JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 6 (2018) 6-11
Wysłane: 2017-09-11
Przyjęte: 2017-09-14

Porównanie wytwarzania aplikacji webowych z użyciem języka PHP oraz platformy Magento

Bartosz Drawdzik*, Maria Skublewska-Paszkowska

Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Artykuł przedstawia proces tworzenia dwóch aplikacji w oparciu o dwie różne metody programowania, jedna z nich to aplikacja PHP oparta na danych przechowywanych w relacyjnej bazie danych MySQL oraz aplikacja o analogicznej funkcjonalności zaimplementowana na platformie Magento 1.9. Głównym celem artykułu jest przetestowanie obu aplikacji oraz weryfikacja w jakim stopniu poszczególne funkcjonalności są odporne na błędy pojawiające się podczas korzystania z aplikacji. Uzyskane rezultaty pozwolą ocenić sposób programowania poprawiający czystość kodu oraz jego jakość.

Słowa kluczowe: Testy jednostkowe; Magento; Zend; PHP; MySQL.

***Autor do korespondencji.**

Adres e-mail: bartosz.drawdzik@pollub.edu.pl

Comparison of web application development using PHP and Magento platforms

Bartosz Drawdzik*, Maria Skublewska-Paszkowska

Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. The paper presents the process of creating two application using two different methods of programming. One of them is simple PHP application based on data stored in a relational MySQL database and a application of the same functionality implemented on the Magento 1.9 platform. The main purpose of the paper is to test applications and verify them of the extent to which functionality is fault-tolerant. It means that tests will be the - individuals function and their resistance to errors. The obtained results allow the authors to evaluate which kind of programming is the best and improve clean code and it's quality.

Keywords: Unit tests; Magento; Zend; PHP; MySQL.

***Corresponding author.**

E-mail address: bartosz.drawdzik@pollub.edu.pl

1. Wstęp

Język PHP (ang. PHP *Hypertext Preprocessor*) znajduje obecnie bardzo szerokie zastosowanie w dziedzinie informatyki, ponieważ jest głównie wykorzystywany do komunikacji z serwerem i posiada narzędzia pozwalające na budowę każdego rodzaju stron internetowych oraz umożliwia komunikację z bazami danych. Te atuty sprawiają, że jest to najczęściej stosowany język do tworzenia stron internetowych oraz aplikacji webowych w czasie rzeczywistym. Wynika to z faktu, że ponad trzy czwarte społeczeństwa w Polsce posiada dostęp do Internetu (według Głównego Urzędu Statystycznego jest to 72,4 % w roku 2017, co jest i tak poniżej średniej unijnej która wynosi 84 %) [1]. Jednoznacznie można wywnioskować więc, że zdecydowana większość ludności na co dzień korzysta z zasobów oraz tego co oferuje język PHP.

Na całym świecie powstają coraz to nowocześniejsze oraz bardziej profesjonalne strony internetowe czy inne aplikacje wykorzystujące to oprogramowanie. Nie można bowiem zapominać, że PHP to nie tylko strony internetowe, ale także wszelkiego rodzaju systemy które opierają się na zasadzie komunikacji z bazami danych czy też

Tworząc każdy system informatyczny należy patrzeć na niego z perspektywy produktu, musi on być jak najlepszy, być konkurencyjny na rynku internetowym przy tak dużej liczbie innych podobnych serwisów, musi się wyróżniać na

programowanie opierające się na przetwarzaniu informacji pobranych z serwera. Jednakże w tej dziedzinie sprawdza się najlepiej ze względu na swoje ogromne atuty i zastosowanie w połączeniu z systemem do zarządzania relacyjnymi bazami danych MySQL [2]. PHP wyróżnia się przede wszystkim szybkością działania, niskim kosztem implementacji, stabilnością, szerokim wsparciem wśród użytkowników, łatwością nauki oraz przede wszystkim, co zostało wcześniej wspomniane, wysokimi wynikami podczas współpracy z różnymi typami serwerów i silnikami bazodanowymi, wyniki te to przede wszystkim bardzo dobra kompatybilność oraz niezwykła łatwość w implementacji kodu dla różnych systemów bazodanowych [3].

Z racji tak ogromnej popularności powstaje również wiele platform programistycznych (ang. Framework) będących szkieletem budowy aplikacji, które definiują strukturę aplikacji i ogólny mechanizm działania. Takie narzędzia zapewniają także szeroki zestaw bibliotek i komponentów. Najpopularniejszym wśród nich w stosunku do języka PHP jest Zend Framework [4]. Został on użyty w tej pracy do stworzenia nie zawodnego i szybkiego systemu przy pomocy znanego i szeroko rozwiniętemu systemu zarządzania treścią jakim jest Magento [5].

ich tle. Dlatego należy dbać o poprawność działania każdego systemu. Tutaj naprzeciw każdemu programiście wychodzą testy, które pozwalają w łatwy i prosty sposób porównać

aplikacje, sprawdzić jakie ewentualne błędy mogą się pojawić [6].

Artykuł ten przedstawia zastosowanie testów jednostkowych przy porównaniu dwóch aplikacji internetowych. Wykorzystanie testów do tworzenia aplikacji pozwala na rozbicie każdego z systemów na poszczególne funkcjonalności i ich analizę. W tym przypadku porównaniu będą poddane wielkość kodu oraz szybkość działania każdej funkcjonalności. Natomiast analiza testów jednostkowych pozwoli na wyciągnięcie wniosków i odpowiedzenie na pytanie jak można poprawić jakość produktu jakim jest aplikacja internetowa pod kątem programowania w języku PHP oraz jaki wpływ na to może mieć zastosowanie tychże testów przy wytwarzaniu takiego oprogramowania.

2. Platformy

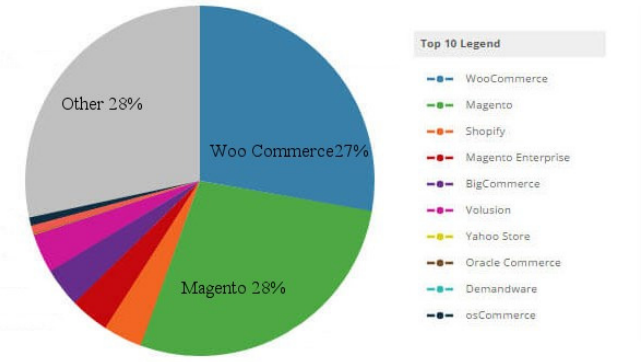
W pracy zostaną porównane dwie aplikacje. Pierwsza z nich jest zaimplementowana w języku PHP bez wykorzystania jakichkolwiek platform czy bibliotek, jedynie z wykorzystaniem PHP w wersji 5.6. Dodatkowo znajdują się tam skrypty z języka HTML oraz JavaScript. Jednak budowa całej funkcjonalności opiera się o narzędzia PHP, które pozwalają programiście na bardzo łatwy i szybki dostęp do bazy danych i komunikację z nią. Zaimplementowany serwis internetowy FitnessApp wspomaga użytkownika w prowadzeniu zdrowego trybu życia, co dzisiaj jest niezwykle popularne. Główną funkcjonalnością systemu jest wspomaganie użytkownika w prowadzeniu zdrowego trybu życia poprzez porzucenie złego nawyku palenia papierosów i notowanie jego postępów w tym procesie. Użytkownik po zalogowaniu, ma możliwość monitorowania swojego postępu w procesie rzucania palenia. Użytkownik po zapaleniu papierosa loguje się do serwisu i wpisuje dane takie jak, cenę paczki papierosów którą kupił, ostatnią datę zapalenia papierosa oraz jakiej marki był to papieros. System zapisuje te dane w bazie danych i każdy użytkownik ma możliwość wyświetlenia własnych postępów po zalogowaniu na swój profil. System wyświetla użytkownikowi najczęściej palone papierosy, daty w jakich palił te papierosy i kwotę jaką mógłby zaoszczędzić rzucając palenie.

Druga aplikacja natomiast została zaimplementowana także w języku PHP jednakże różnica polega na tym, że wykorzystana została platforma Magento. Funkcjonalności obu aplikacji są takie same.

Magento jest to otwarte oprogramowanie sklepu internetowego jednak powstające na tej platformie aplikacje mogą odbiegać od tematu głównego dla którego został zaprogramowany ten system. Przedstawiony system zarządzania treścią (ang. CMS – Content Management System) powstał na bazie najpopularniejszego frameworka PHP – Zend framework. Jest to zbiór uniwersalnych bibliotek przeznaczonych do tworzenia serwisów internetowych. Dostarcza on mechanizmów które wspierają działanie aplikacji internetowych oraz komunikację z nimi. Podstawowymi założeniami w tym zbiorze bibliotek jest stopień swobody jaki jest zostawiony dla programisty oraz wzorce projektowe które są wykorzystywane, czyli przede wszystkim MVC - (ang. Model-View-Controller). Jest to wzorzec architektoniczny którego założenia podzielone są na trzy główne części [7]. Pierwsza z nich to model reprezentujący problem i logikę działania aplikacji, kolejna część to widok, który opisuje sposób w jaki są wyświetlane pewne części modelu. Ostatnią częścią modelu MVC jest kontroler, który przyjmuje dane wejściowe i operuje na danych, wykonuje pewien działania i zwraca przetworzoną wartość.

Wszystkie te części są ze sobą połączone i razem tworzą cały spójny model, wykorzystywany najczęściej przy tworzeniu aplikacji internetowych. Taki system tworzenia serwisu jest bardzo przejrzysty i pozwala na łatwą rozbudowę widoku. Właśnie taki model MVC jest wykorzystany w platformie e-commerce Magento.

Magento jest jedną z najpopularniejszych platform sklepu internetowego. Według danych z Builtwith z 2017 roku porównując wszystkie narzędzia e-commerce, Magento wspiera ponad 28 % wszystkich sklepów internetowych w całym Internecie i ponad 16 % wszystkich stron internetowych. Takie dane plasują Magento na drugim miejscu zaraz za WooCommerce, co zostało przedstawione na rysunku 1 [8].



Rys. 1. Porównanie platform e-commerce [8]

Dane przedstawione na rysunku 1, prezentują jak szerokie zastosowanie znajduje ta platforma oraz, jak wielkim zaufaniem programiści darzą ten system zarządzania treścią. Jest to spowodowane przede wszystkim ogromnymi możliwościami jakie oferuje Magento, na stabilności całej platformy oraz możliwościach wprowadzania wszelkiego rodzaju zmian i modyfikacji całego systemu. Zmiany te mogą dotyczyć bardzo wielu rzeczy, począwszy od nie wielkich zmian graficznych jak np. logo po te bardzo rozbudowane jak przykładowo modyfikacja czy też stworzenie własnej metody wyliczania kosztów dostawy. Z tego względu druga aplikacja została stworzona właśnie przy użyciu Magento.

3. Testy jednostkowe

Testy jednostkowe (ang. Unit test) jest to jedna z metod testowania wytwarzanego programowania i polega na poddawaniu konkretnych metod (funkcji) aplikacji testom, czyli weryfikacji poprawności działania całej funkcji. Następnie otrzymane wyniki poddawane są analizie, niezależnie od tego czy dany wynik jest poprawny – zgodny z obliczeniami wstępnymi, czy też jest błędny – otrzymana wartość na wyjściu różni się od wartości spodziewanej, która jest wartością poprawna i została wyliczona przez programistę lub inną osobę, która ma odpowiednią wiedzę, aby taki poprawny wynik otrzymać. Zaletą takich testów jest

możliwość ciągłego poddawania analizie wyjściowych danych niezależnych części aplikacji na modyfikowanych elementach programu. Umożliwia to bardzo często wychwycenie błędu zanim dane zostaną przekazane dalszej części całego oprogramowania [9].

Testy jednostkowe przeprowadzane są na kilka różnych sposobów. Do najbardziej rozpowszechnionych należy wpisywanie na wejściu danych w odpowiednim miejscu i analizie otrzymanego wyniku. Ta metoda jest nazywana „debugowaniem" (ang. debug – usuwanie błędów). Innymi technikami testowania jednostkowego aplikacji jest metoda testów napędzanych rozwojem w wolnym tłumaczeniu (ang. Test-driven development), w skrócie TDD. Polega ona na testowaniu dodawanej funkcjonalności następnie implementacja funkcjonalności i refaktoryzacji kodu, czyli napisanie kodu spełniającego zadane wymagania i standardy. Obie aplikacje zostały poddane testom. Analizowano wielkość kodu, szybkość działania systemów oraz odporność na błędy i wyświetlane wartości. Aby testy jednostkowe miały sens, analiza musi spełniać pewne kryteria. Każdy z testów musi być niezależny, wykonywanie dwóch testów powinno odbywać się w ten sposób aby jeden nie wpływał bezpośrednio na wynik drugiego testu. Testy jednostkowe muszą spełniać także warunek powtarzalności i muszą być możliwie najłatwiejsze. Aplikacja powinna być wykonana w ten sposób aby w każdym momencie była możliwość przetestowania aplikacji i tak aby nie wpłynęło to na stabilność i jakość projektowanego serwisu. Jednoznaczność testów, to cecha która pozwala krótko i rzeczowo odpowiadać na pytanie czy dana metoda, funkcja działa poprawnie. Najważniejsza cecha w przypadku testów jednostkowych to ich jednostkowość przy tworzeniu testów jednostkowych powinno się pamiętać, że testowana jest tylko i wyłącznie jedną funkcjonalność, jedna metoda, nie należy także sprawdzać w jednym teście wielu zestawów danych.

4. Aplikacja

Obie aplikacje zostały zaimplementowanie w celu przeprowadzenia na nich testów jednostkowych i porównania wyników. Podczas przeprowadzanie testów starano się wyciągnąć wnioski, który z analizowanych systemów jest lepszy pod względem wymienionych wcześniej cech – wielkości i czystości kodu, szybkości działania, funkcjonalności, łatwości rozbudowy, efektywności oraz odporności na błędy. Przede wszystkim sprawdzono w jakim stopniu testy jednostkowe mogą wpłynąć na proces tworzenia aplikacji. Również ważną tutaj cechą jest stabilność i łatwość programowania jak i możliwość przeprowadzanych testów [10].

4.1. Aplikacja PHP/MySQL

Aplikacja FitnessApp została zaprogramowana używając języka PHP oraz MySQL do połączenia z lokalną bazą danych [11]. Baza danych aplikacji została zaprojektowana w programie DBeaver w wersji 3.8.5 oraz stworzona przy pomocy instalatorów w Magento. Poniższy listing przedstawia sposób w jaki to się odbywa.

Przykład 1. Skrypt łącznie z bazą danych.

```
$host='localhost';
$user='root';
$pass='';
$database='traviti';

$link=mysqli_connect($host,$user,$pass);
mysqli_select_db($link,$database) or die ("Błąd przy
wybieraniu bazy danych ".mysql_error());

setlocale(LC_ALL, 'pl_PL', 'pl', 'polish');
mysqli_query($link,"SET CHARSET utf8");
```

Jak zaprezentowano w przykładzie 1 połączenie odbywa się poprzez bezpośrednie podanie adresu serwera i danych do logowania z pliku z rozszerzeniem .php. Jakiekolwiek zmiany wprowadzone w pliku np. wprowadzenie błędnego adresu spowoduje automatycznie rozłączenie z docelową bazą danych.

Aplikacja FitnessApp składa się z trzech podstawowych funkcjonalności poddanych testom: logowanie, rejestracja, zapis i odczyt z bazy danych. Ostatnia funkcjonalność, czyli zapis i odczyt, możliwa jest tylko dla użytkowników zalogowanych. Dla pozostałych użytkowników systemu, czyli niezalogowanych, nie jest wyświetlana informacja o liczbie zapalonych papierosów w danym czasie. Zalogowany użytkownik wprowadza do formularza takie informacje jak liczba spalonych papierosów, cena paczki papierosów oraz ich marka. Takie dane umożliwiają przeprowadzenie analizy danych. W przykładzie 2, przedstawiona zostanie funkcjonalność logowania do bazy danych.

Przykład 2. Skrypt sprawdzenia loginu.

```
$sprawdzenie1 = mysqli_query($link, "SELECT * FROM
userscigarette INNER JOIN questions ON
users_cigarette.id_questions = questions.id_questions
AND login='$login'") or die("Nie ma takiego loginu w bazie");
```

Logowanie do aplikacji odbywa się w 4 krokach. Najpierw sprawdzany jest login, czy istnieje w bazie danych. Jeżeli tak, można przystąpić do kolejnego kroku jakim jest sprawdzenie wpisywanych grup. Logowanie w obu aplikacjach odbywa się poprzez podanie dwóch grup, grupy te definiowane są przez użytkownika przy logowaniu, może to być dowolna nazwa, przykładowo dla jednego użytkownika mogą to być dwie grupy, z czego pierwsza z nich to będą osoby z którymi w dzieciństwie grał w piłkę nożna, natomiast druga grupa będą to osoby z którymi przykładowy użytkownik siedział w ławce w szkole. Dla każdej z grup użytkownik definiuje dwie odpowiedzi. Oznacza to, że do każdej opisanej grupy zdefiniowane są dwie odpowiedzi i tak np. analogicznie, jeżeli jako grupę podano osoby z którymi użytkownik siedział w ławce w szkole podstawowej będą to dwa imiona i nazwiska np. Jan Kowalski, natomiast druga odpowiedź będzie to druga osoba, np. Piotr Nowak. Tak zdefiniowane dane mogą użytkownikowi ułatwić logowanie, natomiast mogą także zdecydowanie utrudnić włamanie się na konto użytkownika i dodatkowo zabezpieczyć jego dane.

Kolejnym krokiem jest sprawdzenie poprawności wpisanych grup., co zostało pokazane w przykładzie 3.

Przykład 3. Skrypt sprawdzenie grup.

```
$sprawdzenie2 = mysqli_query($link, "SELECT * FROM
userscigarette INNER JOIN questions ON
users_cigarette.id_questions=questions.id_questions AND
```

```
login='$login' AND question_one='$grupa1' AND
question_two='$grupa2'") or die("Podałeś złe nazwy grup");
```

Jeżeli podany login istnieje w bazie danych z bazy wczytane są odpowiedzi. Ponieważ login użytkownika i grupy znajdują się w różnych tabelach niezbędne jest wykorzystanie klauzuli JOIN dla MySQL.

Następnie odbywa się pobranie oraz sprawdzenie poszczególnych odpowiedzi do grup poprzez pobranie loginu i tak jak w przypadku grup użycie klauzuli JOIN i porównanie odpowiedzi, tak jak przedstawiono w przykładzie 4.

Przykład 4. Skrypt sprawdzenie pierwszej osoby.

```
$sprawdzenie3=mysqli_query($link, "SELECT * FROM
userscigarette INNER JOIN questions ON
users_cigarette.id_questions=questions.id_questions AND
login='$login' AND answer_two='$osoba1'") or die("Zła
pierwsza odpowiedz w grupie pierwsze");
```

Ostatnim krokiem przy logowaniu jest porównanie wszystkich zapytań, czy są spełnione, jeżeli tak następuje zalogowanie użytkownika, co przedstawia przykład 5.

Przykład 5. Skrypt sprawdzający prawdziwość wszystkich warunków.

```
if((mysqli_fetch_array($sprawdzenie1)) &&
(mysqli_fetch_array($sprawdzenie2)) &&
(mysqli_fetch_array($sprawdzenie3)) &&
(mysqli_fetch_array($sprawdzenie4)) &&
(mysqli_fetch_array($sprawdzenie5)) &&
(mysqli_fetch_array($sprawdzenie6)))
{
    $_SESSION['zalogowany'] = true;
    $_SESSION['login'] = $login;
    $nick = $_SESSION['login'];
    echo "zalogowales sie";
}
```

Kolejną funkcjonalnością którą należy omówić jest rejestracja użytkownika. Wszystkie dane wprowadzane do bazy danych są filtrowane prostą metodą, której działanie opiera się o sprawdzenie, czy podane dane wpisane w pole posiadają długość zawierającą się w przedziale od 1 do 100 znaków, dodatkowo dozwolone są odpowiednie grupy znaków i tak użytkownik, aby przejść walidację może wprowadzić litery zarówno duże jak i małe od a do z. Metodę tą przedstawia przykład 6.

Przykład 6. Skrypt walidacji danych z pól.

```
if(preg_match(‘[A-Za-z]{1,100}/’, $osoba5))
{
    $ok6=1;
}
```

Jeżeli wszystkie dane przejdą tą walidację, następuje sprawdzenie dostępności loginu a następnie zapisanie danych do bazy danych tak jak przedstawia to przykład 7.

Przykład 7. Skrypt rejestracji, zapisanie danych i sprawdzenie loginu.

```
$sprawdzenie=mysqli_query($link, "SELECT * FROM
`users_cigarette` WHERE login='$login'");
if(mysqli_fetch_array($sprawdzenie))
{
    echo "Jest juz taki login<br>";
    $ok1=0;
}
else
{
    $ok1=1;
}
if($ok1==1)
{
    $zapytanie_wstaw_questions = "INSERT INTO
`traviti`.`questions` (`id_questions`, `question_one`,
`answer_one`, `answer_two`, `question_two`,
`answer_three`, `answer_four`) VALUES
('$ile_users_cigarette','$grupa1', '$osoba1', '$osoba2',
'$grupa2', '$osoba5', '$osoba6')";
    $zapytania_wstaw_grupe1 = mysqli_query($link,
$zapytanie_wstaw_questions);
}
```

Ostatnią z omawianych tu funkcjonalności poddawanych testom jednostkowym jest zapis do bazy danych odpowiednich pól, pozwalających na późniejsza analizę danych. Przykład 8 przedstawia funkcjonalność pozwalającą na zapis danych do bazy, sprawdzając przy tym jaki użytkownik jest zalogowany, pobranie jego loginu, następnie zapisanie odpowiednich danych.

Przykład 8. Zapisanie wprowadzonych danych.

```
$sprawdzenie1=mysqli_query($link, "SELECT * FROM
userscigarette INNER JOIN questions ON
users_cigarette.id_questions=questions.id_questions AND
login='$login'") or die("Nie ma takiego loginu w bazie");

$zapytanie_wstaw_brand = "UPDATE `Cigarettes` INNER
JOIN userscigarette ON
Cigarettes.id_cigarette=users_cigarette.id_cigarette AND
login='$login' SET brand='$papieros'";
$zapytania_update1 = mysqli_query($link,
$zapytanie_wstaw_brand);
```

4.2. Magento

Funkcjonalności przedstawione w rozdziale 4.1 zaimplementowano w drugiej aplikacji, przy użyciu Zend Framework i Magento. Przy implementacji tej aplikacji wykorzystano edytor PhpStorm w wersji 2016.1.2 wraz z dodatkiem Magicento.

Pierwszą analizowaną funkcjonalnością będzie logowanie, które odbywa się za pomocą funkcji przedstawionej w przykładzie 9. Metoda ta pobiera kolekcję wszystkich elementów z bazy danych, następnie przekazane parametry do funkcji czyli podany przez użytkownika login oraz email są analizowane pod kątem poprawności z obiektami z bazy danych.

Przykład 9. Logowanie do serwisu, aplikacja zaimplementowana w Magento.

```
public function loggingOn($login, $email, $group, $answer)
{
    $collection =
Mage::getModel('monogo_traviti/usercigarette')-
>getCollection();
    $collection->getSelect()->joinLeft(array('question' =>
'monogo_traviti/questions'), 'main_table.id_questions =
question.id_questions');
    foreach($collection as $user)
    {
        $userLogin = $user->getLogin();
        $userEmail = $user->getEmail();
        if(($login == $userLogin)&&($userEmail == $email))
                                {
            if((in_array($user->getQuestionOne(), $group))
    && (in_array($user->getQuestionTwo(), $group)))
            {
              if($user->getQuestionOne() == $group[0])
                                                    {
```

```
                if(($user->getAnswerOne() == $answer[0])
    && ($user->getAnswerTwo() == $answer[1]))

    {
                    if($user->getQuestionTwo() ==
    $group[1])
                    {
                        if (($user->getAnswerThree() ==
$answer[2]) && ($user->getAnswerFour() == $answer[3]))
                                                    {
                                                    return 1;

                                }
                                else {
                            return 0;

                                }
                    else            {
                       return 0;

      }
                else                {
                   return 0;

    }
            else                    {
                   return 0;
                                                    }
          else  {
                   return 0;
                                    }
        else      {
                   return 0;
                                }
  }
```

Jeżeli funkcja przedstawiona na listingu 9 zwróci wartość jeden oznacza to, że użytkownik o podanym loginie i e-mailu istnieje w bazie danych i można przejść do zalogowania. W przeciwnym wypadku wyświetlany jest komunikat o błędnych danych logowania.

Następnym elementem funkcjonalności jest odczyt z bazy danych podczas rejestracji użytkownika. Poniższa funkcja przedstawiona na listingu 10, ukazuje w jaki sposób jest to wykonane w Magento.

Przykład 10. Rejestracja do serwisu, aplikacja Magento.

```
public function checkLoginEnable($login, $email)
{
    $collection =
Mage::getModel('monogo_traviti/usercigarette')-
>getCollection();
      foreach($collection as $user)   {
         $userLogin = $user->getLogin();
         $userEmail = $user->getEmail();
         if(!($login == $userLogin))    {
            return 0;
         }
         if($userEmail == $email)    {
            return 1;
         }
      }
      return 0;
   }

   public function insertUser($login, $email)
   {
      $collection = Mage::helper('monogo_traviti')-
>getUserCigaretteCollection();
      $sizeCollection = count($collection) + 1;
      $data = array('id_questions' => $sizeCollection,
'id_cigarette' => $sizeCollection, 'email' => $email, 'login'
=> $login);

      $model =
Mage::getModel('monogo_traviti/usercigarette');
      $model->setData($data);
      $model->save();
   }
```

W przykładzie 10 przedstawiono dwie funkcje. Pierwsza „checkingLoginEnable" pozwala sprawdzić dostępność loginu oraz adresu e-mail w serwisie podanych przez użytkownika. Jeżeli login oraz e-mail są dostępne funkcja zwraca wartość 1. Natomiast druga funkcja, „insertUser", przedstawia sposób w jaki odbywa się zapis danych w Magento.

Ostatnia funkcjonalność przedstawia sposób w jaki wykonuje się funkcja zapisu danych do bazy w celu pobrania statystyk późniejszych, co zostało przedstawione w przykładzie 11. Funkcja ta pozwala pobrać jako parametr dane które są niezbędne do wpisania do bazy. Następnie pobierany jest model wykorzystywanych danych w tabeli „cigarettes" i następuje zapis danych.

Przykład 11. Zapis danych w Magento.

```
public function updateSmoking($answer, $group)
{
    $data = array('brand' => $answer[0], 'price' =>
    $answer[1]);

    $model = Mage::getModel('monogo_traviti/cigarettes');
    $model->setData($data);
    $model->save();
}
```

5. Środowisko testowe

Testy obu aplikacji, FitnessApp oraz tej samej aplikacji zaimplementowanej na platformie Magento 1.9, zostały przeprowadzone w jednym środowisku testowym, którego parametry zostały przedstawione w tabeli 1 Wyniki jakie otrzymano podczas testowania zostały zapisane oraz następnie przedstawione poniżej.

W środowisku testowym zainstalowany jest system operacyjny Ubuntu 16.04 LTS 64-bitowy . Natomiast przeglądarka wykorzystywana do testów to Google Chrome w wersji 59.0.3071.104 (Oficjalna wersja) (64-bitowa). Natomiast język PHP jest zainstalowany w wersji 5.6.

Tabela 1. Specyfikacja urządzeń testowych

Model	**ASUS X550JK**
CPU	Intel Core i7-4710HQ 2,50 GHz x8
Pamięć RAM	12 GB DDR3
HDD	SATAII 500GB
GPU	Intel Haswell Mobile
Karta sieciowa	Broadcom BCM43142 802.11b/g/n Wireless Network Adapter + Atheros AR8172/8176/8178 PCI-E Fast Ethernet Controller
Rozdzielczość ekranu	1920x1080px

5.1. Wyniki badań

Analizując powyższe dwie aplikacje otrzymano zestaw wyników, które zostaną przedstawione w tabeli 2. Tabela ta stanowi punkt wyjściowy do dalszej analizy poszczególnych systemów i wyciągnięcia wniosków na temat każdego z nich.

Tabela 2. Zestawienie wyników badań

Cecha	Magento 1.9	PHP/MySQL
Wielkość kodu - logowanie - rejestracja - dodawanie	 73 linii 67 linii 28 linii	 87 linii 140 linii 60 linii
Czas ładowania - logowanie - rejestracja - dodawanie	 2,57 sec 3.56 sec 1,51 sec	 0,78 sec 0,94 sec 0,59 sec
Łatwość rozbudowy (w skali 1-10)	8	5
Efektywność rozbudowy (w skali 1-10)	7	4
Funkcjonalność (w skali 1-10)	9	5
Czystość kodu (w skali 1-10)	10	6
Łatwość programowania (w skali 1-10)	3	8

Testy zostały wykonane dla zestawu 5 danych wejściowych. Po zakończeniu testów, w przypadku czasu ładowania strony została wyciągnięta średnia arytmetyczna z wszystkich wyników i została zaprezentowana w tabeli 2. Pozostałe cechy zostały poddane ocenie dwóm osobom, każda z nich jest programistą korzystającą z platformy Magento, co sprawia, że ocena tych aplikacji może wydawać się subiektywna, jednak starano się oddać całkowity i jak najbardziej realistyczny obraz dwóch systemów i w sposób jednoznaczny przedstawić sposób działania i ogromną wagę jaką powinno się przykładać przy programowaniu na testowanie aplikacji, żeby były one możliwie jak najbezpieczniejsze, odporne na błędy oraz optymalne.

Cała analiza danych i testy zostały przeprowadzone na wszystkich funkcjonalnościach systemu a dane zestawione w tabeli 2 prezentują podsumowanie poszczególnych aplikacji internetowych. Podczas testów nieokreśloną pomocą w aplikacji wykonanej za pomocą Magento okazała się funkcja Mage::log() [12], pozwalająca w pliku logów wyświetlić poszczególne elementy wprowadzane do funkcji jak i ich przetwarzanie. Szeroko rozbudowany system jakim jest platforma e-commerce pozwala także dzięki modelowi MVC na wielokrotne wykorzystanie raz zdefiniowanych funkcji w Helper, odwołując się do nich. Jest to podstawowy atut dla efektywności i łatwości rozbudowy całego systemu, co spowodowało, że ten właśnie serwis wykonany przy pomocy frameworka Zend został wyżej oceniony niż aplikacja FitnessApp. Niestety jednak czas ładowania strony różni się i to znacząco, ponieważ Magento wykorzystuje wiele bibliotek i załączenie każdej z nich zajmuje znaczącą ilość czasu. Na końcu porównując czytelność kodu, zdecydowanie i to bardzo mocno przeważa aplikacja w Magento, jest to spowodowane ogromną liczbą wspomnianych wcześniej bibliotek które pozwalają wykorzystać wbudowane funkcje i wykorzystać je do dalszej pracy

6. Wnioski

Przedstawione w artykule dwa programy różnią się miedzy sobą sposobem wykonania, jednak ich funkcjonalność pozostaje ta sama. Oba serwis działają poprawnie, zapisują i odczytują dane oraz analizują je. Aplikacja FitnessApp nie posiada natomiast walidacji adresu email, w przeciwieństwie do aplikacji korzystającej z Magento 1.9. Liczne biblioteki wybudowane pozwalają systemowi zbudowanemu za pomocą platformy e-commerce na niezwykle szerokie pole do rozwoju i to z ogromną łatwością. Mimo różnicy czasu jaka pojawia się przy tworzeniu aplikacji, aplikacja z wykorzystaniem platformy Magento jest dużo bardziej skuteczna, intuicyjna i posiada większe możliwości. Testy jednostkowe wykazały, że aplikacja też posiada znacznie mniej błędów, czyli jest mniej awaryjna oraz bardziej odporna na wprowadzanie błędnych danych, waliduje je poprawnie. Jednym z minusów tak zaprojektowanej aplikacji jest trudność nauki jaką jest przyswojenie wszystkich funkcji wbudowanych. Jednak ich opanowanie otwiera przed programistą ogromne możliwości.

Literatura

[1] http://www.internetlivestats.com/internet-users-by-country/ Statystyki Internetu [14.07.2017],

[2] Yank K., Build Your Own Database-Driven Website Using PHP & MySQL, SitePoint, 2003,

[3] Ross J., PHP i HTML. Tworzenie dynamicznych stron WWW, Helion, 2010,

[4] http://www.cs.put.poznan.pl/jkobusinski/php.html Opis oraz z statystyki Zend framework [17.07.2017]

[5] Sanborn M., Lehm M., Leising R., Fowler R. Documentation for Magento Developers, 660 York ST San Francisco CA 94110.

[6] Martin R. C., Czysty kod. Podręcznik dobrego programisty, Helion, 2015.

[7] Gajda W., PHP, MySQL i MVC. Tworzenie witryn WWW opartych na bazie danych, Helion, 2010.

[8] http://www.cminds.com/magento-updated-statistics/ Statystyki Magento [10.07.2017]

[9] Beck K. Test Driven Development: By Example 1st Edition

[10] Lerdorf R., Tatroe K., MacIntyre P., Programming PHP, Helion, 2007.

[11] Feathers Michael Working Effectively with Legacy Code (Robert C. Martin Series) 1st Edition, Kindle Edition.

[12] Branko Ajzele, Magento 2 Developer's Guide, 2015.