

## ЕФЕКТИВНІСТЬ РОЗРОБКИ ВЕБ СЕРВІСІВ ДЛЯ ЗАБЕЗПЕЧЕННЯ ЯКОСТІ ПОСЛУГ

Запропоновано структурований підхід до оцінки технічних і ергономічних характеристик веб сайтів. З метою підвищення ефективності розробки, визначені критерії вибору PHP-технологій. Описано характеристики систем і інструментів, які використовуються для розробки веб сайтів та наведені загальні етапи оцінки ефективності їх впровадження.

Proposed a structured approach to the assessment of the technical and ergonomic characteristics of web sites. In order to improve the development of criteria for choosing PHP-technology. We describe the characteristics of systems and tools used for developing web sites and stages are general assess the effectiveness of their implementation.

### Вступ

Дослідження веб-сайтів передбачає оцінку їх технічних і ергономічних характеристик. Застарілі характеристики систем і інструментів спотворюють висновки розробників і користувачів про додатки, які базуються на цих системах. Під технічними розуміємо характеристики, пов'язані з функціонуванням програмного коду системи - доступність сайту, швидкість завантаження сторінок, відповідність стандартам і т.п., а ергономічними - дизайн, зручність навігації, якість подання інформації та інтуїтивність інтерфейсу.

Мета роботи - підвищення ефективності розробки веб сервісів для забезпечення високої якості послуг, що надаються. Актуальність дослідження - у визначенні критеріїв вибору PHP-технологій для створення веб сайтів, з метою підвищення ефективності розробки.

Створення сайту, який реалізує якісну підтримку користувачів торгових центрів, з урахуванням фінансової складової даних проєктів, досить складне завдання для програмістів. Його рішення пов'язане з витратами часу і коштів. Для вирішення подібного роду завдань використовують CMS - системи управління контентом, які за рахунок обмежень, що накладаються на логічну структуру контенту, дизайн і функціональні можливості створюваних динамічних сайтів, істотно знижують витрати на їх розробку і підтримку.

Більшість подібних систем здатні функціонувати в режимі хостингу і включають в себе зберігання, відстеження версій, контроль за до-

ступом, інтеграцію з іншими інформаційними системами та управління потоком документів.

Однак, щоб врахувати всі індивідуальні особливості проєкту, краще створити веб сайт з нуля. Тоді, для спрощення розробки, використовують фреймворки - фундамент, який визначає архітектуру програми і містить у собі налагоджений код, для вирішення завдань, які часто використовуються. На базі фреймворка можна написати CMS, розробити сервіс або веб додаток [1].

### 1. Огляд інструментальних засобів

Практично всі популярні PHP-фреймворки використовують архітектуру MVC - Model-View-Controller (рис. 1).

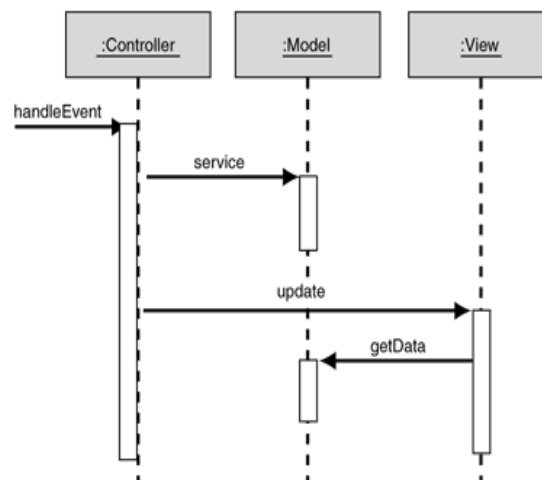


Рис.1. Структура MVC-фреймворку

У схемі, яка описує модель додатку, інтерфейс, призначений для користувача і взаємодія з користувачем розділені на три компонента так, щоб модифікація одного з компонентів мінімально впливала на інші. Фреймворк пропонує вже вбудовані класи для: роботи з базою даних; створення функціональних форм; валідації; логування та ін. [2].

OpenCart - сучасна платформа, побудована за принципом MVC і встановлюється на веб-серверах з підтримкою PHP і MySQL. На її базі можна створити веб сайт будь-якої складності. Joomla - популярна CMS, з можливістю установки на будь-яку кількість сайтів з можливістю зміни вихідного коду [3]. Особливості системи: прискорена завантаження сторінок з кешуванням; режими налагодження і звіти про помилки; доступ по протоколу FTP; загальна поштова система; віддалений виклик процедур через HTML і XML. Symfony - вільний фреймворк, написаний на PHP5 і використовує патерн Model-View-Controller. Сильні сторони Symfony2: висока швидкість за рахунок використання шаблонізатору Twig з оптимізованим PHP-кодом; гнучкість в налаштуванні: конфігурація додатків у трьох різних форматах: анотації, XML, YAML; розширюваність; стабільність; орієнтованість на розробника [4]. Yii - високопродуктивний компонентний PHP фреймворк, з використанням моделі MVC [2]. Переваги: простота установки; використання сучасних технологій; гнучкість і розширюваність; тестування; безпека; скорочення часу на розробку; легко налаштувати кращу продуктивність. Технічна складова сайту - набір показників відповідають за якість ресурсу. Найважливішими характеристиками вважають доступність ресурсу в інтернеті, швидкість відповіді сервера, правильність налаштувань, проект повинен працювати стабільно і відповідати всім стандартам [3]. Зниження швидкості роботи може викликати перевантаження сервера. Для оптимізації графіки слід звертати увагу на розмір зображень перед завантаженням. Скрипти на javascript і flash можуть не зовсім коректно оброблятися пошуковими роботами. До вказаного додаємо кросбраузерність коду і кросплатформеність, валідність, правильне кодування і відповідність коду прийнятим стандартам World Wide Web Consortium (W3C) [5].

Виходячи з цього отримано основні складові оцінки технічних характеристик сайту: швидкість роботи; доступність; валідність коду; відповідність стандартам web; проблема дублювання сторінок.

Практика показує, що між красою і зручністю інтерфейсу існує безпосередній зв'язок. Ідеальний сайт повинен поєднувати інформативність, функціональність і візуальну привабливість. До основних методів оцінки юзабіліті відносяться експертна оцінка, спостереження, юзабіліті-тестування, експеримент. Відомо про п'ять складових юзабіліті: інтуїтивність; ефективність; запам'ятовуваність; помилки; задоволеність. Методика оцінки юзабіліті: найбільш достовірний метод вивчення юзабіліті сайту, вона полягає в спостереженні за тим як вирішують задачу представники цільової аудиторії [6].

## 2. Аналіз методів розробки

Показник продуктивності фреймворка - максимальна кількість оброблюваних системою запитів [3]. За допомогою тестування визначимо приблизні межі чуйності фреймворків: Yii2; Symfony; Cake; Aura і Zend Framework при виклику компонентів controller і view.

Для кожного фреймворка застосовувалися такі скрипти (або еквіваленти): конфігурація за замовчуванням; фронт-контролер, або диспетчер; сторінковий контролер, або ж контролер дії; дію без коду і викликає процесор подання; зріз, що містить виключно символічний рядок "Hello world!". Для автоматизації процесу тестування написаний скрипт вимірювання кількості запитів в секунду на localhost, що дозволяє не враховувати мережеві перешкоди знижують швидкість. В даному випадку - 10 запитів за 60 секунд. Код скрипта вказано в лістингу 1.

Лістинг 1. Виклик Apache benchmark

```
ab -c 10 -t 60 http://localhost/[path],
```

де path - шлях до сторінці згідно ієрархії веб сайту. Тестовий сценарій перезапускає сервер і викликає команду ab 5 разів. Ці дії повторюються для кожного тестованого фреймворка. Результати тестування вказані в таблиці 1.

Таблиця 1.

| Результат тестування для простої html- і php-сторінок |         |         |         |         |         |         |
|-------------------------------------------------------|---------|---------|---------|---------|---------|---------|
| Сторінка                                              | 1       | 2       | 3       | 4       | 5       | AVG     |
| baseline-html                                         | 2613.56 | 2284.98 | 2245.98 | 2234.94 | 2261.01 | 2328.09 |
| baseline-php                                          | 1717.74 | 1321.49 | 1292.86 | 1511.40 | 1327.35 | 1434.17 |

Як видно з таблиці, веб сервер має високу пропускну здатність і для статичних сторінок обробляє 2328 запитів в секунду, у випадку з php-сторінкою (включений XCache), яка виконує команду echo "Hello, world" пропускну здатність сповільнюється до 1434 запитів. Під час тестування в кожному окремому випадку виконано примусове привласнення ідентифікатора сесії. Тестування для Cake запущено з рівнем налагодження рівним нулю і настройками без внесення інших змін. Код контролера для Cake приведено в лістингу 2. Контролер сторінки використовує пов'язане index.html уявлення.

Лістинг 2. Cake контролер

```
<?php
namespace App\Controller;
class HelloController extends AppController
{
    public $autoRender = false;
    public function index()
    { $this->response->body('Hello World!');
      return $this->response; }
}
```

Контролер для Aura наведено в лістингу 3.

Лістинг 3. Aura контролер

```
<?php
namespace Vendor\Package\Web;
use Aura\Web\Controller\AbstractPage;
class Page extends AbstractPage
{
    public function actionIndex()
    { $content = "Hello, World!";
      $this->data->content = $content; }
}
?>
```

Контролер сторінки автоматично використовує файл "index.php" в якості свого уявлення, і не використовує ніяких шаблонів. Код контролера для Symfony вказано в лістингу 4. На початок файлу для web / index.php прописано примусове встановлення ідентифікатора сесії.

Лістинг 4. Symfony контролер

```
<?php
namespace Acme\HelloBundle\Controller;
```

```
use Symfony\Component\HttpFoundation\ Response;
class HelloController
{
    public function indexAction()
    { return new Response('Hello World!');
    }
}
```

Для тестування Zend Framework проведені аналогічні маніпуляції (ліст. 5).

Лістинг 5. Zend Framework контролер

```
<?php
namespace Application\Controller;
use Zend\Mvc\Controller\AbstractActionController;
use Zend\View\Model\ViewModel;
class HelloController extends AbstractActionController
{
    public function indexAction()
    { return $this->getResponse()
->setContent('Hello World!'); }
}
```

Код контролеру Yii2 наведено у лістингу 6.

Лістинг 6. Yii2 контролер

```
<?php
namespace backend\controllers;
use Yii;
use yii\web\Controller;
class SiteController extends Controller
{
    public function actionIndex()
    {
        $hello = 'Hello World';
        return $this->render('index', ['hello' => $hello]);
    }
}
```

Для більшої наочності введена відносна оцінка пропускну здатності - за одиницю приймається значення пропускну здатності фреймворка показав найменший результат. Результати всіх тестів вказані у таблиці 2.

Розглянемо характеристики, створених на фреймворках Yii2 і Symfony сайтів.

Таблиця 2.

| Результати тестування пропускної здатності фреймворків |        |        |        |        |        |        |      |
|--------------------------------------------------------|--------|--------|--------|--------|--------|--------|------|
| Фреймворк                                              | 1      | 2      | 3      | 4      | 5      | AVG    | REL  |
| Cake                                                   | 114,62 | 114,26 | 114,55 | 113,89 | 114,64 | 114,39 | 1    |
| Aura                                                   | 170,8  | 169,22 | 170,29 | 170,75 | 170,22 | 170,26 | 1,49 |
| Symfony                                                | 196,75 | 219,4  | 226,86 | 202,52 | 217,34 | 212,57 | 1,86 |
| Zend                                                   | 315,91 | 308,5  | 307,84 | 310,43 | 311,73 | 310,88 | 2,72 |
| Yii2                                                   | 384,45 | 376,82 | 393,11 | 375,79 | 386,14 | 383,26 | 3,35 |

Протестуємо CMS. Для кожного CMS встановлена віртуальна машина з веб сервером, базою даних і ПЗ для тестування. Кожна CMS встановлена в віртуальному середовищі і в її код додано рядки для збереження інформації про час виконання скрипта і об'ємі пам'яті. Зміни у коді наведено в лістингу 7.

Лістинг 7. Зміни у коді для тестування

```
$_test_loadstart = microtime(true);
// код CMS
$_test_loadend = microtime(true);
$_test_loadtime = number_format($_test_loadend - $_test_loadstart, 3);
$_test_memory = number_format(memory_get_usage()/1048576, 3);
file_put_contents($_SERVER['HTTP_HOST'] . '.txt', $_test_loadtime . ';' . $_test_memory . "\n", FILE_APPEND);
```

На виході - файл domain.local.txt з даними про час завантаження скрипта і об'єм оперативної пам'яті. Для запитів до сайту використано Apache benchmark. Код наведено в лістингу 8.

Лістинг 8. Запуск Apache benchmark

```
ab -n 1000 http://domain.local/ >
~/bench/ab/domain.local.txt
```

В результаті тестування надано оцінку за такими параметрами [7]: швидкість завантаження скрипта; розмір оперативної пам'яті; результат тестування Apache benchmark. У кожному розділі тестування для CMS виставлені оцінки ( $-2 \leq x \leq +2$ ) в залежності від отриманих результатів (особистих) і середніх (avg) всієї вибірки:

- -2 - найгірший результат в даному тесті;
- -1 - результат, гірший ніж середнє в тесті;
- +0,5 - результат близький до середнього в межах standart deviation;
- +1 - результат, кращий за середнє в тесті;
- +2 - кращий результат в тесті.

За результатами тестування складена підсумкова таблиця за всіма тестами. Результати оцінки швидкості завантаження скрипта надані в таблиці 3, розміру оперативної пам'яті в таблиці 4, тестування утилітою Apache benchmarks в таблиці 5, а підсумкові надано в таблиці 6.

Таблиця 3.

| Оцінка швидкості завантаження скрипта |           |        |        |          |            |        |        |        |
|---------------------------------------|-----------|--------|--------|----------|------------|--------|--------|--------|
| Test/CMS, час завантаження (сек)      | Wordpress | Joomla | Drupal | OpenCart | InstantCMS | AVG    | MIN    | MAX    |
| AVG                                   | 0,4977    | 0,2106 | 0,2137 | 0,3253   | 0,5482     | 0,3591 | 0,2106 | 0,5482 |
| MIN                                   | 0,467     | 0,19   | 0,194  | 0,305    | 0,518      | 0,3348 | 0,19   | 0,518  |
| MAX                                   | 0,661     | 0,317  | 1,387  | 0,489    | 0,851      | 0,741  | 0,317  | 1,387  |
| Зведені дані                          | -1,5      | 6      | 4      | 2,5      | -3,5       |        |        |        |

Таблиця 4.

| Оцінка розміру оперативної пам'яті |           |        |        |          |            |         |        |        |
|------------------------------------|-----------|--------|--------|----------|------------|---------|--------|--------|
| Test/CMS, пам'ять                  | Wordpress | Joomla | Drupal | OpenCart | InstantCMS | AVG     | MIN    | MAX    |
| AVG (mb)                           | 22,992    | 12,556 | 18,144 | 6,4850   | 5,9538     | 13,226  | 5,9538 | 22,992 |
| MIN (mb)                           | 22,992    | 12,556 | 18,144 | 6,485    | 5,951      | 13,2256 | 5,951  | 22,992 |
| MAX (mb)                           | 22,992    | 12,556 | 18,147 | 6,504    | 6,885      | 13,4168 | 6,504  | 22,992 |
| Зведені дані                       | -6        | 1,5    | -3     | 5        | 6          |         |        |        |

Таблиця 5.

| Оцінка Apache benchmarks |           |         |        |          |             |         |         |         |
|--------------------------|-----------|---------|--------|----------|-------------|---------|---------|---------|
| Test/CMS                 | Wordpress | Joomla  | Drupal | OpenCart | Instant CMS | AVG     | MIN     | MAX     |
| Час тестування (сек)     | 520,498   | 340,219 | 230,91 | 224,176  | 555,886     | 374,338 | 224,176 | 555,886 |
| Повні запити             | 1000      | 1000    | 1000   | 1000     | 1000        | 1000    | 1000    | 1000    |
| Невиконані               | 0         | 0       | 0      | 0        | 0           | 0       | 0       | 0       |
| Запитів за секунду       | 1,92      | 2,94    | 4,33   | 4,46     | 1,8         | 3,09    | 1,8     | 4,46    |
| Швидкість передачі даних | 14,9      | 28,61   | 32,99  | 13,33    | 92,36       | 36,438  | 13,33   | 92,36   |
| Зведені дані             | -3        | 3,5     | 7      | 8        | -8          |         |         |         |

Таблиця 6.

| Підсумкові результати |           |        |        |          |            |
|-----------------------|-----------|--------|--------|----------|------------|
| Test/CMS              | Wordpress | Joomla | Drupal | OpenCart | InstantCMS |
| phpload_time          | -1,5      | 6      | 4      | 2,5      | -3,5       |
| phpload_memory        | -6        | 1,5    | -3     | 5        | 6          |
| Apache benchmark      | -3        | 3,5    | 7      | 8        | -8         |
| Зведені дані          | -10,5     | 11     | 8      | 15,5     | -5,5       |

Як видно з таблиці, треба розглядати додатки, створені за допомогою CMS OpenCart і Joomla, з огляду на отримані ними найкращі оцінки при тестуванні за різними критеріями.

### 3. Модель системи

Для розробки системи в якості каркаса обрано фреймворк Yii2. Структура програми відповідає механізму його роботи [8]. Класи створені згідно вимог моделі MVC, розміщуються у відповідності з advanced-шаблоном і належать необхідному простору імен.

Послідовність процесу обробки запиту: користувач виконує запит за URL і веб сервер обробляє його, запускаючи скрипт ініціалізації, який створює екземпляр додатку і запускає його на виконання; додаток отримує інформацію про запит користувача від компонента додатка request, визначає контролер і дію за допомогою компонента urlManager; потім створює екземпляр контролера для подальшої обробки запиту; дія зчитує з бази даних модель Product з id рівним 1 та підключає представлення show, передаючи в нього модель Product; представлення отримує і відображає атрибути моделі Product

та підключає деякі віджети; сформоване представлення вставляється в макет сторінки і дія завершує його формування та виводить результат.

Модель системи - набір класів (рис. 2), рознесених на frontend, backend і common частини, всередині кожної з них класи, які знаходяться в директорії відповідно до моделі MVC. Так як авторизацію виконують класи фреймворка Yii2, то для роботи програми внесені правки в модель User при взаємодії її з моделлю Profile профілю користувача. За маршрутизацію запитів до профілю та механізми пов'язані з авторизацією відповідає UserController.

Окремо реалізована модель зображення, змінювати яку можна за допомогою відповідного контролера.

Відображення контролерами візуальної частини здійснюється завдяки походженням їх від головного контролера \yii\base\Controller. Він містить спеціалізовані методи рендеринга, які, в свою чергу, використовують універсальний клас View, що відповідає за відображення html-коду.

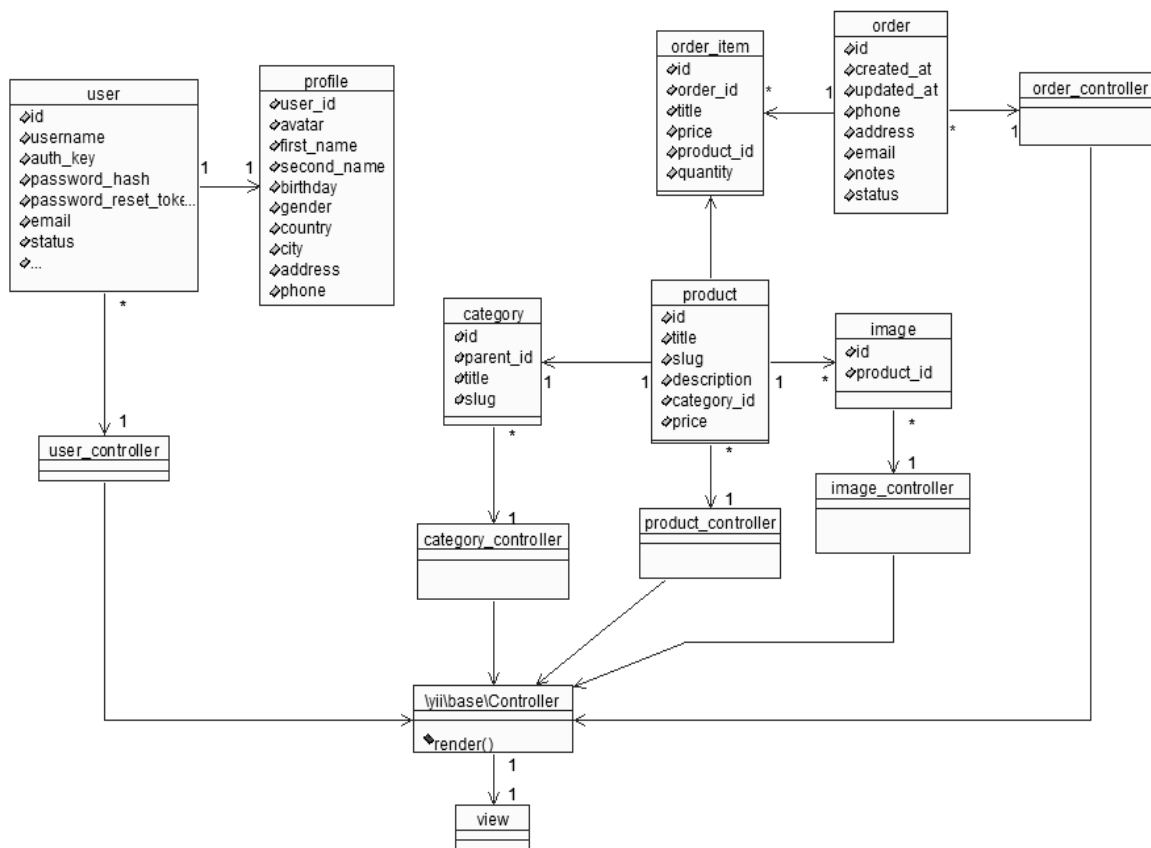


Рис.2. Модель системи

Внесення змін в структуру бази даних вимагає зміни вихідного коду, фреймворк Yii2 підтримує можливість міграції баз даних. Вона дозволяє відстежувати зміни в базі за допомогою системи контролю версій. Згідно з концепцією ActiveRecord, для кожної з таблиць створюється спеціальний AR-клас, який є відображенням таблиці в об'єктно-орієнтованому підході. Так таблиці бази даних відповідають класам моделі в структурі програми. Система сервісного обслуговування підтримує три призначені для користувача ролі: адміністратор, клієнт і гість, які відповідають різним можливостям.

#### 4. Оптимізація системи

Кешування - простий і ефективний спосіб підвищити продуктивність. Yii підтримує всі механізми кешування: кешування даних; фрагментів; сторінок; HTTP-кешування [9]. Найбільш ефективним є кешування сторінки цілком. Кешування статичної сторінки проводиться методом кешування фрагментів (лістинг 9).

Лістинг 9. Кешування статичної сторінки

```
<?php if ($this->beginCache(__FILE__ . __LINE__, ['duration' => 24 * 3600])): ?>
```

```
<!-- Зміст статичної сторінки -->
<?php endif; ?>
```

Тут `__FILE__` `__LINE__` є конкатенацією імені .php-файлу і номеру рядка в скрипті, а параметр `duration` вказує на тривалість зберігання копії. Результати застосування кешування фрагмента наведено на рисунках 3 і 4.

### Performance Profiling

Total processing time: **187 ms**; Peak memory: **5.8 MB**.

Рис.3. Час завантаження до включення кешування

### Performance Profiling

Total processing time: **83 ms**; Peak memory: **5.8 MB**.

Рис.4. Час завантаження після включення кешування

Yii2 дозволяє налаштувати `yii \ filters \ HttpCache` як фільтр для дії контролера, що відображає результат кешування на стороні клієнта. Він працює для GET і HEAD запитів.

Важливим елементом в оптимізації продуктивності є кешування схеми - спеціальний тип кешування, який включають при використанні ActiveRecord. Він за допомогою додаткових

SQL запитів, отримує інформацію про схему (імена та типи стовпців, обмеження) таблиці бази даних без необхідності описувати її вручну. При включенні кешування схеми, ця інформація зберігається в кеші і повторно використовується при нових запитах [10].

Для включення кешування схеми, потрібно налаштувати компонент cache, а для зберігання інформації про схему встановити у `\ db \ Connection :: enableSchemaCache` в `true` у програмі. Підключення наведено у лістингу 10.

Лістинг 10. Кешування схеми

```
return [
    'cache' => [
        'class' => 'yii\caching\FileCache', ],
    'db' => [
        'enableSchemaCache' => true,
        'schemaCacheDuration' => 3600,
        'schemaCache' => 'cache', ],
];
```

При розробці сайту компонентом кешу застосовано `ApсCache`. Він використовує розширення `PHP APC` та є обгорткою для `Yii2`. Ця опція вважається найшвидшою при роботі з кешем в «товстому» централізованому додатку.

Не менш важливим в оптимізації роботи сайту є кешування самих запитів до бази даних. Кешування запитів - функція, побудована на основі кешування даних. Вона призначена для кешування результатів запитів до бази даних.

Наведений нижче код (лістинги 11 - 12) демонструє підключення кешування запитів до бази даних сайту. Виконується кешування запитів на видачу інформації про профілі користувача. Кеш вважається недійсним, якщо користувач вносить зміни в профіль, змінивши поле `updated_at`.

Лістинг 11. Кешування запитів до бази даних

```
$dependency = new DbDependency;
$dependency->sql = 'SELECT
MAX(updated_at) FROM {{%user}}';
$profile = \Yii::$app->db->cache(function ($db) {
    return UserProfile::find()->where
    ([ 'user_id' => \Yii::$app->user->identity->getId()
    ])->one();
},
3600, $dependency);
```

Лістинг 12. Кешування `dataProvider` на сторінці з категоріями

```
$dataProvider = new ActiveDataProvider([
    'query' => $productsQuery,
    'pagination' => [
        'pagesize' => 3,
    ], ]);
$dependency = new DbDependency;
$dependency->sql = 'SELECT count(id) FROM
product';
\Yii::$app->db->cache(function ($db) use
($dataProvider) { $dataProvider->prepare();
}, 3600, $dependency);
```

Ресурс в `Yii` – файл. Він може бути `CSS` чи `JavaScript` файл, зображення або відео файл. Ресурси розміщують в `Web` директоріях і обслуговуються безпосередньо `Web` серверами. `Web` сторінка включає багато `CSS` або `JavaScript` файлів. Для скорочення кількості запитів і загального розміру файлів, практикують об'єднання і стиснення декількох `CSS/ JavaScript` файлів в один та розміщення нових на `Web` сторінках.

`Yii` надає консольну команду з ім'ям `asset` для автоматизації роботи по мініфікації ресурсів. Такий підхід позбавляє розробника від рутинної роботи з розробки каркасу для конфігураційних налаштувань і допомагає уникнути помилок. Для використання команди, спочатку створюємо файл конфігурації, далі використаємо команду `asset / template`, щоб згенерувати перший шаблон і відредагувати під свої потреби. Ця команда згенерує файл із зазначеним ім'ям.

Налаштування полягає в зміні файлу, вказавши в `bundles` параметри комплектів, які планується об'єднати. У параметрі `targets` вказується як комплекти розділені по групах.

Параметри `jsCompressor` і `cssCompressor` в лістингу 13 вказують на консольні команди чи зворотний виклик `PHP`, виконують `JavaScript` і `CSS` об'єднання / стиснення. Для об'єднання / стиснення `CSS` і `JavaScript` файлів у комплектах використані спеціальні сторонні інструменти - `Closure Compiler` для мініфікації `JavaScript`-файлів, і `YUI Compressor` для `css`-файлів відповідно.

Лістинг 13. Налаштування файлу `asset-config.php`

```
...
'jsCompressor' =>
'java -jar compiler.jar --js {from} --
js_output_file {to}',
'cssCompressor' =>
```

```
'java -jar yuicompressor.jar --type css {from} -o {to}',
'bundles' => [ 'frontend\assets\SaleAsset' ],
'targets' => [
'all' => [
'class' => 'yii\web\AssetBundle',
'basePath' => '@webroot/assets',
'baseUrl' => '@web/assets',
'js' => 'js/all-{hash}.js',
'css' => 'css/all-{hash}.css',
],
],
...

```

Для коректної роботи налаштуємо asset manager в конфігурації програми (лістинг 14).

Лістинг 14. Налаштування asset manager

```
'assetManager' => [
'bundles' => [
'all' => [
'class' => 'yii\web\AssetBundle',
'basePath' => '@webroot/assets',
'baseUrl' => '@web/assets',
'css' => ['css/allbc92550a861e387fc9adf706a56c5be7.css'],
'js' => ['js/all-04cb2989a84063893c3aaeb83e9de729.js'], ], 'frontend\assets\SaleAsset' =>
['css' => [], 'js' => [], 'depends' =>
['all']], ],
]

```

Тепер завантаження сторінки прискорюється через зниження ваги ресурсів, які отримуються.

### Висновки

У роботі проведено порівняльний аналіз сайтів, які використовують різні технології та мову PHP. На підставі тестування ефективності CMS і фреймворків обрано ті, які показали найкращу пропускну здатність і використання пам'яті для свого сегменту. У випадку з CMS це OpenCart і Joomla, а у випадку з фреймворками - Yii2 і Symfony. Для усіх сайтів проведено тес-

тування загальних, технічних та ергономічних характеристик, які безпосередньо впливають на якість, швидкість і ефективність роботи системи. За результатами аналізу виявлено, що хоча системи, розроблені на основі CMS дешевше і простіше в розробці, рівень їх продуктивності на порядок нижче. А строгі обмеження і можливість тонкої індивідуальної настройки модулів додатків, розроблених на основі фреймворков, дають кращі результати. Також проведена оптимізація системи, що дозволило підвищити рівень ефективності її роботи.

### СПИСОК ЛІТЕРАТУРИ

1. *CMS, PHP фреймворк или CMF*. – Режим доступа: <http://promo-creative.com/cms-php-frejmwork-ili-cmf.html>.
2. *Model-View-Controller*. – Режим доступа: <https://msdn.microsoft.com/en-us/library/ff649643.aspx>.
3. Колисниченко Д. Выбираем лучший бесплатный движок для сайта. CMS Joomla и Drupal /Д. Колисниченко.– СПб.:БХВ-Петербург, 2010.– 288 с.
4. *Symfony. The Architecture*. – Режим доступа: [http://symfony.com/doc/current/quick\\_tour/the\\_architecture.html](http://symfony.com/doc/current/quick_tour/the_architecture.html).
5. *Критерии качества сайта. Технические требования к сайту*. – Режим доступа: <http://websartor.ru/sozdanie-saita.htm>.
6. *Эргономика и красота* – Режим доступа: <http://habrahabr.ru/company/alee/blog/117313/>.
7. *New Year's Benchmarks* – Режим доступа: <http://paul-m-jones.com/archives/238>.
8. *Митрохин Е. 7 причин в пользу выбора фреймворка Yii 2.0* / Е.Митрохин. – Режим доступа: <http://exabyte.ws/php-i-mysql/7-prichin-v-polzu-vybora-frejmworka-yii-2-0.html/>.
9. *Кеширование в фреймворке Yii2*. – Режим доступа: <https://yiiframework.com.ua/ru/doc/guide/2/caching-overview/>.
10. *Оптимизация производительности*. – Режим доступа: <https://yiiframework.com.ua/ru/doc/guide/2/tutorial-performance-tuning/>.

G.G. Kirichek. **The effectiveness the development web-services to ensure quality of service.**

Г.Г. Киричек. **Эффективность разработки веб-сервисов для обеспечения качества услуг.**

Предложен структурированный подход к оценке технических и эргономических характеристик веб-сайтов. С целью повышения эффективности разработки определены критерии выбора PHP-технологий. Описаны характеристики систем и инструментов, которые используются для разработки веб-сайтов и приведены общие этапы оценки эффективности их внедрения.