

**Aplikacje Internetowe – Projekt**

**Proces wytworzenia prostej aplikacji webowej opartej o framework cakePHP.**

**Kois Krzysztof**

**117625**

Spis treści

[Wstęp 3](#_Toc453939941)

[Specyfikacja założeń 3](#_Toc453939942)

[Framework CakePHP 4](#_Toc453939943)

[CakePHP a model MVC 5](#_Toc453939944)

[Struktura katalogów w CakePHP 6](#_Toc453939945)

[Struktura katalogów MVC w CakePHP 7](#_Toc453939946)

[Narzędzia 8](#_Toc453939947)

[Narzędzia wymagane do realizacji aplikacji 8](#_Toc453939948)

[Wykorzystane narzędzia i technologie 8](#_Toc453939949)

[Konfiguracja i implementacja systemu 11](#_Toc453939950)

[Pieczenie Ciastek - wytworzenie podstawowego szkieletu aplikacji 15](#_Toc453939951)

[Poszerzanie funkcjonalności aplikacji 17](#_Toc453939952)

[Logowanie 17](#_Toc453939953)

[Wylogowanie 19](#_Toc453939954)

[Podsumowanie 20](#_Toc453939955)

# Wstęp

Celem projektu jest przedstawienie, jak od podstaw stworzyć podstawową aplikację webową, opartą w znacznym stopniu o framework CakePHP. Aplikacja będzie opierać się na wzorcu projektowym MVC oraz korzystać będzie dodatkowo z lokalnego serwera bazy danych.

# Specyfikacja założeń

Założenia jakie będzie posiadać wytworzona aplikacja to:

* Wykorzystanie modelu MVC
* Autoryzacja użytkowników
* Logowanie/wylogowanie użytkowników
* Haszowanie haseł
* Graficzna prezentacja tabel z bazy danych
* Możliwość wyświetlania, dodawania, edycji i usuwania elementów z bazy danych z poziomu aplikacji

# Framework CakePHP



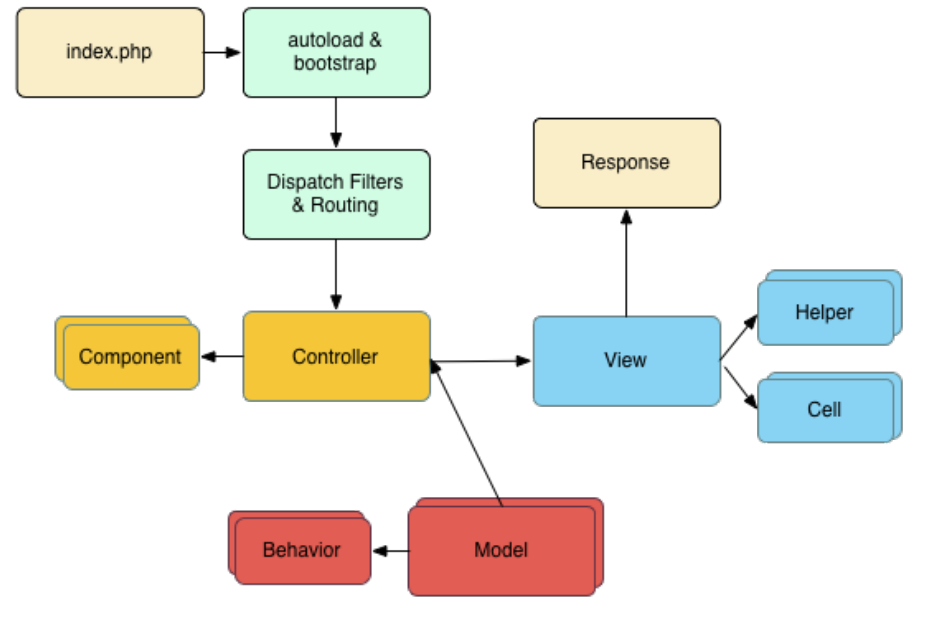
Rysunek 1.Logo frameworku CakePHP

CakePHP jest to darmowy open-sourcowy framework PHP, którego zadaniem jest stworzenie struktury do tworzenia aplikacji internetowych. Celem producentów jest umożliwienie użytkownikowi pracę w sposób szybki i uporządkowany.

Podstawowe zalety frameworku to:

* Generowanie podstawowego szablonu modelu MVC
* Elementy niezbędne do budowy aplikacji są wbudowane i nie trzeba samemu pisać żmudnych elementów aplikacji (mnogość zaimplementowanych rozwiązań)
* CookBooka – ogólnie dostępna dokumentacja zawierająca ponad 600 stron opisu oraz przykładów działania frameworka
* Posiada generator kodów, co pozwala szybko budować pliki składowe aplikacji
* Pozwala za pomocą bazy danych wygenerować podstawową funkcjonalność aplikacji
* Stosuje dobre zwyczaje jak konwencje nazw czy widoku
* Jest na bieżąco aktualizowana i posiada spore zaplecze osób wspierających jej rozwój
* Jest rozpowszechniana na licencji MIT – daje nieograniczone prawa do frameworku

## CakePHP a model MVC



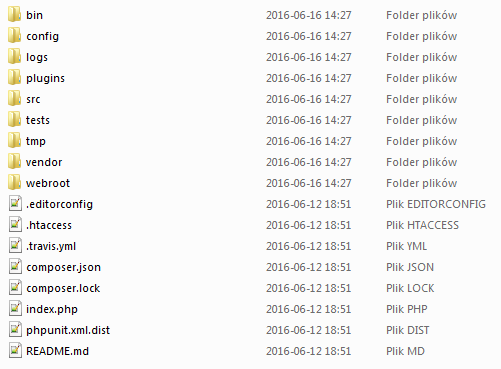
Rysunek 2. Budowa zapytania w CakePHP

Analizując powyższy rysunek, przedstawiając cykl obsługi żądania w Cake’u. Typowy cykl zapytania rozpoczyna się w momencie, gdy użytkownik żąda dostępu do strony bądź zasobu w aplikacji. Patrząc z wysoko poziomowej perspektywy programowania, każde żądanie przechodzić będzie przez następujące kroki:

1. Serwer sieciowy przekierowuje żądanie bezpośrednio do weebroot/index.php
2. Auto wyzwalacze aplikacji oraz pliki frameworku bootstrap zostają wczytane
3. Jeśli ustawiono jakieś filtry wysyłkowe, mogą one przechwycić żądanie i ewentualnie wygenerować odpowiedź
4. Filtr wybiera odpowiedni kontroler i akcję według zasad routingu
5. Zostaje wywołana akcja kontrolera, kontroler współgra z odpowiednim modelem i komponentem
6. Kontroler oddelegowuje żądanie do widoku, gdzie na podstawie modelu danych generowany jest wynik wyjściowy
7. Widok korzysta z pomocnika i ogniw, aby wygenerować ciało i nagłówek odpowiedzi
8. Odpowiedź jest odsyłana do klienta

## Struktura katalogów w CakePHP

Model MVC wymaga zastosowania odpowiedniej struktury katalogów, choć zasada działania jest jedna, to używa się różnych konwencji katalogowania plików. Omawiany framework ma własną strukturę plików, przedstawioną poniżej.

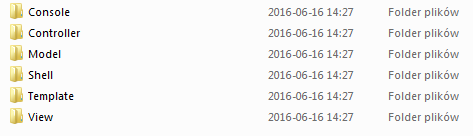


Rysunek 3. Czysta struktura plików CakePHP

Najważniejszymi katalogami są:

* Bin – zawiera pliki niezbędne do wygenerowania podstawowej funkcjonalności tj. plików kontrolera, modelu i widoku. Wymaga do tego tylko dostępu do bazy danych
* Config – zawiera ustawienia aplikacji m. in. ustawienia połączenia z bazą danych, ustawienia bezpieczeństwa jak soil seed, czy domyślnego kontrolera
* Vendor – w tym katalogu umieszczone są klasy i biblioteki spoza macierzystego frameworka, z których można skorzystać w projekcie, nie należy ich edytować, gdyż jest to serce frameworku
* Src – główny katalog plików, w którym przechowywany jest model mvc
* Weebroot – katalog, do którego są przekierowywane zapytania o główna stronę aplikacji.
* Tmp – przechowuje tymczasowe pliki, z których korzysta aplikacja. Zazwyczaj są to informacje dotyczące sesji

## Struktura katalogów MVC w CakePHP



Rysunek 4. Czysta struktura plików w folderze src – CakePHP

Składa się z następujących katalogów:

* Console – przetrzymuje polecenia i zadania konsoli aplikacji. Stworzone do przechowywania zadań Cron’a oraz bash’owych.
* Controller – znajdują się w nim kontrolery i ich komponenty
* Model – zawiera schematy budowy tabel bazy danych połączonej z aplikacją, a także zachowania i typy zmiennych
* Shell – zawiera ewentualną konsolę aplikacji
* Template – zawiera pliki wyglądu każdej z podstrony: elementy, strony błędów, style
* View – zawiera pliki ustawiające ogólny wygląd strony

# Narzędzia

Przed przystąpieniem do realizacji aplikacji internetowej należało zastanowić się nad doborem odpowiednich narzędzi, które pomogą w tworzeniu bazy danych oraz aplikacji, a także języków przy których pomocy aplikacja zostanie stworzona. Narzędzia były dobierane pod względem:

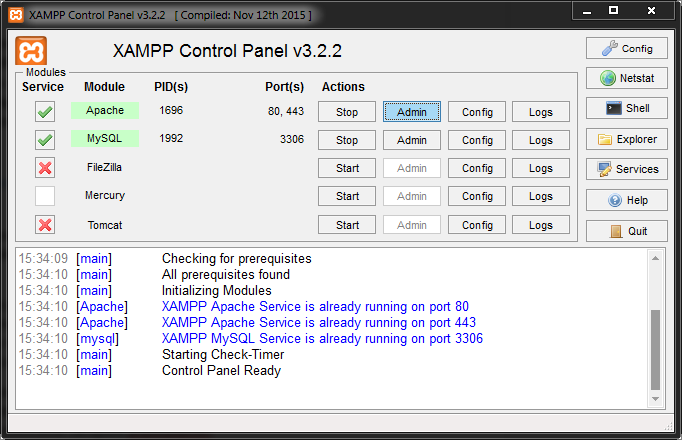
* Kompatybilności z systemem operacyjnym Windows
* Programy muszą być rozpowszechniane na zasadach darmowej licencji
* Narzędzia muszą być ze sobą zgodne i obsługiwać te same języki programowania

## Narzędzia wymagane do realizacji aplikacji

* Serwer http
* Serwer bazy danych wraz z systemem jego obsługi
* Aplikacja do edycji kodu źródłowego
* Aplikacja do obsługi systemu
* Język/-i do wykonania interfejsu użytkownika
* Język/-i do wykonania logicznej warstwy aplikacji

## Wykorzystane narzędzia i technologie

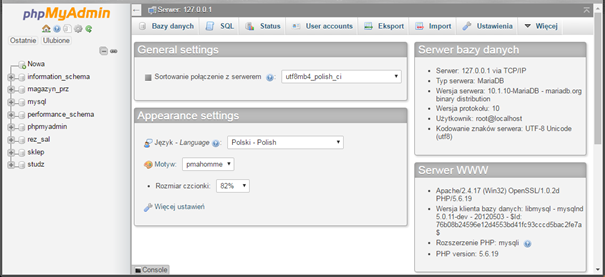
Do obsługi oraz stworzenia systemu działającego jako aplikacja internetowa użyto gotowego wieloplatformowego pakietu aplikacji zwanego **XAMPP 3.2.2** (X-cross-platform Apache + MariaDB + PHP + Perl ).



Rysunek 5. Panel kontrolny XAMPP

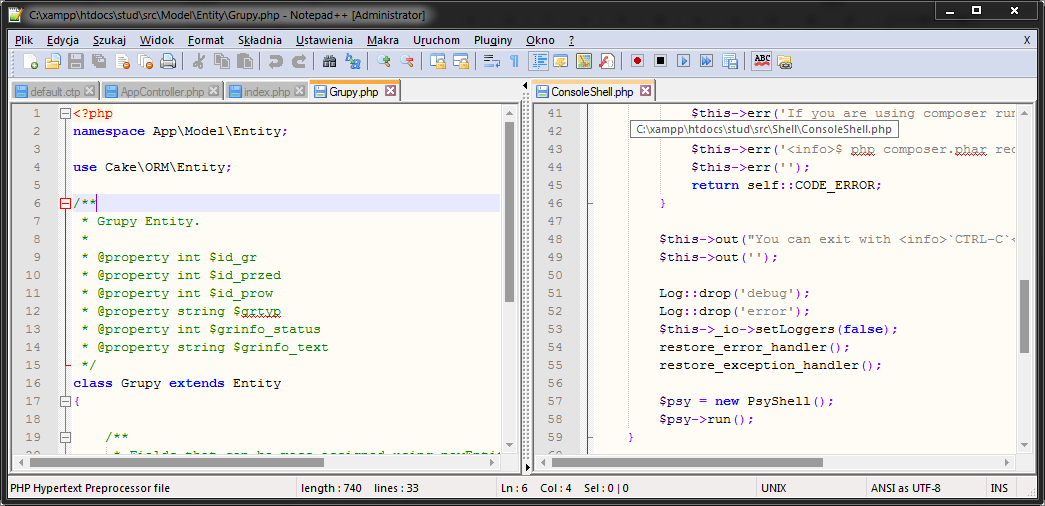
**Serwer Apache 2.4.17N**- Apache jest to wieloplatformowy serwer WWW utworzony przez firmę Apache Foundation. Jest dostępny na wielu systemach operacyjnych (Linux, Windows, OS X).Apache jest to wieloplatformowy serwer WWW utworzony przez firmę Apache Foundation. Jest dostępny na wielu systemach operacyjnych (Linux, Windows, OS X).

**MySQL -** Jest to system służący do zarządzania relacyjnymi bazami danych rozwijany przez firmy MySQL AB, następnie Sun Microsystems, a obecnie Oracle. Oparty jest na darmowej licencji GNU. Do obsługi MySQL zostało stworzone narzędzie phpMyAdmin.



Rysunek 6. Okno główne phpMyAdmin

**Notepad++** - jest programem opartym na wieloplatformowym komponencie do budowy edytorów tekstu Scintilla. Główna zaletą programu jest podświetlanie składni kilkudziesięciu różnych języków programistycznych. Notepad++ pozwala na samodzielne opracowanie własnego systemu kolorowania dzięki wbudowanemu modułowi, co przyda się przy pisaniu w phpCake, gdyż rozszerzenie tego frameworku (.ctp) nie jest rozpoznawane zazwyczaj przez środowiska programistyczne bez doinstalowania dodatków.



Rysunek 7. Zrzut okna Notepad++

**Przeglądarka internetowa -** Pliki źródłowe aplikacji są plikami stron www, więc obsługa odbywa się poprzez dowolną przeglądarkę internetową, obsługująca zastosowane języki.

**HTML5 (HyperText Markup Language)** - HTML5 jest wykorzystywany do tworzenia stron internetowych WWW. Jest to język znaczników i został stworzony przez firmę W3C oraz HATWG. Język ten został wykorzystany do stworzenia interfejsu graficznego, ponieważ jest on najbardziej popularnym językiem oraz kompatybilnym ze wszystkimi przeglądarkami.

**CSS3.0 (Cascading Style Sheets) -** CSS jest to język służący do opisu wyświetlania stron WWW. Stanowi listę reguł ustalająca sposób wyświetlania przez przeglądarkę internetową zawartości określonych elementów. Został stworzony przez organizację W3C. Język ten został użyty podczas tworzenia aplikacji ponieważ współpracuje z HTML5, umożliwia lepszą organizację wyglądu oraz ewentualne zmiany wprowadzane do aplikacji muszą zostać wprowadzone tylko w jednym pliku. Bez użycia kaskadowego arkusza stylów definiowanie wyglądu odbywało by się poprzez powtarzania definicji wyglądu przy każdym elemencie.

**SQL (Structured Query Language) -** Strukturalny język zapytań służący do tworzenia oraz modyfikowania baz danych. Jest oficjalnym standardem wspieranym przez Międzynarodową Organizację Normalizacyjną (ISO). Jest on wykorzystywany wyłącznie do komunikacji z bazą danych i nie pozwala na stworzenie kompletnego programu. Został wybrany ze względu na współpracę z językami PHP oraz MySQL.

**PHP 5.6.19 -** Jest obiektowym językiem programowania, który służy do generowania stron internetowych oraz budowania aplikacji w czasie rzeczywistym. Stosowany jest głównie do tworzenia skryptów po stronie serwera. Został wybrany ponieważ współpracuje z pozostałymi językami wybranymi do tworzenia aplikacji.

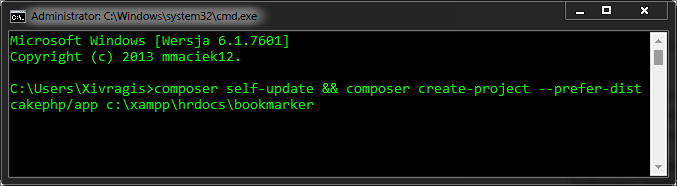
**Frameworki** – są to nakładki na dany język programowania udostępniające w uproszczonej formie gotowe funkcjonalności. Wykorzystano farmerki takie, jak **Bootstrap 3.3.6**, **CakePHP 3.2.10**

Dodatkowo niezbędny w procesie wytwarzania projektu był: **Composer 1.1.2**- narzędzie to pozwala zarządzać zależnościami między bibliotekami na poziomie projektu w sposób łatwy i przejrzysty. Dzięki niemu CakePHP będzie mógł stworzyć swoją strukturę folderów według modelu mvc.

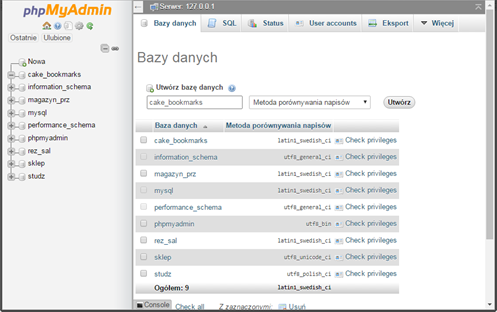
# Konfiguracja i implementacja systemu

Aby aplikacja mogła być rozpoczęta, należy ustawić odpowiednio nasze oprogramowanie. Pominiemy tu instalowanie domyślnych ustawień instalacji, a skupiono na opisie elementów, które należy wprowadzić „ręcznie”.

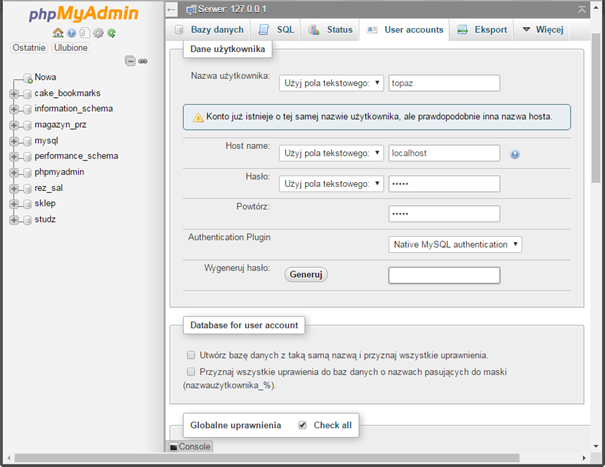
1. W pierwszej kolejności należy określić lokalizację dla aplikacji. Jako, że aplikacja będzie ona hostowana lokalnie oraz, że korzystamy z XAMPP’a, to domyślnie utworzymy folder w folderze xampp\htdocs\nazwaaplikacji



1. Kolejnym etapem jest włączenie możliwości nadpisywania ścieżek URI serwera Apache. W tym celu z panelu administracyjnego XAMPP’a wybieramy przycisk Config przypadający na serwer Apache. Następnie wybierzemy element httpd.conf i znajdziemy w nim wartość „LoadModule rewrite\_module modules/mod\_rewrite.so” w tym miejscu należy usunąć z początku linii znak „;”
2. Następnie należy utworzyć bazy danych, używając phpMyAdmin tworzymy bazę o nazwie „cake\_bookmarks”.



1. Teraz należy umożliwić przyszłej aplikacji dostęp do utworzonej bazy. Tworzymy w tym celu nowe konto użytkownika o parametrach podanych na poniższym obrazku. Hasło w tym przypadku jest takie samo jak nazwa użytkownika. **Nie wolno zapomnieć o zaznaczeniu checkboxa Check all przy ustawieniu Globalne uprawnienia.**



1. Teraz łączymy aplikację z bazą danych poprzez edycję ustawień frameworku. Domyślnie pod adresem c:\xampp\htdocs\bookmarker\config\app.php musimy nawiązać połączenie aplikacji i bazy danych. W tym celu wprowadzamy poniższe zmiany w sekcji **Datasources**.

'default' => [

'host' => 'localhost',

'port' => '3306',

'username' => 'topaz',

'password' => 'topaz',

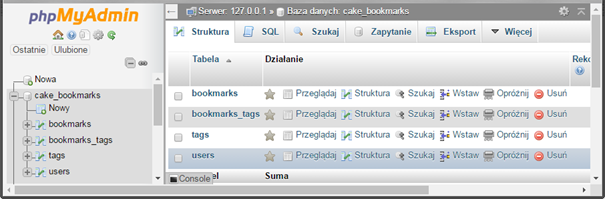
'database' => 'studz',

'encoding' => 'utf8',

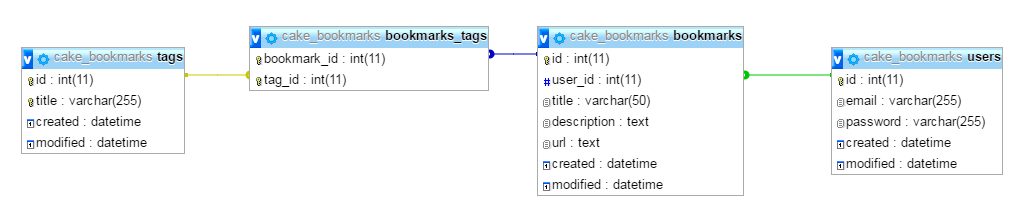
1. Po ustawieniu połączenia należy już tylko uzupełnić bazę danych o odpowiednie tabele. W tym celu w phpMyAdmin po wybraniu bazy cake\_bookmarks, przechodzimy do zakładki sql, wklejamy poniższy kod i wykonujemy go.

CREATE TABLE users (  
id INT AUTO\_INCREMENT PRIMARY KEY,  
email VARCHAR(255) **NOT NULL**,  
password VARCHAR(255) **NOT NULL**,  
created DATETIME,  
modified DATETIME  
);  
CREATE TABLE bookmarks (  
id INT AUTO\_INCREMENT PRIMARY KEY,  
user\_id INT **NOT NULL**,  
title VARCHAR(50),  
description TEXT,  
url TEXT,  
created DATETIME,  
modified DATETIME,  
FOREIGN KEY user\_key (user\_id) REFERENCES users(id)  
);  
CREATE TABLE tags (  
id INT AUTO\_INCREMENT PRIMARY KEY,  
title VARCHAR(255),  
created DATETIME,  
modified DATETIME,  
UNIQUE KEY (title)  
);  
CREATE TABLE bookmarks\_tags (  
bookmark\_id INT **NOT NULL**,  
tag\_id INT **NOT NULL**,  
PRIMARY KEY (bookmark\_id, tag\_id),  
FOREIGN KEY tag\_key(tag\_id) REFERENCES tags(id),  
FOREIGN KEY bookmark\_key(bookmark\_id) REFERENCES bookmarks(id)  
);

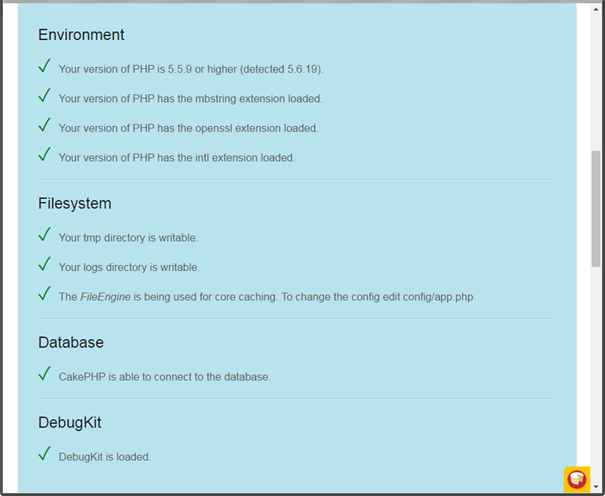
1. Po pomyślnym i bezbłędnym przetworzeniu żądania, baza danych powinna wyglądać następująco.



1. Wchodząc w phpMyAdmin w zakładkę naszej bazy, następnie z górnego menu wybierając opcję widok projektu, możemy ujrzeć relacje naszej bazy.

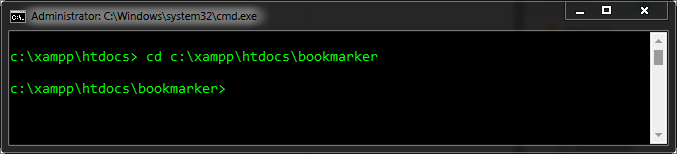


1. Wszystko zostało już ustawione jak powinno, można więc wejść przez przeglądarkę do naszego szablonu aplikacji, w celu upewnienia się, że aplikacja działa, jak należy. W przeglądarce jako adres podajemy **localhost/bookmarker**, po czym powinna się nam ukazać strona domyślna, którą na ten moment jest strona debugu. Nas interesują informacje widoczne po przeskrolowaniu okna niżej, jak na obrazku. Jeśli wszystkie parametry są oznaczone pozytywnie to aplikacja jest dobrze skonfigurowana i można przystąpić do kolejnego etapu.



# Pieczenie Ciastek - wytworzenie podstawowego szkieletu aplikacji

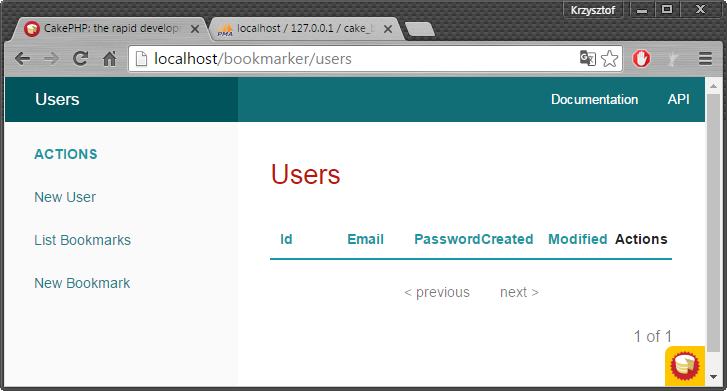
Nasz szkielet aplikacji będzie się składał z podstawowych kontrolerów, modeli i widoków niezbędnych do dodawania, usuwania, edycji i wyświetlenia informacji pochodzących z dołączonej bazy danych. Do utworzenia takiego szablonu będzie nam potrzebne konsola wbudowana w projekt. Aby się do niej dostać należy z poziomu konsoli poleceń dostać się do lokalizacji naszego projektu.



Konsola znajduje się w folderze bin, nazywa się cake, natomiast polecenie, którego użyjemy do stworzenia szablonu to bake, po opatrzeniu go w stosowne atrybuty. Aby upiec wszystkie szablony należy kolejno wpisywać polecenie bake dla każdej z tabel w bazie danych. W naszym przypadku będą to następujące polecenia:

bin\cake bake all users  
bin\cake bake all bookmarks  
bin\cake bake all tags

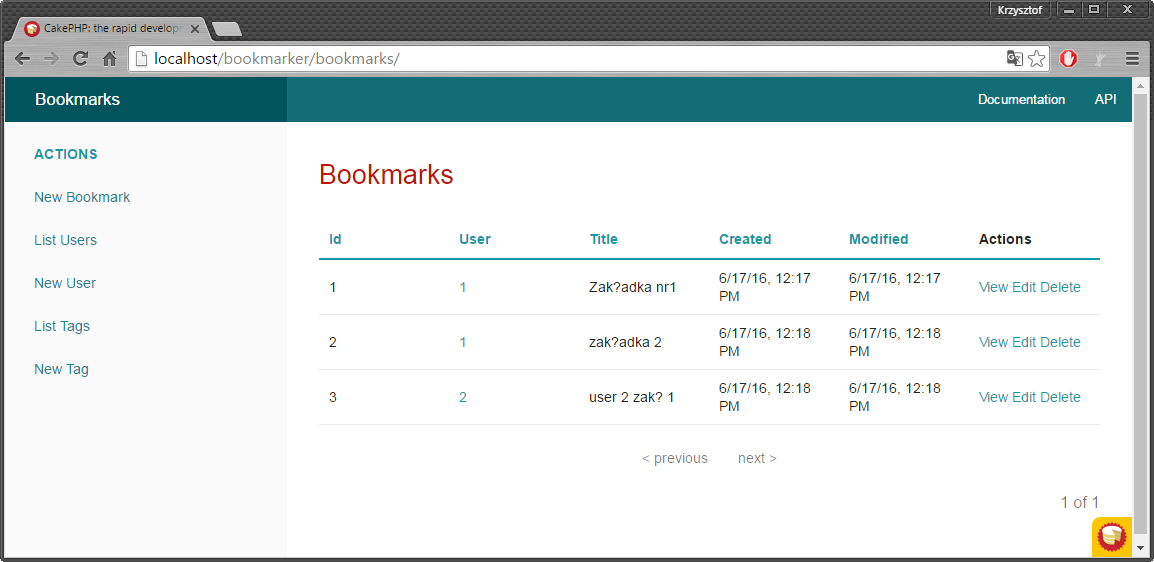
Po „upieczeniu” szkieletu można zobaczyć rezultat działania konsoli poprzez wejście w przeglądarce na stronę naszej aplikacji i dodanie do adresu nazwy kontrolera. Są już dostępne też podstawowe funkcjonalności dostępne w lewym panelu aplikacji.



Teraz należy dodać do bazy jakąś zawartość, aby można było coś wyświetlać. Przed tym jednak mając na uwadze bezpieczeństwo przyszłych użytkowników, dodamy haszowanie haseł zapisywanych w bazie danych. Będą one wtedy zapisywane w postaci ciągu znaków nie reprezentujących na pierwszy rzut oka żadnej spójnej wartości. Należy przy okazji wspomnieć o metodzie przechowywania metod. Metody, które operują na zbiorze elementów są umieszczane w klasie Table np. UsersTable. Z kolei metody operujące na pojedynczym rekordzie tabeli, są przechowywane w klasie Entity. Haszowanie polega na działaniu na jednym elemencie, więc musimy dodać pewne elementy do pliku **src/Model/Entity/User.php**, by kod wyglądał jak poniższy

**namespace** App\Model\Entity;  
**use** Cake\Auth\DefaultPasswordHasher;  
**use** Cake\ORM\Entity;  
**class User extends** Entity  
{  
*// Istniejący kod.*  
**protected function** \_setPassword($value)  
{  
$hasher = **new** DefaultPasswordHasher();  
**return** $hasher->hash($value);  
}  
}

Można już przystąpić do dodawania kilku elementów do bazy korzystając z naszej aplikacji.



# Poszerzanie funkcjonalności aplikacji

Gdy już mamy gotowy i funkcjonujący szablon, pora pokusić się o dodanie kolejnych podstawowych elementów. W ramach tego projektu rozszerzymy aplikacje o możliwości logowania i wylogowania.

## Logowanie

Do odpowiednich plików podanych w pierwszych linijkach ramek, dodajemy/modyfikujemy zawarte w nich kody.

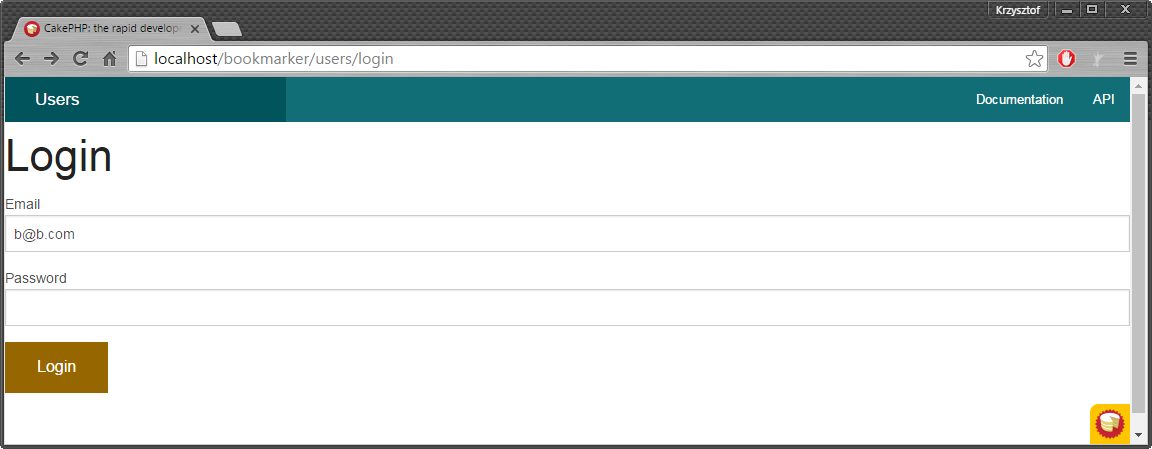
*//src/Controller/AppController.php*  
**namespace** App\Controller;  
**use** Cake\Controller\Controller;  
**class AppController extends** Controller  
{  
**public function** initialize()  
{  
$this->loadComponent('Flash');  
$this->loadComponent('Auth', [  
'authenticate' => [  
'Form' => [  
'fields' => [  
'username' => 'email',  
'password' => 'password'  
]  
]  
],  
'loginAction' => [  
'controller' => 'Users',  
'action' => 'login'  
]  
]);

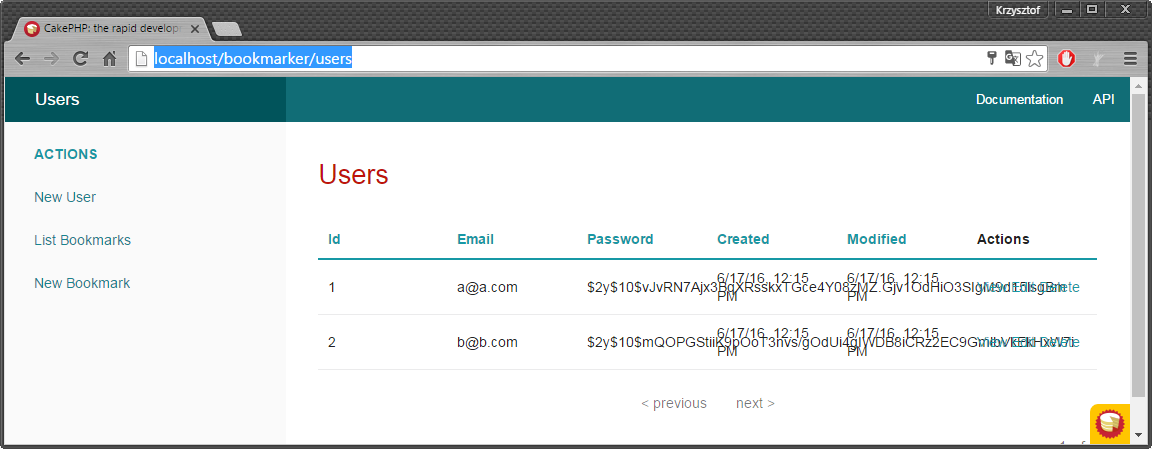
$this->Auth->allow(['display']);  
}  
}

*//src/Controller/UsersController.php*  
**public function** login()  
{  
**if** ($this->request->is('post')) {  
$user = $this->Auth->identify();  
**if** ($user) {  
$this->Auth->setUser($user);  
**return** $this->redirect($this->Auth->redirectUrl());  
}  
$this->Flash->error('Your username or password is incorrect.');  
}  
}

*//src/Template/Users/login.ctp*  
<h1>Login</h1>  
*<?*= $this->Form->create() *?>  
<?*= $this->Form->input('email') *?>*  
*<?*= $this->Form->input('password') *?>*  
*<?*= $this->Form->button('Login') *?>*  
*<?*= $this->Form->end() *?>*

Po odpowiedniej modyfikacji, możliwe powinno być zalogowanie się jednego z użytkowników. Dokonać tego można z każdej istniejącej strony aplikacji poza główną, gdyż poprzez włączenie autentykacji, każda strona będzie wymagała zalogowania przed wyświetleniem.



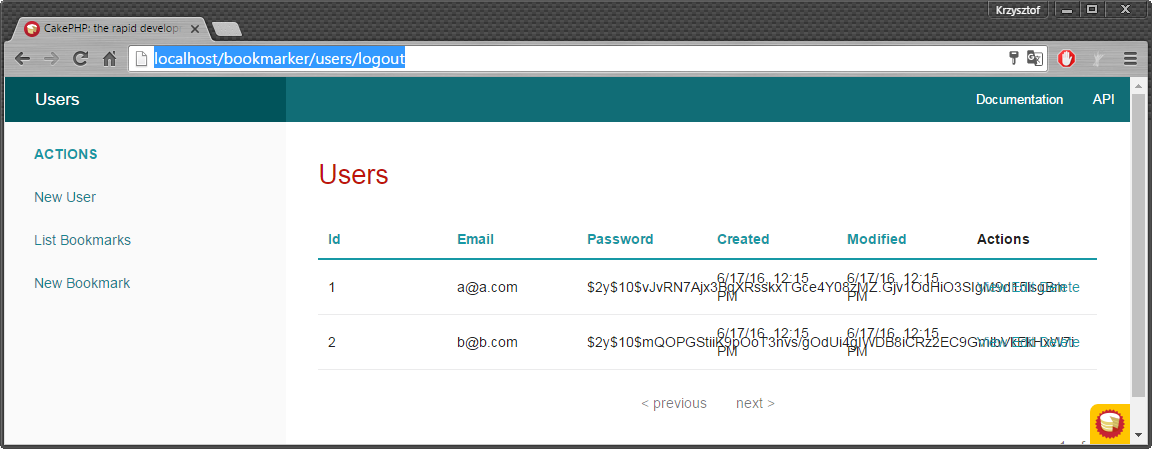


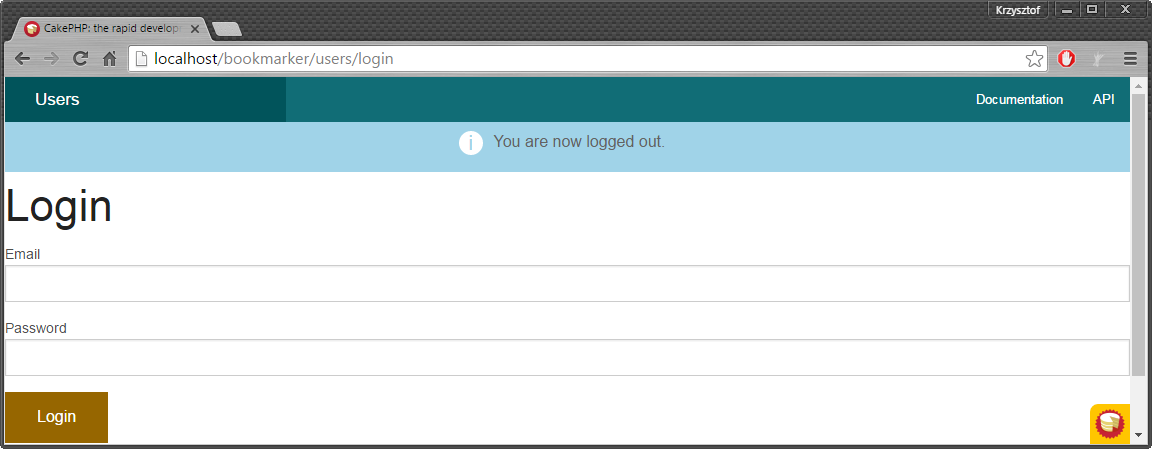
## Wylogowanie

Skoro można się już zalogować to wypadało by dać możliwość wylogowania, postępujemy tu jak poprzednio dodając odpowiedni kod do odpowiedniego pliku.

*// src/Controller/UsersController.php*   
**public function** initialize()  
{  
**parent**::initialize();  
$this->Auth->allow(['logout']);  
}  
**public function** logout()  
{  
$this->Flash->success('You are now logged out.');  
**return** $this->redirect($this->Auth->logout());  
}

Poprzez odwiedzenie strony users/logout, można dokonać wylogowania, co spowoduje odesłanie użytkownika do ponownego zalogowania się.





# Podsumowanie

CakePHP jest bardzo dużym i złożonym frameworkiem, jego opanowanie zajmuje trochę czasu, jednak jest on bardzo przydatny i potrafi znacząco usprawnić pisanie aplikacji, szczególnie jeśli chodzi o wsparcie początkowego etapu tworzenia szkieletu, przez co znacząco wpływa na skrócenie procesu wytwarzania. Posiada własne zasady nazewnictwa ( bez dodatkowej konfiguracji tylko w języku angielskim), konwencje modelu mvc, akceptuje kod pisany ręcznie bez użycia wbudowanych elementów, można go dowolnie edytować i używać dzięki licencji MIT. Z punktu widzenia niedoświadczonej osoby, jest on dobrze udokumentowany, dzięki sporemu zapleczu „how to” w postaci dużego społeczeństwa ludzi związanych z tym frameworkiem i pokaźnego manuala.

Szczególnie przy tworzeniu pierwszej aplikacji należy uważać na zaimplementowana konwencje nazw bazy danych. Jeśli jej nie zastosujemy, bądź nie umiejętnie ominiemy, otrzymamy masę błędów i niespójny kod z punktu widzenia laika CakePHP. Dzieje się tak, gdyż framework wyznaje zasadę COC ( Convention over configuration) co oznacza, że nazewnictwo jest ważniejsze od ustawień. Z racji, że spotkałem się z takim problemem w innym projekcie pozwolę sobie przybliżyć tu te zasady:

* Stosowanie angielskich nazw w bazie danych (użytkownicy -> users)
* Nazwy tabel, kontrolerów powinny być mnogimi odpowiednikami ich zawartości tj(user->users)
* Adresy URL są zawsze pisane małymi literami
* Nazwy tabel powinny być pisane w konwencji CamelCase tj. łączne bez znaku, kolejne wyrażenia rozpoczęte z Dużej litery poza pierwszym (BoxControler -> boxControler
* Pola z liczbą słów większą, bądź równą dwa maja używać twardej spacji (first\_name)
* Klucze obce w relacyjnych bazach danych są rozpoznawane domyślnie na zasadzie dodania do ich nazwy ciągu **„\_id” (co może skutecznie utrudnić pracę z frameworkiem przy przeoczeniu)**
* Łączenie tabel wykonywane jest na zasadzie podania nazwy dwóch tabeli oddzielonych twardą spacją w kolejności alfabetycznej (dogs\_cats->cats\_dogs)