хилендарски”
Факултет по математика и информатика

Курсова работа

По дисциплина: „Фреймуърк системи“

На Фреймуърк: „CakePHP“

Изготвил: Атанас Воденичаров;
Иво Яшев; Георги Караджов
Факултетен номер: 1901681065;
1901681069; 1901681063

Специалност: Софтуерни
технологии и дизайн

Пловдив 2023

Съдържание

| | |
|---|----|
| 1. Структура на папката на CakePHP | 2 |
| 2. Какво е CakePHP ?..... | 3 |
| 3. История..... | 4 |
| 4. The Model Layer(Моделен слой) | 4 |
| 5. The View Layer (Слоят за преглед) | 5 |
| 6. The Controller view (Слоят на контролера) | 6 |
| 7. Инсталация на CakePHP | 6 |
| 8. Command-line installation | 7 |
| 9. Създаване на CakePHP проект | 7 |
| 10. Технологии и принципи, които използва CakePHP | 8 |
| 11. Routing(Ротиране/Маршрутизиране)..... | 9 |
| 12. Работа с бази данни | 10 |
| 13. Компоненти, които предоставя рамката | 10 |
| a. Конфигуриране на компоненти (Configuring Components) | 11 |
| b. Псевдоним на компоненти (Aliasing Components)..... | 12 |
| c. Използване на компонентите | 12 |
| d. Създаване на компоненти..... | 13 |
| e. Включване на компоненти в контролерите | 13 |
| 14. Описание на разработеното приложение..... | 14 |
| 15. Литература | 17 |

1. Структура на папката на CakePHP

След като изтеглите скелета на приложението CakePHP, трябва да видите няколко папки от най-високо ниво:

- Папката *bin* съдържа изпълнимите файлове на конзолата Cake.
- Конфигурационната папка съдържа конфигурационните *файлове* , които CakePHP използва. Тук трябва да се съхраняват подробности за връзката с базата данни, първоначалното зареждане, основните конфигурационни файлове и други.
- Папката *с добавки* е мястото, където се съхраняват добавките, които вашето приложение използва.

- Папката *с регистрационни* файлове обикновено съдържа вашите регистрационни файлове, в зависимост от конфигурацията на вашия журнал.
- Папката *src* ще бъде мястото, където ще бъдат поставени изходните файлове на вашето приложение.
- Папката *с шаблони* има презентационни файлове, поставени тук: елементи, страници с грешки, оформления и файлове с шаблони за преглед.
- Папката *с ресурси* има подпапка за различни типове файлове с ресурси. Подпапката *локали* съхранява езикови файлове за интернационализация.
- Папката *с тестове* ще бъде мястото, където поставяте тестовите случаи за вашето приложение.
- Папката *tmp* е мястото, където CakePHP съхранява временни данни. Действителните данни, които съхранява, зависят от това как сте конфигурирали CakePHP, но тази папка обикновено се използва за съхраняване на съобщения за превод, описания на модели и понякога информация за сесии.
- Папката *на доставчика* е мястото, където CakePHP и други зависимости на приложението ще бъдат инсталирани от Composer . Редактирането на тези файлове не се препоръчва, тъй като Composer ще презапише вашите промени следващия път, когато актуализирате.
- Webroot директорията е публичният корен на документа на вашето приложение . Той съдържа всички файлове, които искате да бъдат публично достъпни.

Уверете се, че папките *tmp* и *logs* съществуват и могат да се записват, в противен случай производителността на вашето приложение ще бъде сериозно засегната. В режим на отстраняване на грешки CakePHP ще ви предупреди, ако тези директории не могат да се записват.

2. Какво е CakePHP ?

CakePHP е Framework за уеб разработка използващ PHP 8.1. CakePHP е създаден за улесняване на често срещаните задачи свързани с разработката на уеб. CakePHP предоставя организационна структура, която съдържа, class names, filenames, database table names и други конвенции. Въпреки че конвенциите отнемат известно време за изучаване, като следват конвенциите, които CakePHP предоставя, можете да избегнете ненужна конфигурация и да направите еднаква структура на приложението, която улеснява работата с различни проекти. CakePHP използва добре познати концепции за софтуерно инженерство и шаблони за проектиране на софтуер , като например конвенция над конфигурация , модел–изглед–контролер ,

активен запис , картографиране на асоциирани данни и преден контролер. Разпространява се под лиценза на MIT(Масачузетския технологичен институт)

3. История

CakePHP стартира през април 2005 г., когато полският програмист Michał Tatarunowicz написа минимална версия на рамка за бързо разработване на приложения в PHP , като я нарече Cake. Той публикува рамката под лиценза на MIT и я отвори за онлайн общността на разработчиците. През декември 2005 г. L. Masters и GJ Woodworth основават Cake Software Foundation за насърчаване на развитието, свързано с CakePHP. Версия 1.0 беше пусната през май 2006 г. Едно от вдъхновенията на проекта беше [Ruby on Rails](#) , използвайки много от неговите концепции. Оттогава общността се разрасна и създаде няколко подпроекта. През октомври 2009 г. ръководителят на проекта Woodworth и разработчикът N. Abele се оттеглиха от проекта, за да се съсредоточат върху собствените си проекти, включително уеб рамката [Lithium](#) (преди това част от проекта CakePHP). Останалият екип за разработка продължи да се фокусира върху първоначалната пътна карта, която беше определена преди това.

4. The Model Layer(Моделен слой)

Моделният слой представлява частта от вашето приложение, която реализира бизнес логиката. Той отговаря за извличането на данни и преобразуването им в основните смислени концепции във вашето приложение. Това включва обработка, валидиране, асоцииране или други задачи, свързани с обработката на данни.

В случай на социална мрежа, моделният слой би се погрижил за задачи като запазване на потребителски данни, запазване на асоциации на приятели, съхраняване и извличане на потребителски снимки, намиране на предложения за нови приятели и т.н. Моделните обекти могат да се разглеждат като „Приятел“, „Потребител“, „Коментар“ или „Снимка“. Ако искаме да заредим някои данни от нашата таблица с потребители, бихме могли да направим:

```
use Cake\ORM\TableRegistry;

// Prior to 3.6 use TableRegistry::get('Users')
$users = TableRegistry::getTableLocator()->get('Users');
$query = $users->find();
foreach ($query as $row) {
    echo $row->username;
}
```

Фигура 1

Може да забележите, че не трябваше да пишем никакъв код, преди да можем да започнем да работим с нашите данни. Използвайки конвенции, CakePHP ще използва стандартни класове за класове на таблици и обекти, които все още не са дефинирани.

Ако искаме да вкараме нов потребител и да го запазим с валидации ще използваме:

```
use Cake\ORM\TableRegistry;

// Prior to 3.6 use TableRegistry::get('Users')
$users = TableRegistry::getTableLocator()->get('Users');
$user = $users->newEntity(['email' => 'mark@example.com']);
$users->save($user);
```

Фигура 2

5. The View Layer (Слоят за преглед)

Слоят View изобразява представяне на моделирани данни. Тъй като е отделен от обектите на модела, той е отговорен за използването на информацията, с която разполага, за създаване на всеки презентационен интерфейс, от който вашето приложение може да се нуждае.

Например, изгледът може да използва данни от модела, за да изобрази HTML шаблон за изглед, който го съдържа, или XML форматиран резултат, който другите да използват:

```
// In a view template file, we'll render an 'element' for each user.
<?php foreach ($users as $user): ?>
    <li class="user">
        <?= $this->element('user_info', ['user' => $user]) ?>
    </li>
<?php endforeach; ?>
```

Фигура 3

Слоят View предоставя редица точки за разширение като View Templates, Elements и View Cells, за да ви позволи да използвате повторно вашата логика за презентиране. Слоят View не е ограничен само до HTML или текстово представяне на данните. Може да се използва за доставяне на често срещани формати на данни като JSON, XML и чрез плъгируема архитектура всеки друг формат, от който може да се нуждаете, като CSV.

6. The Controller view (Слоят на контролера)

Слоят на контролера обработва заявки от потребители. Той отговаря за изобразяването на отговор с помощта както на слоя Model, така и на слоя View. Контролерът може да се разглежда като мениджър, който гарантира, че всички ресурси, необходими за изпълнение на задача, са делегирани на правилните работници. Той изчаква петиции от клиенти, проверява тяхната валидност според правилата за удостоверяване или оторизация, делегира извличане или обработка на данни към модела, избира вида на представянето на данните, които клиентите приемат, и накрая делегира процеса на изобразяване на слоя View. Пример за контролер за регистрация на потребители би бил:

```
public function add()
{
    $user = $this->Users->newEntity();
    if ($this->request->is('post')) {
        $user = $this->Users->patchEntity($user, $this->request->getData());
        if ($this->Users->save($user, ['validate' => 'registration'])) {
            $this->Flash->success(__('You are now registered.));
        } else {
            $this->Flash->error(__('There were some problems.));
        }
    }
    $this->set('user', $user);
}
```

Фигура 4

7. Инсталация на CakePHP

CakePHP има определени изисквания, за да може да бъде инсталиран.

- HTTP Server. For example: Apache. Having mod_rewrite is preferred, but by no means required. You can also use nginx, or Microsoft IIS if you prefer.
- Minimum PHP 5.6 (7.4 supported).
- mbstring PHP extension
- intl PHP extension
- simplexml PHP extension
- PDO PHP extension

CakePHP поддържа различни видове бази данни като :

MySQL, MariaDB, PostgreSQL, Microsoft SQL Server и SQLite3

Преди да започнете инсталацията, трябва да се уверите, че вашата версия на PHP е PHP 5.6 или по-висока. Версията на PHP на вашият сървър трябва също да бъде по PHP 5.6 или по-висока.

Инсталацията на CakePHP (за Windows) започва със свалянето на Composer(Композитор). Инсталаторът - който изисква вече да имате инсталиран PHP - ще изтегли Composer вместо вас и ще настрои вашата променлива на средата PATH, така че можете просто да извикате composer от всяка директория.

8. Command-line installation

За бърза инсталация на Composer в настоящата директория се използва следният код в терминала:

```
php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"
php -r "if (hash_file('sha384', 'composer-setup.php') ===
'55ce33d7678c5a611085589ff3dd8b3c52d662cd01d4ba75c0ee0459970c2200a51f492d557530c71c15d8dba01eae')
{ echo 'Installer verified'; } else { echo 'Installer corrupt'; unlink('composer-setup.php');
} echo PHP_EOL;"
php composer-setup.php
php -r "unlink('composer-setup.php');"
```

Фигура 5

Този инсталационен скрипт просто ще провери някои настройки на php.ini, ще ви предупреди, ако са зададени неправилно, и след това ще изтегли най-новия composer.phar в текущата директория.

Най-вероятно искате да поставите composer.phar в директория на вашия PATH, така че можете просто да извикате composer от която и да е директория:

```
sudo mv composer.phar /usr/local/bin/composer
```

Фигура 6

9. Създаване на CakePHP проект

За да се създаде нова CakePHP апликация е нужно използването на create-project команда в composer-a:

```
composer create-project --prefer-dist cakephp/app:"^3.10" my_app_name
```

Фигура 7

След като Composer завърши изтеглянето на скелета на приложението и основната библиотека CakePHP, трябва да имате работещо CakePHP приложение, инсталирано чрез Composer. Не забравяйте да запазите файловете composer.json и composer.lock с останалата част от сорс кода.

Вече можете да посетите зададения път от вас и да видите своята CakePHP апликация и базовата Номе страница.

10. Технологии и принципи, които използва CakePHP

Приложението е сърцето на вашето приложение. Той контролира как е конфигурирано вашето приложение и какви добавки, междинен софтуер, конзолни команди и маршрути са включени.

Можете да намерите своя клас `Application` на `src/Application.php`. По подразбиране ще бъдат доста малко и ще дефинира само няколко `Middleware` по подразбиране.

`bootstrap` Използва се за зареждане на конфигурационни файлове, дефиниране на константи и други глобални функции. По подразбиране това ще включва `config/bootstrap.php`. Това е идеалното място за зареждане на добавки и слушатели на глобални събития. Ако имате нужда от допълнителна конфигурация, трябва да ги добавите към файла `config/bootstrap.php` на вашето приложение. Този файл се включва преди всяка заявка и CLI команда.

Този файл е идеален за редица общи задачи за стартиране:

- Дефиниране на функции за удобство.
- Деклариране на константи.
- Дефиниране на конфигурация на кеша.
- Дефиниране на конфигурация за регистриране.
- Зареждане на персонализирани флексии.
- Зареждат се конфигурационни файлове.

В допълнение към файла `config/bootstrap.php`, който трябва да се използва за конфигуриране на проблеми на ниско ниво на вашето приложение, можете също да използвате метода за закачане `Application::bootstrap()` за зареждане/инициализиране на плъгини:

```
// in src/Application.php
namespace App;

use Cake\Core\Plugin;
use Cake\Http\BaseApplication;

class Application extends BaseApplication
{
    public function bootstrap()
    {
        // Call the parent to 'require_once' config/bootstrap.php
        parent::bootstrap();

        $this->addPlugin('MyPlugin', ['bootstrap' => true, 'routes' => true]);
    }
}
```

Фигура 8

11. Routing(Ротиране/Маршрутизиране)

Маршрутизирането ви предоставя инструменти, които съпоставят URL адресите с действията на контролера. Като дефинирате маршрути, можете да отделите начина, по който вашето приложение е внедрено от начина, по който са структурирани неговите URL адреси.

Маршрутизирането в CakePHP също така включва идеята за обратно маршрутизиране, където масив от параметри може да се трансформира в URL низ. С помощта на обратното маршрутизиране можете да промените URL структурата на вашето приложение, без да се налага да актуализирате целия си код.

В повечето случаи искаме да покажем нещо като целева страница, за това се добавя следния код към routes.php файла:

```
use Cake\Routing\RouteBuilder;
use Cake\Routing\Router;

// Using a scoped route builder.
Router::scope('/', function (RouteBuilder $routes) {
$routes->connect('/', ['controller' => 'Articles', 'action' => 'index']);
});

// Using the static method.
Router::connect('/', ['controller' => 'Articles', 'action' => 'index']);
```

Фигура 9

Router предоставя два интерфейса за свързване на маршрути Статичен и Динамичен. Статичният метод е обратно съвместим интерфейс, докато създателите с обхват предлагат по-кратък синтаксис при изграждане на множество маршрути и по-добра производителност.

Това ще изпълни метода на индексване в ArticlesController, когато бъде посетена началната страница на вашия сайт. Понякога имате нужда от динамични маршрути, които ще приемат множество параметри, това би бил случаят, например маршрут за преглед на съдържанието на статия:

```
$routes->connect('/articles/*', ['controller' => 'Articles', 'action' => 'view']);
```

Фигура 10

12. Работа с бази данни

Слоят за достъп до базата данни на CakePHP абстрахира и предоставя помощ за повечето аспекти на работа с релевантни бази данни, като поддръжка на връзки към сървър, изграждане на заявки, активиране на SQL инжекции, проверка и промяна на схеми и отстраняване на грешки и профилиране на заявки, изпратени до базата данни.

Най-лесният начин за създаване на връзка с база данни е използването на DSN низ:

```
use Cake\Datasource\ConnectionManager;

$dsn = 'mysql://root:password@localhost/my_database';
ConnectionManager::config('default', ['url' => $dsn]);
```

Фигура 11

Веднъж щом е създаден а можете да получите достъп до обекта чрез използването на :

```
$connection = ConnectionManager::get('default');
```

Фигура 12

Базите данни, които CakePHP поддържа са следните:

- MySQL 5.5+
- SQLite 3
- PostgreSQL 8.3+
- SQLServer 2008+
- Oracle (Чрез използването на плъгин)

13. Компоненти, които предоставя рамката

Компонентите са пакети от логика, които се споделят между контролерите. CakePHP идва с фантастичен набор от основни компоненти, които можете да използвате, за да помогнете при различни общи задачи. Можете също да създадете свои собствени компоненти. Ако установите, че искате да копирате и поставяте неща между контролерите, трябва да обмислите създаването на свой собствен компонент, който да съдържа функционалността. Създаването на компоненти поддържа кода на контролера чист и ви позволява да използвате повторно код между различни контролери.

а. Конфигуриране на компоненти (Configuring Components)

Много от основните компоненти изискват конфигурация. Някои примери за компоненти, изискващи конфигурация, са AuthComponent и Cookie.

Конфигурацията за тези компоненти и за компонентите като цяло обикновено се извършва чрез loadComponent() в метода initialize() на вашия контролер или чрез масива \$components:

```
class PostsController extends AppController
{
    public function initialize()
    {
        parent::initialize();
        $this->loadComponent('Auth', [
            'authorize' => 'Controller',
            'loginAction' => ['controller' => 'Users', 'action' => 'login']
        ]);
        $this->loadComponent('Cookie', ['expires' => '1 day']);
    }
}
```

Фигура 13

Също така можете да конфигурирате компоненти по време на изпълнение, като използвате метода config(). Често това се прави в метода beforeFilter() на вашия контролер

```
public function beforeFilter(Event $event)
{
    $this->Auth->config('authorize', ['controller']);
    $this->Auth->config('loginAction', ['controller' => 'Users', 'action' => 'login']);

    $this->Cookie->config('name', 'CookieMonster');
}
```

Фигура 14

Подобно на помощниците, компонентите изпълняват getConfig()и setConfig()методи за четене и запис на конфигурационни данни:

```
// Read config data.
$this->FormProtection->getConfig('unlockedActions');

// Set config
$this->Csrf->setConfig('cookieName', 'token');
```

Фигура 15

б. Псевдоним на компоненти (Aliasing Components)

Една обичайна настройка за използване е опцията `className`, която ви позволява да създавате псевдоними на компоненти. Тази функция е полезна, когато искате да замените `$this->Auth` или друга обща препратка към компонент с персонализирана реализация:

```
// src/Controller/PostsController.php
class PostsController extends AppController
{
    public function initialize()
    {
        $this->loadComponent('Auth', [
            'className' => 'MyAuth'
        ]);
    }
}

// src/Controller/Component/MyAuthComponent.php
use Cake\Controller\Component\AuthComponent;

class MyAuthComponent extends AuthComponent
{
    // Add your code to override the core AuthComponent
}
```

Фигура 16

с. Използване на компонентите

След като включите някои компоненти във вашия контролер, използването им е доста лесно. Всеки компонент, който използвате, е изложен като свойство на вашия контролер. Ако сте заредили `Cake\Controller\Component\FlashComponent` във вашия контролер, можете да получите достъп до него така:

```
class PostsController extends AppController
{
    public function initialize()
    {
        parent::initialize();
        $this->loadComponent('Flash');
    }

    public function delete()
    {
        if ($this->Post->delete($this->request->getData('Post.id')) {
            $this->Flash->success('Post deleted.');
```

Фигура 17

d. Създаване на компоненти

Да предположим, че нашето приложение трябва да извърши сложна математическа операция в много различни части на приложението. Можем да създадем компонент, който да съхранява тази споделена логика за използване в много различни контролери.

Първата стъпка е да създадете нов компонентен файл и клас. Създайте файла в `src/Controller/Component/MathComponent.php`. Основната структура на компонента ще изглежда така:

```
namespace App\Controller\Component;

use Cake\Controller\Component;

class MathComponent extends Component
{
    public function doComplexOperation($amount1, $amount2)
    {
        return $amount1 + $amount2;
    }
}
```

Фигура 18

Много е важно всички компоненти да разширява `Cake\Controller\Component`. Ако това не бъде направено ще се появяват грешки.

e. Включване на компоненти в контролерите

След като нашият компонент е готов, можем да го използваме в контролерите на приложението, като го заредим по време на метода `initialize()` на контролера. Веднъж зареден, на контролера ще бъде даден нов атрибут, наречен на компонента, чрез който можем да получим достъп до негов екземпляр:

```
public function initialize()
{
    parent::initialize();
    $this->loadComponent('Math');
    $this->loadComponent('Csrf');
}
```

Фигура 19

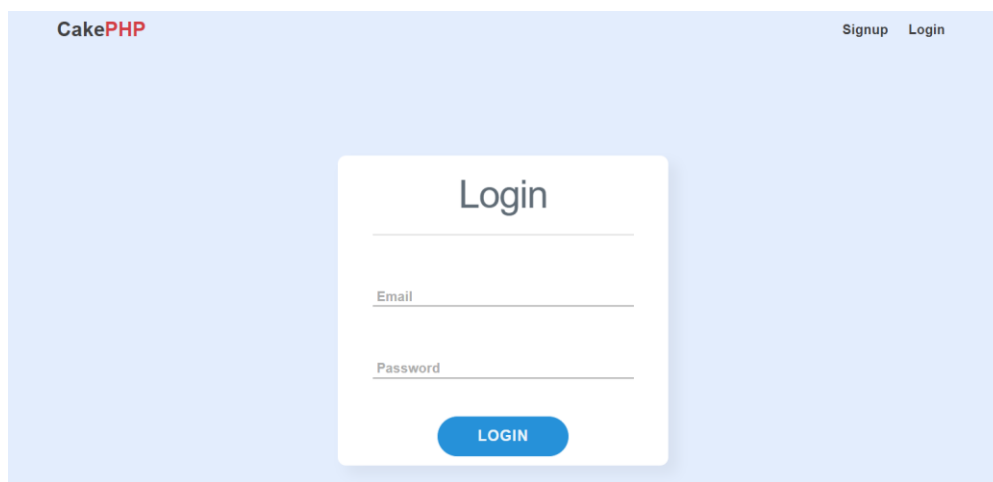
Когато включвате компоненти в контролер, можете също да декларирате набор от параметри, които ще бъдат предадени на конструктора на компонента. След това тези параметри могат да бъдат обработени от компонента чрез :

```
public function initialize()
{
    parent::initialize();
    $this->loadComponent('Math', [
        'precision' => 2,
        'randomGenerator' => 'srand'
    ]);
    $this->loadComponent('Csrf');
}
```

Фигура 20

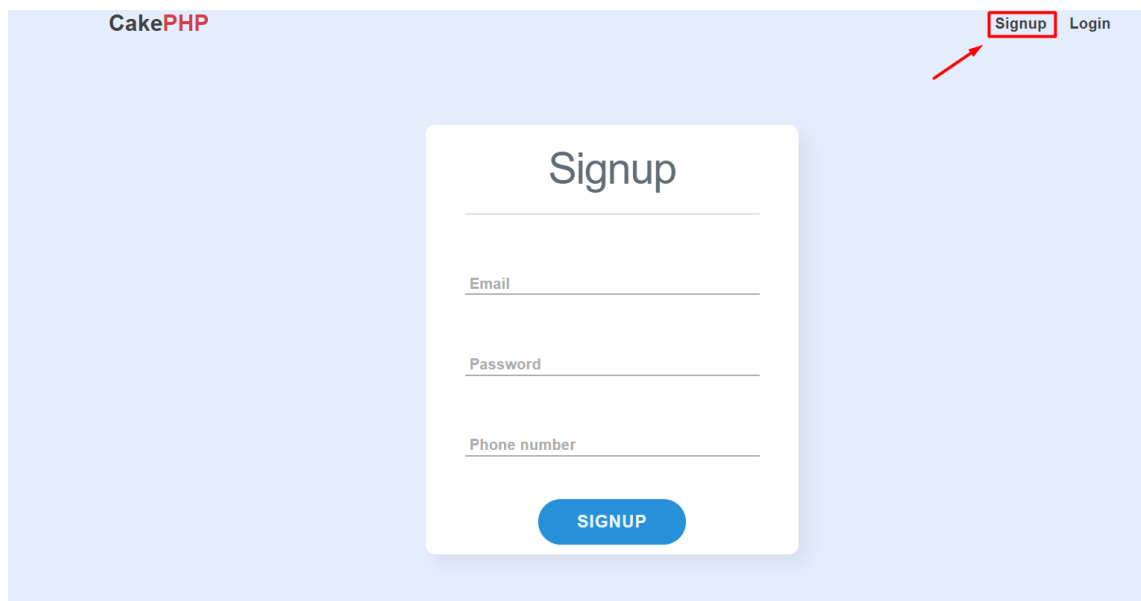
14. Описание на разработеното приложение

Когато влезем в приложението се появява **Login** форма, която служи за идентифициране на регистрираните потребители.



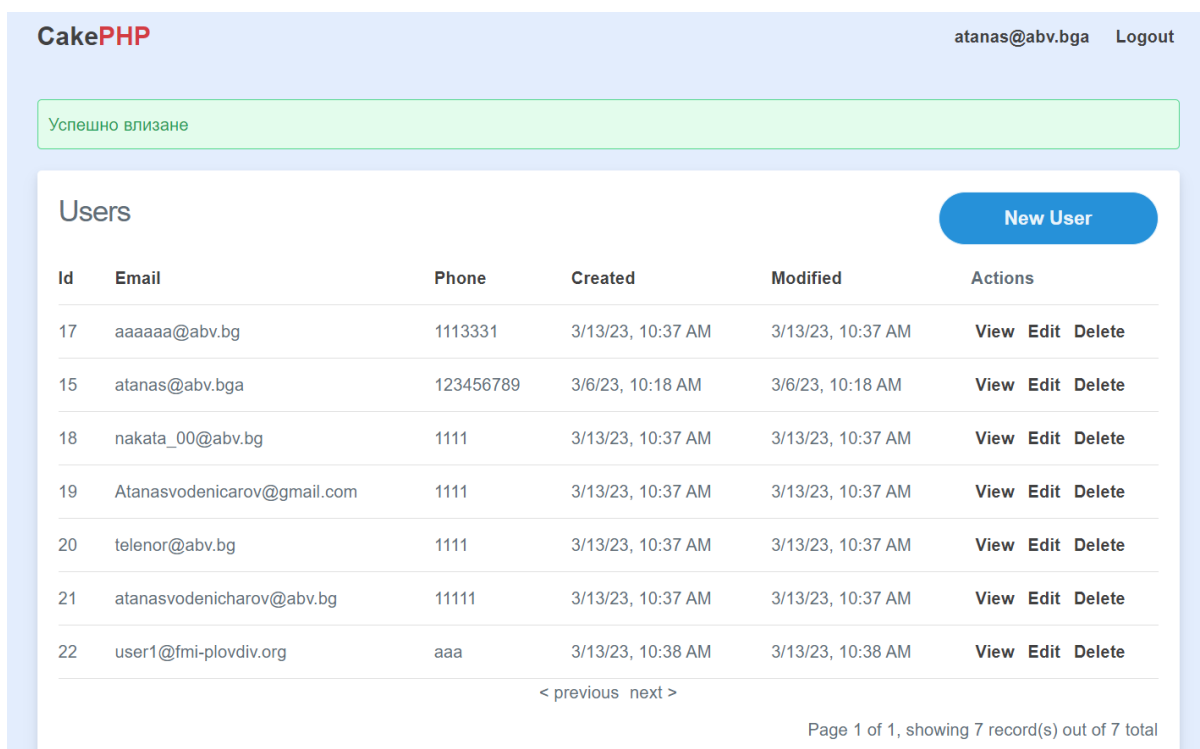
Фигура 21

За да се регистрира потребител се избира бутон “**Signup**” в горния десен ъгъл. Отваря се форма за регистрация, в която се попълва Имейл, парола и телефонен номер.



Фигура 22

Ако въведените данни от потребителя при логване са валидни, PHP скриптът обикновено задава променлива на сесията, за да поддържа състоянието на удостоверяване на потребителя и да пренасочва потребителя към подходяща страница или табло за управление. Потребителят е пренасочен към страницата за управление на потребители.



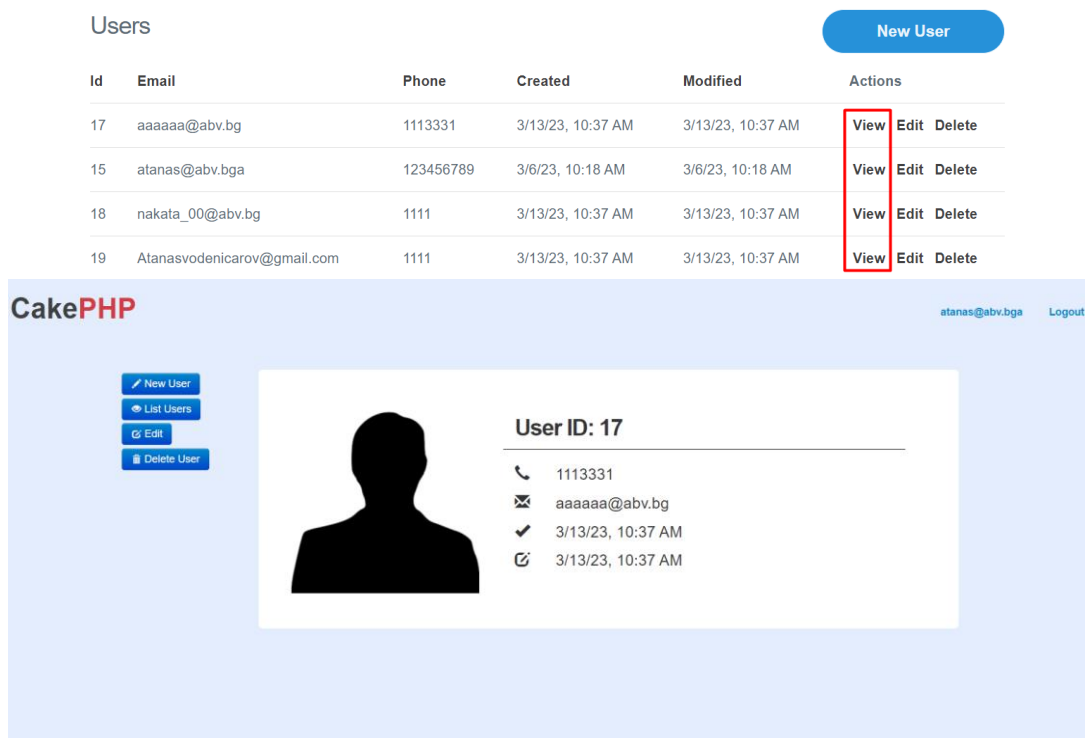
The screenshot shows the CakePHP Users management interface. At the top, there's a header with "CakePHP" and a user "atanas@abv.bga" with a "Logout" link. Below the header, a green message box says "Успешно влизане". The main content area is titled "Users" and features a "New User" button. A table lists 7 users with columns: Id, Email, Phone, Created, Modified, and Actions. The "Actions" column contains "View", "Edit", and "Delete" links for each user. At the bottom, there's a pagination link "< previous next >" and a message "Page 1 of 1, showing 7 record(s) out of 7 total".

| Id | Email | Phone | Created | Modified | Actions |
|----|-----------------------------|-----------|-------------------|-------------------|------------------|
| 17 | aaaaaa@abv.bg | 1113331 | 3/13/23, 10:37 AM | 3/13/23, 10:37 AM | View Edit Delete |
| 15 | atanas@abv.bga | 123456789 | 3/6/23, 10:18 AM | 3/6/23, 10:18 AM | View Edit Delete |
| 18 | nakata_00@abv.bg | 1111 | 3/13/23, 10:37 AM | 3/13/23, 10:37 AM | View Edit Delete |
| 19 | Atanasvodenicarov@gmail.com | 1111 | 3/13/23, 10:37 AM | 3/13/23, 10:37 AM | View Edit Delete |
| 20 | telenor@abv.bg | 1111 | 3/13/23, 10:37 AM | 3/13/23, 10:37 AM | View Edit Delete |
| 21 | atanasvodenicharov@abv.bg | 11111 | 3/13/23, 10:37 AM | 3/13/23, 10:37 AM | View Edit Delete |
| 22 | user1@fmi-plovdiv.org | aaa | 3/13/23, 10:38 AM | 3/13/23, 10:38 AM | View Edit Delete |

Фигура 23

В таблото за управление на регистрации потребителят може да:

- Преглежда регистрацията чрез кликане на бутона "View".



The screenshot shows the CakePHP Users management interface with the "View" action selected for user ID 17. The "View" link in the "Actions" column of the user list is highlighted with a red box. Below the list, there's a "New User" button. The main content area displays the user details for User ID: 17, including a silhouette icon, phone number (1113331), email (aaaaaa@abv.bg), and creation/modification dates (3/13/23, 10:37 AM). On the left side, there are buttons for "New User", "List Users", "Edit", and "Delete User".

Фигура 24

- Добавя нови потребители чрез кликване на бутона „New User”

Users

[New User](#)

| Id | Email | Phone | Created | Modified | Actions |
|----|------------------|-----------|-------------------|-------------------|--|
| 17 | aaaaaa@abv.bg | 1113331 | 3/13/23, 10:37 AM | 3/13/23, 10:37 AM | View Edit Delete |
| 15 | atanas@abv.bga | 123456789 | 3/6/23, 10:18 AM | 3/6/23, 10:18 AM | View Edit Delete |
| 18 | nakata_00@abv.bg | 1111 | 3/13/23, 10:37 AM | 3/13/23, 10:37 AM | View Edit Delete |

Фигура 25

- Редактира потребители чрез кликване на бутона “Edit”.

Users

[New User](#)

| Id | Email | Phone | Created | Modified | Actions |
|----|-----------------------------|-----------|-------------------|-------------------|--|
| 17 | aaaaaa@abv.bg | 1113331 | 3/13/23, 10:37 AM | 3/13/23, 10:37 AM | View Edit Delete |
| 15 | atanas@abv.bga | 123456789 | 3/6/23, 10:18 AM | 3/6/23, 10:18 AM | View Edit Delete |
| 18 | nakata_00@abv.bg | 1111 | 3/13/23, 10:37 AM | 3/13/23, 10:37 AM | View Edit Delete |
| 19 | Atanasvodenicarov@gmail.com | 1111 | 3/13/23, 10:37 AM | 3/13/23, 10:37 AM | View Edit Delete |

Фигура 26

- Да изтрива потребители чрез кликване на бутона „Delete”.

Users

[New User](#)

| Id | Email | Phone | Created | Modified | Actions |
|----|-----------------------------|-----------|-------------------|-------------------|--|
| 17 | aaaaaa@abv.bg | 1113331 | 3/13/23, 10:37 AM | 3/13/23, 10:37 AM | View Edit Delete |
| 15 | atanas@abv.bga | 123456789 | 3/6/23, 10:18 AM | 3/6/23, 10:18 AM | View Edit Delete |
| 18 | nakata_00@abv.bg | 1111 | 3/13/23, 10:37 AM | 3/13/23, 10:37 AM | View Edit Delete |
| 19 | Atanasvodenicarov@gmail.com | 1111 | 3/13/23, 10:37 AM | 3/13/23, 10:37 AM | View Edit Delete |
| 20 | telenor@abv.bg | 1111 | 3/13/23, 10:37 AM | 3/13/23, 10:37 AM | View Edit Delete |
| 21 | atanasvodenicharov@abv.bg | 11111 | 3/13/23, 10:37 AM | 3/13/23, 10:37 AM | View Edit Delete |
| 22 | user1@fmi-plovdiv.org | aaa | 3/13/23, 10:38 AM | 3/13/23, 10:38 AM | View Edit Delete |

< previous next >

Фигура 27

15. Литература

- <https://book.cakephp.org/3/en/installation.html>
- <https://getcomposer.org/download/>
- <https://getcomposer.org/doc/faqs/how-to-install-composer-programmatically.md>
- <https://book.cakephp.org/3/en/controllers/components.html>
- <https://book.cakephp.org/3/en/development/configuration.html#application-bootstrap>
- <https://book.cakephp.org/3/en/development/routing.html>
- <https://book.cakephp.org/3/en/orm/database-basics.html>
- <https://book.cakephp.org/3/en/controllers/components.html>
- https://en.wikipedia.org/wiki/MIT_License
- https://en.wikipedia.org/wiki/Ruby_on_Rails