Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Дубика Дениса Павловича*
(ПІБ)

академічної групи *122-18ск-1*
(шифр)

спеціальності *122 Комп'ютерні науки*
(код і назва спеціальності)

освітньої програми *Комп'ютерні науки*
(назва освітньої програми)

на тему: *Розробка веб-ориентованого додатку для продажу побутового обладнання для очистки води на мові програмування PHP з використанням фреймворку Zend*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*доц. Гуліна І.Г.*			
розділів:				
спеціальний	*доц. Гуліна І.Г.*			
економічний	*проф. Вагонова О.Г.*			
Рецензент	*доц. Шедловський І.А.*			
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2021

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » _____ 2021 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *122-18ск-1* *Дубика Дениса Павловича*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка веб-ориентованого додатку для продажу побутового обладнання для очистки води на мові програмування PHP з використанням фреймворку Zend*

затверджена наказом ректора НТУ «ДП» від «07 » 06. 2021 р. № 317-с

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів виробничої практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі.*** ***Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми.***	***13.05.2021 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки.***	***27.05.2021 р.***

Завдання видав _____ *доц. Гуліна І.Г.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Дубик Д.П.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2021 р.*

Термін подання кваліфікаційної роботи до ЕК: *10.06.2021 р.*

РЕФЕРАТ

Пояснювальна записка: 80 с., 19 рис., 3 дод., 29 джерел.

Об'єкт розробки: інтернет-магазину з продажу побутового обладнання для очистки води.

Мета кваліфікаційної роботи: надати можливість користувачу робити покупки через інтернет-магазину з продажу побутового обладнання для очистки води та впровадити методи, що збільшать ефективність роботи інтернет-магазину.

У вступі розглядається аналіз та сучасний стан проблеми, конкретизується мета кваліфікаційної роботи та галузь її застосування, наведено обґрунтування актуальності теми та уточнюється постановка завдання.

У першому розділі проведено аналіз предметної галузі, визначено актуальність завдання та призначення розробки, розроблена постановка завдання, задані вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі виконано аналіз існуючих рішень, обрано вибір платформи для розробки, виконано проектування і розробка програми, наведено опис алгоритму і структури функціонування підсистеми, визначені вхідні і вихідні дані, наведені характеристики складу параметрів технічних засобів, описаний виклик та завантаження застосунку, описана робота програми.

В економічному розділі визначено трудомісткість розробленої інформаційної підсистеми, проведений підрахунок вартості роботи по створенню застосунку та розраховано час на його створення.

Практичне значення полягає у створенні підсистеми, що забезпечує швидку та точну реакцію інтерфейсу веб-сайту на дії користувачів, і як наслідок – збільшує ефективність роботи інтернет-магазину.

Актуальність підсистеми маршрутизації запитів визначається великим попитом на подібні розробки, що оптимізують та спрощують дії щодо обслуговування клієнтів інтернет-магазину, скорочують час на оформлення замовлень.

Список ключових слів: МАРШРУТИЗАЦІЯ, ЗАПИТ, ІНТЕРНЕТ-МАГАЗИН, САЙТ, БАЗА ДАНИХ, PHP, HTML, MVC.

ABSTRACT

Explanatory note: 80 pp., 19 figs., 3 apps., 29 sources.

Object of development: online store for the sale of household equipment for water purification.

The purpose of the diploma project: to enable the user to make purchases through the online store for the sale of household water treatment equipment and to introduce methods that will increase the efficiency of the online store.

The introduction considers the analysis and the current state of the problem, specifies the purpose of the qualification work and the field of its application, provides a justification for the relevance of the topic and clarifies the problem.

In the first section the analysis of the subject area is carried out, the urgency of the task and purpose of development is defined, the statement of the task is developed, requirements to software realization, technologies and software are set.

The second section analyzes the existing solutions, selects the platform for development, performs design and development of the program, describes the algorithm and structure of the subsystem, determines the input and output data, provides characteristics of the parameters of hardware, describes the call and application load, describes the program .

In the economic section, the complexity of the developed information subsystem is determined, the cost of work on creating the application is calculated and the time for its creation is calculated.

The practical significance lies in the creation of a subsystem that provides a fast and accurate response of the website interface to user actions, and as a result - increases the efficiency of the online store.

The relevance of the query routing subsystem is determined by the high demand for such developments, which optimize and simplify actions to serve customers of the online store, reduce the time to place orders.

Keywords: ROUTING, REQUEST, ONLINE STORE, SITE, DATABASE, PHP, HTML, MVC.

ЗМІСТ

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ

БД - база даних

ЕОМ - електронно-обчислювальна машина

ІС - інформаційна система

ПЗ - програмне забезпечення

СУБД - система управління базами даних

CMS - система управління контентом

ІТ - інформаційні технології

SEO - пошукова оптимізація

ВСТУП

Інтернет-магазин - один з найбільш перспективних способів ведення бізнесу в даний час. Все більше покупок відбувається в онлайні, і все частіше користувачі Інтернету, чия діяльність пов'язана з бізнесом, створюють інтернет-магазини для збільшення прибутку і зниження витрат. Ця тенденція виникла завдяки простоті управління і гнучкості функціональних рішень торгових веб-сайтів.

В умовах жорсткої конкуренції кожен клієнт на рахунку, між ресурсами йде серйозна боротьба за покупців, та використовуються всілякі методи збільшення продажів інтернет магазину. В інтернет-середовищі маркетологи, веб-майстри та власники сайтів використовують велику кількість способів, що дозволяють утримати користувача на сайті. Заходи із залучення й утримання потенційних клієнтів збільшують конверсію ресурсу, що дає можливість будь-якому інтернет-магазину впевнено конкурувати на сучасному ринку.

Користувач в першу чергу орієнтується на привітний інтерфейс і швидкість завантаження веб-сторінки. Будь-яке зволікання в роботі ресурсу, нагромадження інформації або погана її структурованість ведуть до втрати потенційного покупця. У більшості випадків, взаємодія користувача з веб-застосунком проходить за допомогою переходів по посиланнях, отриманих в результаті виконання запитів до вхідного контролера застосунку. Однак обробка запитів до веб-сайта зі складною структурою передбачає виконання великої кількості аналогічних дій, і якщо поведінка вхідного контролера розподілена по декількох об'єктах, це може привести до значного уповільнення роботи веб-застосунку.

Таким чином, актуальною задачею є розробка підсистеми маршрутизації запитів, призначення якої полягає у забезпеченні централізованої обробки запитів до компонентів і сторінок інтерактивного веб-сайту, що зробить процес придбання товару швидшим і зручнішим, і підвищить ефективність роботи інтернет-магазину.

Завдання кваліфікаційної роботи та об'єкт її діяльності безпосередньо пов'язані з зі спеціальностю 122 «Комп’ютерні науки» та відповідає узагальненій тематиці кваліфікаційних робіт і переліку зазначених компетенцій.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Загальні відомості з предметної галузіі

Обробка запитів до Web-сайтів зі складною структурою передбачає виконання великої кількості аналогічних дій. Це перевірка безпеки, забезпечення підтримки інтернаціоналізації та відкриття спеціальних уявлень для особливих категорій користувачів. Якщо поведінку вхідного контролера буде розкидано по декількох об'єктах, це призведе до дублювання великої кількості коду. Крім того, це значно ускладнить зміну поведінки контролера під час виконання запиту.

Типове рішення «контролер запитів» об'єднує всі дії з обробки запитів в одному місці, розподіляючи їх виконання за допомогою єдиного об'єкта-обробника. Як правило, цей об'єкт реалізує загальну поведінку, яку може бути змінено під час виконання з допомогою декораторів. Для виконання конкретного запиту обробник викликає відповідний об'єкт команди.

Розглянемо життєвий цикл запита для MVC-додатку (рис. 1.1).

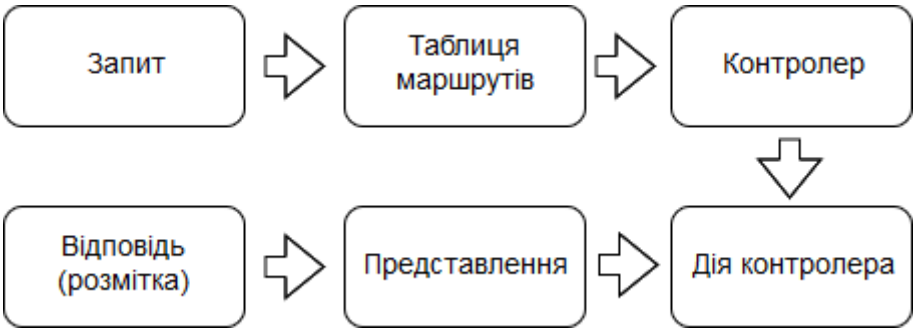

Рис. 1.1. Життєвий цикл запита для MVC-додатку

HTTP-запит, що надходить до веб-сервера, передається до середовища виконання, яке ініціалізує інфраструктуру MVC і передає запит для обробки компоненту маршрутизації. На підставі таблиці маршрутизації, що завантажується при запуску веб-додатку, модуль маршрутизації визначає імена

контролера і методу контролера, який повинен обробити запит, а також параметри запиту, які повинні бути передані контролеру. Після цього генерується контекст запиту, що містить параметри запиту та середовища виконання програми (такі як URL запиту, IP-адреса клієнта і сервера і т.п.), створюється екземпляр класу контролера, і йому передається управління шляхом виклику відповідного методу класу контролера - дії контролера в термінах MVC.

Метод контролера на підставі параметрів запиту виконує деяку логіку, вибирає представлення, яке має бути відображене користувачеві, і передає представлення механізму генерації розмітки (движку представлення в термінах MVC), який вже відображає представлення.

Для обміну даних між представленням і контролером використовується спеціальна колекція ViewData, яка є основною сполучною ланкою між контролером і представленням.

Після того, як розмітка була згенерована движком представлення, веб-сервер повертає її в якості відповіді користувачу по протоколу HTTP.

Контролер запитів обробляє всі запити, що надходять до Web-сайту, і зазвичай складається з двох частин: Web-обробника і ієрархії команд. Web-обробник-це об'єкт, який виконує фактичне отримання POST- або GET-запитів, що надійшли на Web-сервер. Він витягує необхідну інформацію з адреси URL і вхідних даних запиту, після чого вирішує, яку дію необхідно ініціювати, і делегує його виконання відповідної команді.

Web-обробник зазвичай реалізується у вигляді класу (рис. 1.2), а не сторінки сервера, оскільки він не генерує ніяких відгуків. Команди також є класами, а не сторінками сервера; більш того, їм не потрібно знати про наявність Web-оточення, не дивлячись на те, що їм часто передається інформація з HTTP-запитів. У більшості випадків Web- обробник - це досить проста програма, функції якої полягають у виборі потрібної команди.

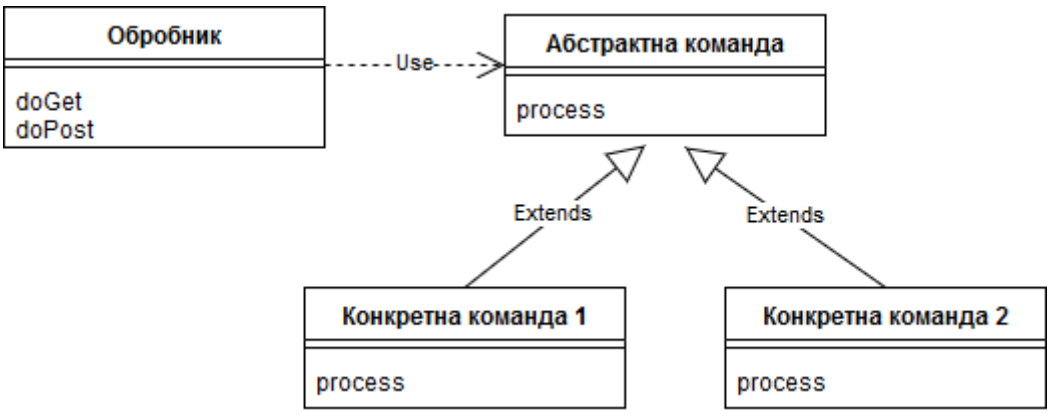

Рис. 1.2. Диаграма класів контролера запитів

Вибір команди може відбуватися статично або динамічно. Статичний вибір команди передбачає проведення синтаксичного аналізу адреси URL і застосування умовної логіки, а динамічний - витяг деякого стандартного фрагмента адреси URL і динамічне створення екземпляра класу команди.

Переваги статичного вибору команди складаються у використанні явного коду, наявності перевірки часу компіляції і високої гнучкості можливих варіантів написання адреси URL. У свою чергу, використання динамічного підходу дозволяє додавати нові команди, не вимагаючи зміни Web-обробника.

При динамічному виборі команд ім'я класу команди можна помістити безпосередньо до адреси URL або скористатися файлом властивостей, який буде прив'язувати адреси URL до імен класів команд. Зрозуміло, це потребує створення додаткового файлу властивостей, проте дозволить легко змінювати імена класів, не переглядаючи всі наявні на сервері Web-сторінки.

Контролер запитів має більш складну структуру, ніж його очевидний суперник - контролер сторінок (Page Controller). Але він має декілька важливих переваг. По-перше, Web-сервер повинен бути налаштований на використання тільки одного контролера запитів; всю іншу роботу по розподілу запитів буде виконувати Web-обробник. Це значно полегшить конфігурацію Web-сервера (особливо якщо цей процес досить трудомісткий). Використання динамічного вибору команд дозволяє додавати нові команди без будь-яких змін. Крім того,

динамічні команди полегшують переносимість, бо потрібно лише "зареєструвати" обробник в настройках сервера.

Оскільки для виконання кожного запиту створюються нові екземпляри класів команд, не доведеться турбуватися про те, щоб вони функціонували в окремих потоках. Це дозволить уникнути всіх неприємностей, пов'язаних з багатопотоковим програмуванням; тим не менш, доведеться стежити за відсутністю спільного використання інших об'єктів, таких, як об'єкти моделі.

Найбільш згадуваною перевагою контролера запитів є можливість реалізації спільного коду, який так чи інакше, дублюється в численних контролерах сторінок. Проте, багато хто забуває, що загальна поведінка останніх з не меншим успіхом може бути винесено в суперклас контролерів сторінок.

Контролер запитів тільки один, що дозволяє легко змінювати його поведінку під час виконання з допомогою численних декораторів. Декоратори можуть застосовуватися для проведення аутентифікації, вибору кодування, забезпечення підтримки інтернаціоналізації і т.п. Більш того, їх можна додавати не тільки за допомогою файлу налаштувань, але і прямо під час роботи сервера.

1.2. Призначення розробки та область застосування

Як об'єкт впровадження розроблюваної підсистеми маршрутизації запитів розглядається веб-сайт з продажу водоочисного обладнання «Бриз», який реалізує фільтри для очищення води різних типів, комплектуючі вироби та аксесуари. На сайті також розміщені новини та статті, завдяки яким клієнт магазину може самостійно розібратися в великому асортименті фільтруючого обладнання.

По суті, даний сайт є структурованим каталогом товарів з функціями пошуку товару, підбору водоочисної техніки по ряду критеріїв. Інтернет-магазин «Бриз» розрахований на роздрібну торгівлю, і для підвищення

продажів веб-сайт повинен бути візуально привабливим і зручним у використанні.

Розроблений застосунок призначений для:

- надання універсального інструменту перенаправлення URL;
- забезпечення перенаправлення некоректних URL на відповідні сторінки;
- синхронізації взаємодії дочірніх контролерів.

1.3. Підстава для розробки

Підставами для розробки та виконання кваліфікаційної роботи є:

- освітня програма спеціальності 122 «Комп'ютерні науки»;
- навчальний план та графік навчального процесу;
- наказ ректора Національного технічного університету «Дніпровська політехніка» № 317-с від 07.06.2021 р;
- завдання на кваліфікаційну роботу на тему: «Розробка веб-орієнтованого додатку для продажу побутового обладнання для очистки води на мові програмування PHP з використанням фреймворку Zend».

1.4. Постановка завдання

Завданням кваліфікаційної роботи є розробка підсистеми маршрутизації запитів для інтерактивного веб-сайту з продажу водоочисного обладнання. Програмне забезпечення призначене для забезпечення швидкої та точної реакції інтерфейсу веб-сайту на дії користувачів.

Програма повинна реалізувати наступні функції:

- класифікація і парсинг вхідного URL;
- виклик необхідних методів дочірніх контролерів в залежності від атрибутів URL;

- переадресація користувальницького запиту на відповідні сторінки в разі некоректного URL.

Для досягнення поставленої мети необхідно:

- вивчити предметну галузь розв'язуваної задачі;
- створити алгоритм для реалізації поставленого завдання;
- створити базу даних і клієнтську програму, що працює з нею.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Для досягнення поставлених цілей програмне забезпечення, що розробляється, повинно підтримувати виконання наступних дій:

- надання віддаленого доступу до застосунку через веб-браузер на комп'ютері користувача;
- зчитування вихідних даних з адресного рядка веб-браузера;
- зберігання даних підсистеми в реляційній базі даних.

Для виконання перерахованих вище функцій у застосунку повинні бути реалізовані:

- можливість інтеграції в платформу веб-сайту з продажу водоочисного обладнання;
- можливість отримати доступ до програми через веб-браузер;
- наявність типової конфігурації, що забезпечує можливість швидкого введення застосунку в експлуатацію;
- програмно-апаратна переносимість.

1.5.2 Вимоги до інформаційної безпеки

Для уникнення некоректної роботи програми необхідно реалізувати:

- семантичний та синтаксичний контроль вхідних даних;
- обробку виняткових ситуацій;
- виведення повідомлень про помилки;
- можливість повторного введення даних;
- можливість безперервної роботи протягом не менше 120 годин (5 діб);
- платформну незалежність.

Як складовий елемент системи, повязаної із торгівельною діяльністю, підсистема маршрутизації запитів повинна мати наступні характеристики:

- час відновлення після збою - 5 хв;
- час відновлення після відмови одного з елементів підсистеми не повинно перевищувати половини операційного банківського дня;
- вірогідність виникнення не більше 2 логічних помилок на 1000 операторів за 1 рік експлуатації;
- забезпечення неушкодженого стану даних, що зберігаються в базі даних, у випадку відмови підсистеми.

1.5.3 Вимоги до складу та параметрів технічних засобів

Для нормального функціонування програми необхідно, щоб обчислювальна машина, на якій буде функціонувати веб-орієнтована підсистема, відповідала наступним вимогам:

- ✓ процесор класу Intel Xeon з тактовою частотою не менш 2.4 ГГц;
- ✓ не менше 2 GB оперативної пам'яті;
- ✓ рідкокристалічний монітор з діагоналлю не менше 17 ";
- ✓ 20 Гб вільного місця на жорсткому диску;

- ✓ доступ до мережі Internet;
- ✓ клавіатура;
- ✓ маніпулятор "миша".

Наведені вище технічні характеристики є рекомендованими, тобто при наявності технічних засобів не нижче зазначених, розроблений програмний виріб буде функціонувати відповідно до вимог щодо надійності, швидкості обробки даних і безпеки, висунутими замовником.

1.5.4 Вимоги до інформаційної та програмної сумісності

Для нормального функціонування програми необхідно, щоб програмне забезпечення обчислювальної машини, на якій буде функціонувати веб-орієнтована підсистема, відповідало наступним вимогам:

- ✓ операційна система сімейства Windows (XP, Vista, 7, 8, 10), Linux;
- ✓ веб-браузер Firefox / Google Chrome / Opera / Internet Explorer.

Застосунок має бути реалізовано на мові програмування PHP з використанням фреймворку Zend та СУБД MySQL.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Функціональне призначення системи

В рамках даної кваліфікаційної роботи була виконана розробка програмного забезпечення, що реалізує підсистему маршрутизації запитів, призначену для забезпечення швидкої та точної реакції інтерфейсу веб-сайту на дії користувачів.

Призначення розробленої підсистеми:

- надання універсального інструменту перенаправлення URL;
- забезпечення перенаправлення некоректних URL на відповідні сторінки;
- синхронізація взаємодії дочірніх контролерів.

Розроблена програма реалізує наступні функції:

- класифікація і парсинг вхідного URL;
- виклик необхідних методів дочірніх контролерів в залежності від атрибутів URL;
- переадресація користувальницького запиту на відповідні сторінки в разі некоректного URL.

Для досягнення поставленої задачі розроблене програмне забезпечення підтримує виконання таких операцій:

- надання віддаленого доступу до застосунку через веб-браузер на комп'ютері користувача;
- зчитування вихідних даних з адресного рядка веб-браузера;
- зберігання даних підсистеми в реляційній базі даних.

2.2. Опис застосованих математичних методів

Оскільки особливості предметної галузі розв'язуваної задачі не передбачають застосування математичних методів, при розробці підсистеми маршрутизації запитів математичні методи не використовувалися.

2.3. Опис використаних технологій та мов програмування

Застосунок має бути реалізовано на мові програмування PHP з використанням фреймворку Zend. В якості СУБД застосовано MySQL Server.

Опис СУБД MySQL

MySQL - це популярна система управління базами даних, дуже часто застосовується в поєднанні з PHP. У реляційній базі даних дані зберігаються в окремих таблицях, завдяки чому досягається виграш в швидкості і гнучкості. Таблиці зв'язуються між собою за допомогою відношень, завдяки чому забезпечується можливість об'єднувати при виконанні запиту дані з декількох таблиць. SQL як частина системи MySQL можна охарактеризувати як мову структурованих запитів та найбільш поширену стандартну мову, яка використовується для доступу до баз даних.

MySQL - це ПЗ з відкритим кодом, кожен користувач може вивчити вихідний код якого і змінити його відповідно до своїх потреб. Використання програмного забезпечення MySQL регламентується ліцензією GPL (GNU General Public License), http://www.gnu.org/licenses/, в якій зазначено, що дозволяється і забороняється робити з цим програмним забезпеченням в різних ситуаціях.

Веб-програмісти часто віддають перевагу СУБД MySQL, оскільки вона є дуже швидкою, надійною і легкою у використанні. Спочатку сервер MySQL розроблявся для управління великими базами даних з метою забезпечити більш високу швидкість роботи в порівнянні з існуючими на той момент аналогами. І

ось уже протягом кількох років даний сервер успішно використовується в умовах промислової експлуатації з високими вимогами. Завдяки своїй доступності, швидкості і безпеки MySQL дуже добре підходить для доступу до баз даних по Internet.

Застосунок на PHP, що використовує для зберігання інформації базу даних (зокрема MySql) завжди працює швидше, ніж застосунки, побудовані на файлах. Справа в тому, що бази даних написані на мові C ++, а написати на PHP програму, яка працювала б з жорстким диском ефективніше бази даних - завдання нерозв'язна за визначенням, оскільки програми на PHP в принципі працюють повільніше, ніж програми на C ++, бо PHP - інтерпретатор, а C ++ - компілятор.

MySQL є системою клієнт-сервер, яка містить багатопотоковий SQL-сервер, що забезпечує підтримку різних обчислювальних машин баз даних, а також кілька різних клієнтських програм і бібліотек, засоби адміністрування і широкий спектр програмних інтерфейсів (API). Поставка сервера MySQL у вигляді багатопотокової бібліотеки, яку можна підключити до призначеного для користувача застосунку, дозволяє отримати компактний, більш швидкий і легкий в управлінні продукт. Доступна також велика кількість програмного забезпечення для MySQL, в більшій частині - безкоштовного.

Структура MySQL трирівнева: бази даних - таблиці - записи. Бази даних і таблиці MySQL фізично представляються файлами з розширеннями frm, MYD, MYI. Логічно таблиця являє собою сукупність записів. А записи - це сукупність полів різного типу. Ім'я бази даних MySQL унікально в межах системи, а таблиці - в межах бази даних, поля - в межах таблиці. Один сервер MySQL може підтримувати відразу кілька баз даних, доступ до яких може розмежовуватися логіном і паролем. Знаючи ці логін і пароль, можна працювати з конкретною базою даних. Наприклад, можна створити або видалити в ній таблицю, додати записи і т. Д. Зазвичай ім'я-ідентифікатор і пароль назначаються хостинг провайдерами, які і забезпечують підтримку MySQL для своїх користувачів.

MySQL є повнофункціональною реляційною СУБД, що відрізняється функціональністю і надійністю, а також великими перспективами за рахунок її розширюваності і вільної ліцензії. З огляду на все вищесказане, дана СУДБ була обрана в якості СУБД для розроблюваної системи.

Опис мови програмування PHP

PHP (англ. PHP: HypertextPreprocessor - «PHP: препроцесор гіпертексту») - мова програмування, створена для генерування HTML-сторінок на веб-сервері і роботи з базами даних. В даний час підтримується переважною більшістю хостинг-провайдерів. Входить в LAMP - «стандартний» набір для створення веб-сайтів (Linux, Apache, MySQL, PHP (Python або Perl)).

PHP - це мова сценаріїв загального призначення, однак він створювався спеціально для Web і може використовуватися безпосередньо в HTML-коді. Переважним призначенням PHP є надання web-розробникам можливості швидкого створення web-сторінок, що динамічно генеруються. Великою перевагою PHP є наявність великого набору вбудованих функцій для конкретних СУБД, в тому числі функцій для MySQL.

Синтаксис PHP дуже нагадує синтаксис мови C і багато в чому запозичений з таких мов, як Java і Perl. Програміст C дуже швидко освоїть мову PHP і зможе використовувати її з максимальною ефективністю. У PHP є практично всі оператори і функції, наявні в стандартному GNU C (або їх аналоги), наприклад, є цикли (while, for), оператори вибору (if, switch), функції роботи з файловою системою і процесами (fopen, * dir, stat, unlink, popen, exec), функції введення-виведення (fgets, fputs, printf) і безліч інших.

PHP є одною з найпопулярніших скриптових мов (разом з JSP, Perl і мовами, використовуваними в ASP.NET) завдяки своїй простоті, швидкості виконання, багатій функціональності і розповсюдженню початкових кодів на основі ліцензії PHP. PHP відрізняється наявністю ядра і модулів, «розширень»: для роботи з базами даних, сокетами, динамічною графікою, криптографічними бібліотеками, документами формату PDF і т. П. Будь-який бажаючий може розробити своє власне розширення і підключити його. Існують сотні

розширень, проте в стандартну поставку входить лише кілька десятків, що добре зарекомендували себе. Інтерпретатор PHP підключається до веб-серверу або через модуль, створений спеціально для цього сервера (наприклад, для Apache або IIS), або як CGI-програми.

Крім цього, він може використовуватися для вирішення адміністративних завдань в операційних системах UNIX, GNU / Linux, MicrosoftWindows, Mac OS X і AmigaOS.

В даний час PHP використовується сотнями тисяч розробників. Близько 20 мільйонів сайтів повідомляють про роботу з PHP, що складає більше п'ятої частки доменів Інтернету.

Фреймворк Zend

Zend Framework - це каркас веб - застосунку, розроблений компанією Zend (компанія, що здійснює підтримку і координацію проекту php).

Zend Framework - це бібліотека класів, на основі якої за певними правилами будується застосунок. Варто зазначити, що використання бібліотек класів істотно скорочують час на розробку програми, за рахунок використання раніше створеного і налагодженого коду. І що не менш важливо, цей код можна модифікувати, використовуючи механізм успадкування. Розробники Zend Framework реалізували безліч класів, які дозволяють реалізувати стандартні завдання, які стоять перед веб - програмістом. Наприклад, доступ до баз даних, механізм аутентифікації, кешування і тд.

Zend Framework використовує архітектуру MVC (Model-view-controller).

Розробники Zend Framework наводять такі переваги фреймворку:

- ✓ Zend розширює мову php, зберігаючи її концепцію;
- ✓ головним критерієм Zend є простота;
- ✓ використані найкращі прийоми об'єктно - орієнтованого програмування;
- ✓ дружня ліцензія;
- ✓ добре протестований швидко - виконуваний код.

Основний упор в Zend Framework зроблений на можливість побудови добре захищених, надійних і сучасних веб 2.0 застосунків і веб-сервісів, і

всепоглинаючих широкодоступних API - функцій від лідируючих в даній сфері команд, таких, як Google, Amazon, Yahoo !, Flickr.

Zend Framework слідує за останніми напрямками в сфері веб-застосунків, таким, як: підтримка Ajax, Search - php редакція Lucene індустріального стандарту пошукових систем, і т.д.

Zend Framework надає якісну бібліотеку класів на php 5, використовуючи такі добре зарекомендовані прийоми, як шаблони проектування (designpatterns), модульне тестування (unit - тестування), слабкі зв'язки (oosecoupling).

2.4. Опис структури системи та алгоритмів її функціонування

Сучасні системи управління контентом використовують кілька ключових підходів до власної внутрішньої архітектурі. Однак підхід, що зустрічається найбільш часто, полягає в тому, що сайт фактично складається з трьох основних компонентів: підсистеми візуалізації, контролера і сховища даних. Такий підхід дозволяє відокремити бізнес-логіку web-застосунку від технології зберігання даних і програмного коду, керуючого відображенням графічної і текстової інформації. Крім того, такий підхід дозволяє будувати систему, в якій можуть бути присутні кілька систем візуалізації, в тому числі і не web-орієнтованих. Web-браузер клієнта звертається до web-серверу, під керуванням якого функціонують контролери та підсистеми візуалізації. Підсистема візуалізації може включати в себе XML-парсери, парсери шаблонів сторінок, код, отвевающій за формування web-сторінок на підставі різних описів.

У момент, коли користувач заходить на сторінку веб-застосунку, Контролер викликає Модель, яка повертає останні 10 записів, витягнутих з бази даних. Далі дані передаються з Контролера в Підсистему візуалізації, яка формує і виводить користувачеві web-сторінку. Для реалізації підсистеми візуалізації в PHP проектується один або кілька спеціальних класів, організованих у вигляді сукупності окремих PHP-файлів.

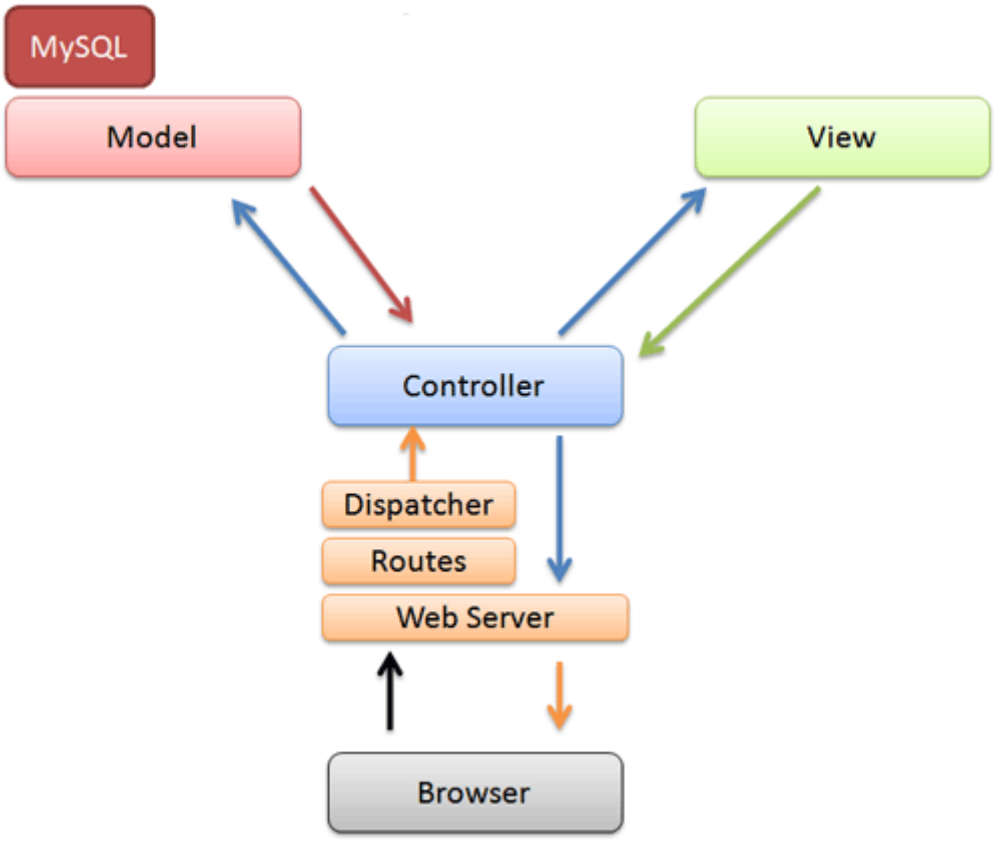

Рис. 2.1. Схема взаємодії компонентів сайту

FrontController (фронт-контролер, єдина точка входу) - шаблон проектування, який є спеціалізованою варіацією шаблону проектування Посередник. Завданням фронт-контролера є надання єдиної точки входу для обробки всіх запитів і виклик відповідної поведінки в залежності від запиту. Фронт-контролер найчастіше використовується в веб-застосунках, де є багато подібних речей, які потрібно виконати при обробці запиту. Це може бути безпека, інтернаціоналізація, забезпечення певного виду для певних користувачів. Якщо обробка вхідних запитів розподілена між декількома контролерами, це може привести до дублювання поведінки. Крім того, виникають складнощі зі зміною поведінки під час виконання. Фронт-контролер об'єднує обробку запитів шляхом їх направлення через єдиний об'єкт-обробник. Цей об'єкт реалізує загальну поведінку, яка може бути змінена під час виконання з допомогою декораторів. Після цього FrontController створює

необхідні об'єкти за запитом і викликає методи для реалізації конкретного завдання.

Фронт контролер може бути реалізований у вигляді Java - об'єкта, або, як скрипт PHP, ASP, JSP або CFML, що викликається на кожен запит веб - сесії. Цей скрипт, наприклад, index.php, буде обробляти всі запити, які є загальними для веб-застосунку або фреймворка, наприклад, обробка сесій, кешування і фільтрація вхідних даних.

Альтернативою фронт-контролера можуть бути окремі скрипти, наприклад, login.php і order.php, для обробки певного типу запиту. Кожному скрипту доведеться дублювати код або об'єкти, які є загальними для всіх запитів, але кожен скрипт має порівняно велику гнучкість для обробки конкретного запиту.

Для проектування роботи підсистеми був побудований алгоритм маршрутизації запитів всередині веб-застосунку інтернет-магазину, яке реалізує розгорнуту систему пов'язаних між собою посиланнями сторінок (рис. 2.2). Інтерфейсом розробляється підсистеми є клас Controller.

При розробці підсистеми маршрутизації був застосований підхід, при якому запити до сайту обробляються централізовано (через один фронт-контролер), а адреси та вміст сторінок зберігаються в базі даних. При цьому адреса сторінки може бути динамічною, а вміст - включати в себе мітки для підстановки змінних. Сторінці також може бути призначений виконуваний PHP-код.

За допомогою файлу .htaccess всі запити до сайту перенаправляються на вхідну точку - файл index.php, що лежить в корені document root. Не перенаправляються тільки запити на реально існуючі файли. index.php запускає $ bootstrap-> run () - основний метод завантажувального класу Bootstrap.php. Цей клас призначений для ініціалізації і налаштування оточення. При зверненні до будь-якої сторінки веб-сайту викликається фронт-контролер, який аналізує URL запиту і на підставі правил маршрутизації викликає відповідний метод для подальшої обробки запиту.

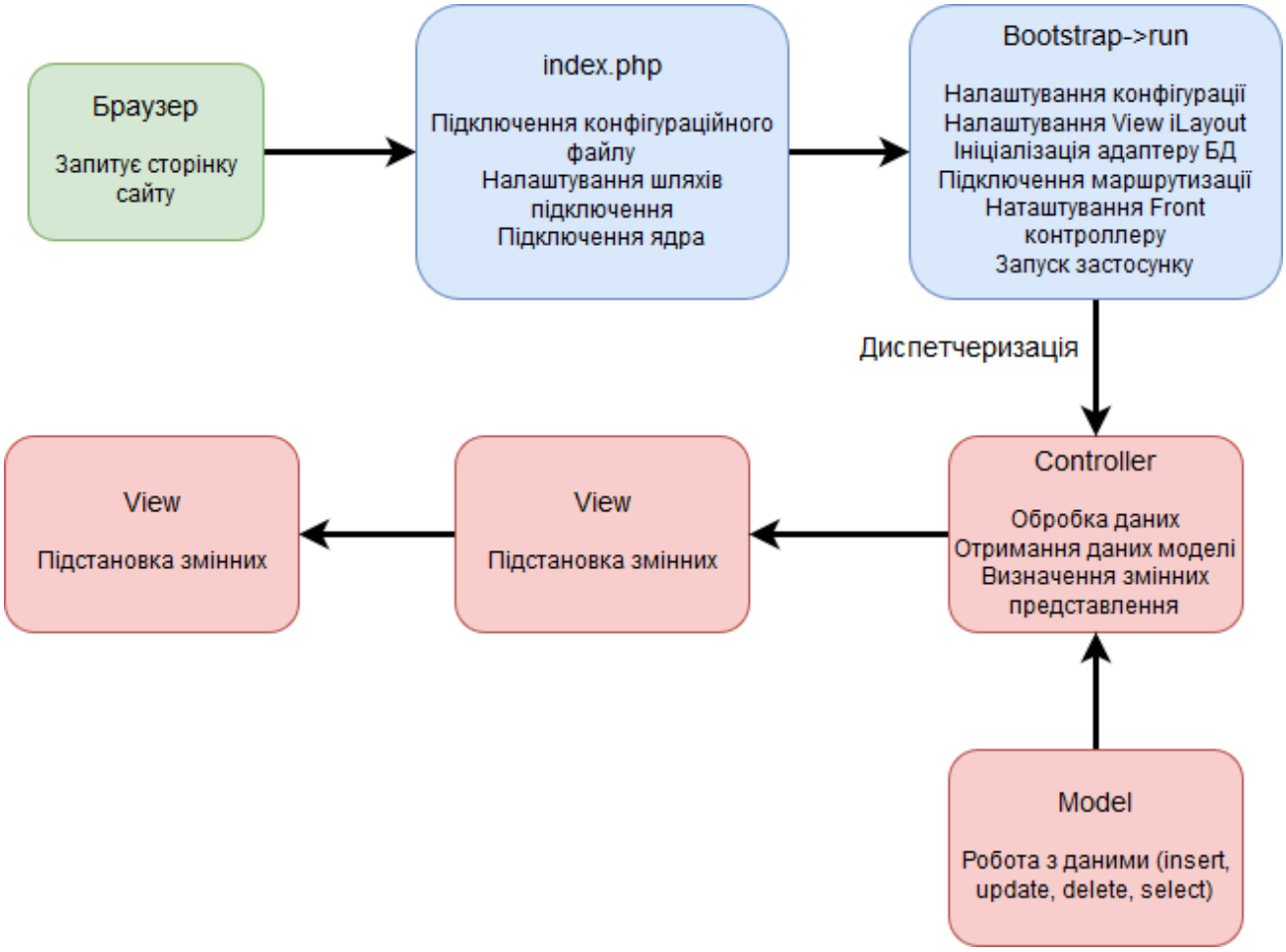

Рис. 2.2 Алгоритм процесу диспетчеризації запитів всередині веб-сайту

Нижче наведено код обробки переходу на каталог фільтрів механічного очищення:

```
<?include($_SERVER['DOCUMENT_ROOT']."/modules/main/include/prolog_
before.php");
CModule::IncludeModule("iblock");
CModule::IncludeModule("catalog");
CModule::IncludeModule("sale");
functiongetParent($id){
$tt = CIBlockSection::GetList(array(), array('ID'=>$id));
        $as=$tt->GetNext();
static $a;
if($as['DEPTH_LEVEL']==1)  $a = $as['NAME'];
else{
getParent($as['IBLOCK_SECTION_ID']);
```

```
        }
return $a; }

header("Content-Type: text/xml; charset=UTF-8");?>
<? echo "<"."?xml version=\"1.0\" encoding=\"UTF-8\"?".">"?>
<!DOCTYPEyml_catalog SYSTEM "shops.dtd">

$res = CIBlockElement::GetList(Array(), $arFilter, false);
while($ob = $res->GetNextElement()){
 $arFields = $ob->GetFields();

     $arProps = $ob->GetProperties();?>
     <?$rsPrices = CPrice::GetList(array(), array('PRODUCT_ID' =>
$arFields["ID"], 'CATALOG_GROUP_ID' => 1));
        $arPrice = $rsPrices->Fetch();
     if ($arPrice["PRICE"] >0){?>
     <offer id="<?=$arFields["ID"]?>" type="vendor.model"
available="true" <?if($arProps['bid']) echo
'bid="'.$arProps['bid']['VALUE'].'"';?><?if($arProps['cbid']) echo
'cbid="'.$arProps['cbid']['VALUE'].'"';?>>

<url>http://briz.com.ua<?=$arFields["DETAIL_PAGE_URL"]?></url>
<price>
<?{echo $arPrice["PRICE"];}?></price>
<currencyId>RUR</currencyId>
<categoryId><?=$arFields["IBLOCK_SECTION_ID"]?></categoryId>
<picture>http://briz.com.ua <?=CFile::GetPath($arFields
["DETAIL_PICTURE"])?>
</picture>
<typePrefix>Фільтри механічного очищення</typePrefix>
<vendor><?=getParent($arFields["IBLOCK_SECTION_ID"])?></vendor>
<model>
<?=$arFields["NAME"]?>Главная

<?$kr=0;
$printid=$arProps['PRINTER']['~VALUE'];
$arFilterw = Array( "IBLOCK_ID"=>"2" ,"ID"=>$printid);
$resw = CIBlockElement::GetList(Array("NAME"=>"ASC"), $arFilterw);
while($ar_res3 = $resw->GetNext()){
     echo $ar_res3["NAME"].", "; }?></model>
<description></description>
</offer>

<?}}
?>
```

У контролері відбувається підключення моделі і отримання даних, необхідних запитаній сторінці. Потім відбувається визначення змінних для виду. Нарешті, в головний макет application / views / scripts / layout.tpl

підставляються потрібні дані і весь отриманий результат відправляється в браузер.

На підставі аналізу об'єктної моделі документа для сторінок інтернет-магазину була створена реляційна база даних підсистеми маршрутизації, що працює під управлінням СУБД MySQL. Використання бази даних дозволяє економити місце на жорсткому диску, знизити витрати на запис даних, полегшити маніпулювання окремими блоками, даними і логічно розбити на частини все безліч інформації.

На рис. 2.3 представлена фізична модель бази даних підсистеми маршрутизації.

Вся необхідна інформація розміщується в таблицях бази даних в такий спосіб:

- Users - дані про зареєстрованих користувачів інтернет-магазину;
- User_groups – групи користувачів;
- Users_groups – дані про належність користувачів до груп (один користувач може належати до кількох груп одночасно)
- Amida_places_places – інформація про внутрішні та зовнішні посилання на сторінках веб-сайту;
- Amida_places_maps - інформація про банери та графічні елементи;
- Amida_places_map_place – розподіл посилань по графічним елементам;
- Amida_pages_datasource – дані про сторінки сайту;
- Amida_staticblocks – інформація про статичні елементи;
- Migrations – перелік перенаправлених посилань.

Розроблена підсистема має модульну структуру і заснована на архітектурі «клієнт-сервер», в якій клієнтом виступає браузер, а сервером - веб-сервер інтернет-магазину.

Логіка роботи підсистеми зосереджена на сервері, в той час як клієнтська робота із застосунком відбувається через Web-браузер, який входить в набір стандартних програм будь-якій операційній системи.

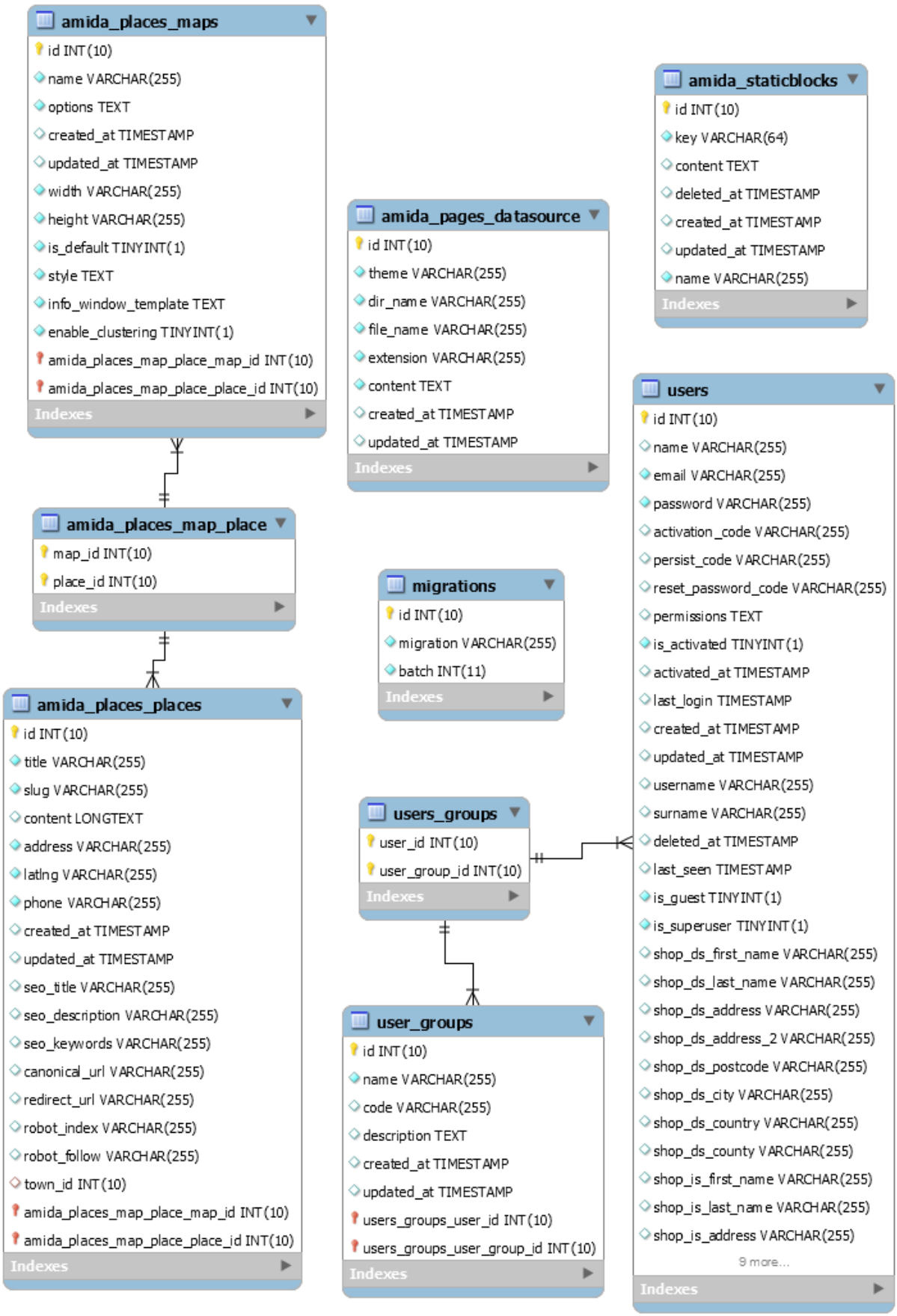
amida_places_maps
id INT(10)
name VARCHAR(255)
options TEXT
created_at TIMESTAMP
updated_at TIMESTAMP
width VARCHAR(255)
height VARCHAR(255)
is_default TINYINT(1)
style TEXT
info_window_template TEXT
enable_clustering TINYINT(1)
amida_places_map_place_map_id INT(10)
amida_places_map_place_place_id INT(10)
Indexes
amida_pages_datasource
id INT(10)
theme VARCHAR(255)
dir_name VARCHAR(255)
file_name VARCHAR(255)
extension VARCHAR(255)
content TEXT
created_at TIMESTAMP
updated_at TIMESTAMP
Indexes
amida_staticblocks
id INT(10)
key VARCHAR(64)
content TEXT
deleted_at TIMESTAMP
created_at TIMESTAMP
updated_at TIMESTAMP
name VARCHAR(255)
Indexes
amida_places_map_place
map_id INT(10)
place_id INT(10)
Indexes
migrations
id INT(10)
migration VARCHAR(255)
batch INT(11)
Indexes
users
id INT(10)
name VARCHAR(255)
email VARCHAR(255)
password VARCHAR(255)
activation_code VARCHAR(255)
persist_code VARCHAR(255)
reset_password_code VARCHAR(255)
permissions TEXT
is_activated TINYINT(1)
activated_at TIMESTAMP
last_login TIMESTAMP
created_at TIMESTAMP
updated_at TIMESTAMP
username VARCHAR(255)
surname VARCHAR(255)
deleted_at TIMESTAMP
last_seen TIMESTAMP
is_guest TINYINT(1)
is_superuser TINYINT(1)
shop_ds_first_name VARCHAR(255)
shop_ds_last_name VARCHAR(255)
shop_ds_address VARCHAR(255)
shop_ds_address_2 VARCHAR(255)
shop_ds_postcode VARCHAR(255)
shop_ds_city VARCHAR(255)
shop_ds_country VARCHAR(255)
shop_ds_county VARCHAR(255)
shop_is_first_name VARCHAR(255)
shop_is_last_name VARCHAR(255)
shop_is_address VARCHAR(255)
9 more...
Indexes
amida_places_places
id INT(10)
title VARCHAR(255)
slug VARCHAR(255)
content LONGTEXT
address VARCHAR(255)
latlng VARCHAR(255)
phone VARCHAR(255)
created_at TIMESTAMP
updated_at TIMESTAMP
seo_title VARCHAR(255)
seo_description VARCHAR(255)
seo_keywords VARCHAR(255)
canonical_url VARCHAR(255)
redirect_url VARCHAR(255)
robot_index VARCHAR(255)
robot_follow VARCHAR(255)
town_id INT(10)
amida_places_map_place_map_id INT(10)
amida_places_map_place_place_id INT(10)
Indexes
users_groups
user_id INT(10)
user_group_id INT(10)
Indexes
user_groups
id INT(10)
name VARCHAR(255)
code VARCHAR(255)
description TEXT
created_at TIMESTAMP
updated_at TIMESTAMP
users_groups_user_id INT(10)
users_groups_user_group_id INT(10)
Indexes

Рис.2.3. Фізична модель бази даних

Відповідно до принципів об'єктно-орієнтованого програмування, код застосунку розділений по смисловим ознаками в окремі класи. Оскільки веб-застосунок інтернет-магазину створено на основі програмного каркаса Zend Framework, його файлова система копіює структуру файлової системи фреймворка. На рис. 2.4 відображений список файлів і папок веб-сайту.

Веб-застосунок інтернет-магазину реалізовано у вигляді сукупності взаємопов'язаних файлів, що містять код PHP, в кожному з яких описується певний функціонал внутрішнього або зовнішнього поведінки системи. Файли з програмним кодом модулів підсистеми маршрутизації запитів знаходяться на жорсткому диску сервера за адресою \\ ... \ home \ local \ host \ www \ briz, в папці під назвою Controller.

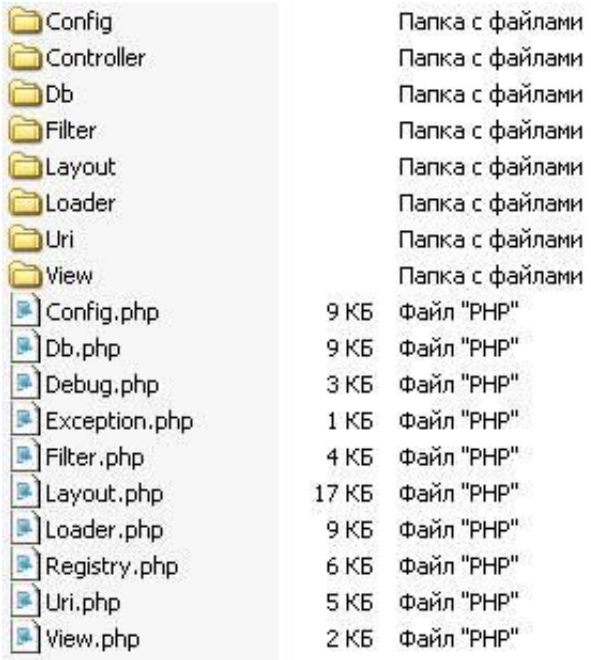

Рис. 2.4. Файлова структура веб-сайту

2.5. Обгрунтування та організація вхідних та вихідних даних програми

Програмне забезпечення отримує вхідні дані шляхом зчитування інформації з адресного рядка браузера.

Вхідні дані:

- URL в адресному рядку браузера.

Вихідні дані:

- веб-сторінки сайту з продажу водоочисного обладнання.

2.6. Опис роботи розробленої системи

2.6.1. Використані технічні засоби

Для серверних технічних засобів рекомендована конфігурація, що забезпечує цілодобову роботу програми з резервуванням даних:

- ✓ процесор класу Intel ® Xeon з тактовою частотою 2.4GHz;
- ✓ шина даних - 1066MHz,
- ✓ кеш другого рівня - 2048 КБ;
- ✓ оперативна пам’ять 2 x DIMM DDR2-800 1024 Мб;
- ✓ жорсткі диски 3x 250 Гб SATA 2 16 Мб буфер, 7200 RPM;
- ✓ рідкокристалічний монітор з діагоналлю не менше 17 ";
- ✓ доступ до мережі Internet;
- ✓ клавіатура;
- ✓ маніпулятор "миша".

Наведені вище технічні характеристики є рекомендованими, тобто при наявності технічних засобів не нижче зазначених, розроблений програмний виріб буде функціонувати відповідно до вимог щодо надійності, швидкості обробки даних і безпеки, висунутими замовником.

2.6.2. Використані програмні засоби

Проект реалізований на мові програмування PHP 5, в якості СУБД була обрана MySQL - реляційна СУБД з вільною ліцензією. Також для коректної

роботи розробленого застосунку необхідний довільний веб-браузер з підтримкою технології JavaScript.

Необхідні програмні засоби:

Для клієнтського комп'ютера:

- ✓ ОС сімейства Windows (XP, Vista, 7,8, 10), Linux;
- ✓ веб-браузер Firefox / Google Chrome / Opera / Internet Explorer.

Для сервера:

- ✓ серверна ОС сімейства Microsoft Windows, Linux, FreeBSD;
- ✓ СУБД MySQL;
- ✓ мова програмування PHP 5.

2.6.3. Виклик та завантаження програми

Розроблена підсистема маршрутизації запитів використовується в складі інтернет-магазину Briz, завантаження якого виконується за допомогою веб-браузера з підтримкою JavaScript. Для виклику підсистеми в рядку запиту веб-браузера необхідно ввести http: // www.briz.com.ua/.

2.6.4. Опис інтерфейсу користувача

Для початку роботи з підсистемою маршрутизації необхідно увійти на головну сторінку сайту інтернет-магазину Briz (рис.2.5). У верхньому правому кутку сторінки знаходится логотип магазину та контактні дані, ліву та центральну частини хедеру займає головне меню, що містить такі пункти: «Продукція», «Про компанію», «Партнерам», «Блог», та перемикач локалізації. Основну частину головної строінки займають слайдер промо-зображень, опис найбільш популярних груп товару (рис.2.6), та новинки магазину (рис.2.7).

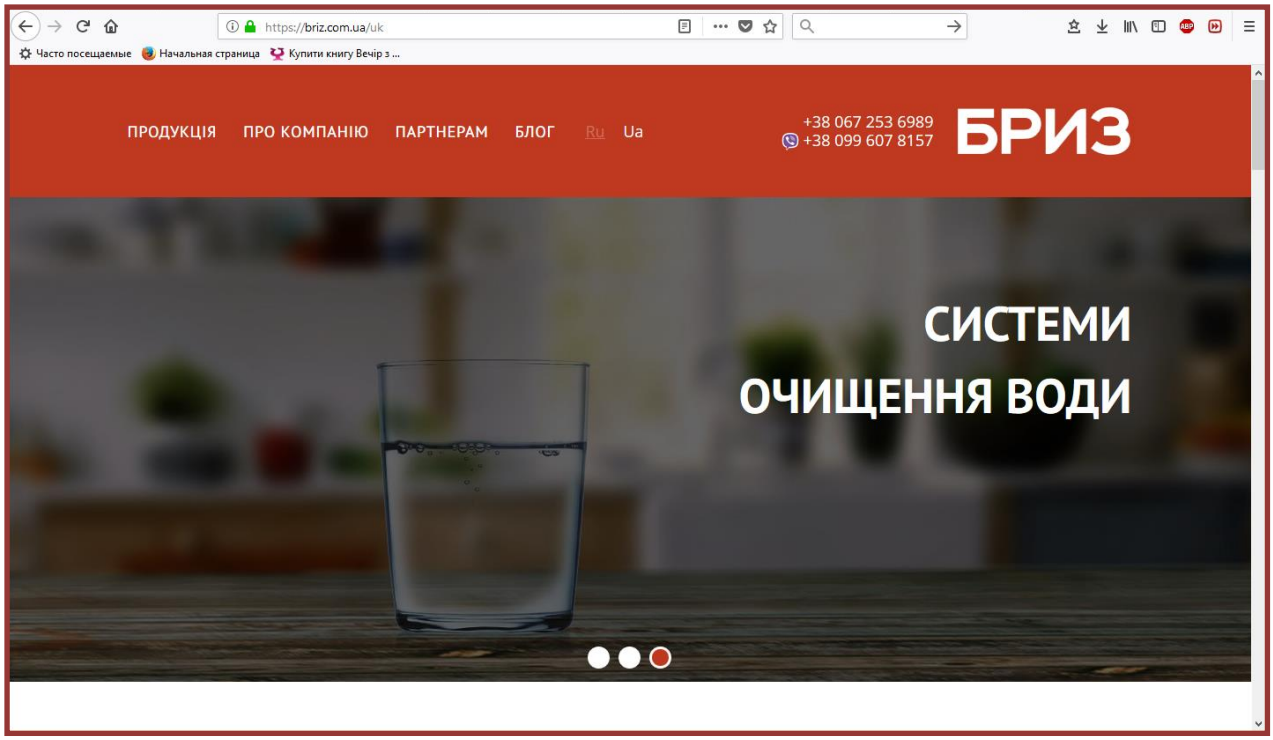

Рис. 2.5. Головна сторінка інтернет-магазину Briz

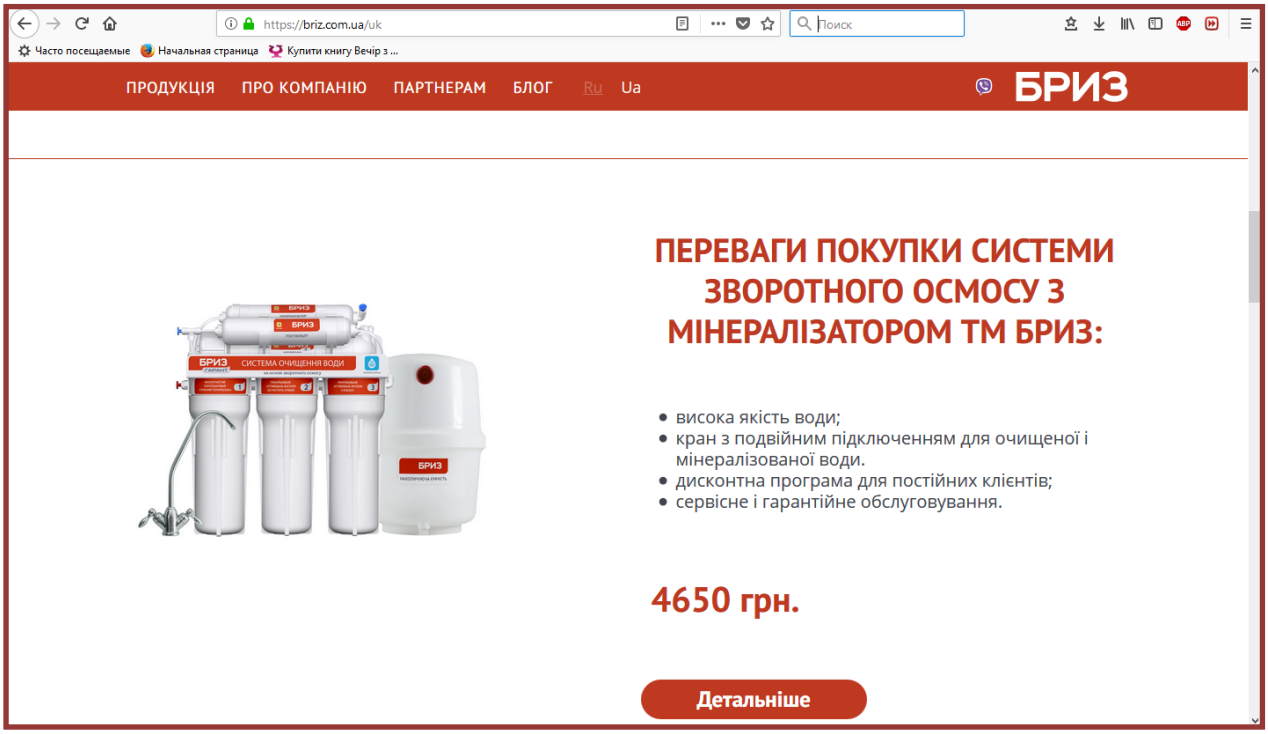

Рис. 2.6. Реклама популярного товару

Каталог продукції магазину «Бриз» доступний за посиланням «Продукція» у головному меню сайту. Після переходу до цього розділу на сторінці відображається плитковий каталог груп товару (рис.2.8).

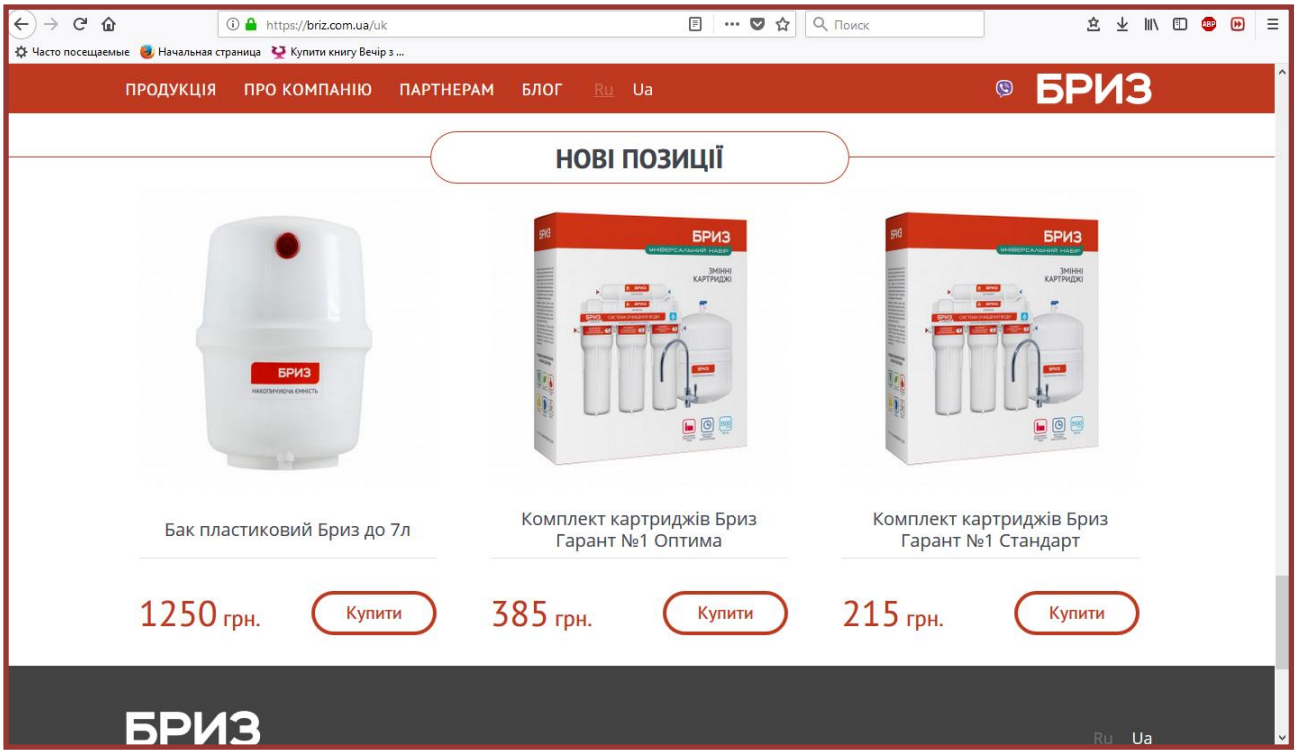

Рис. 2.7. Останні надходження товару

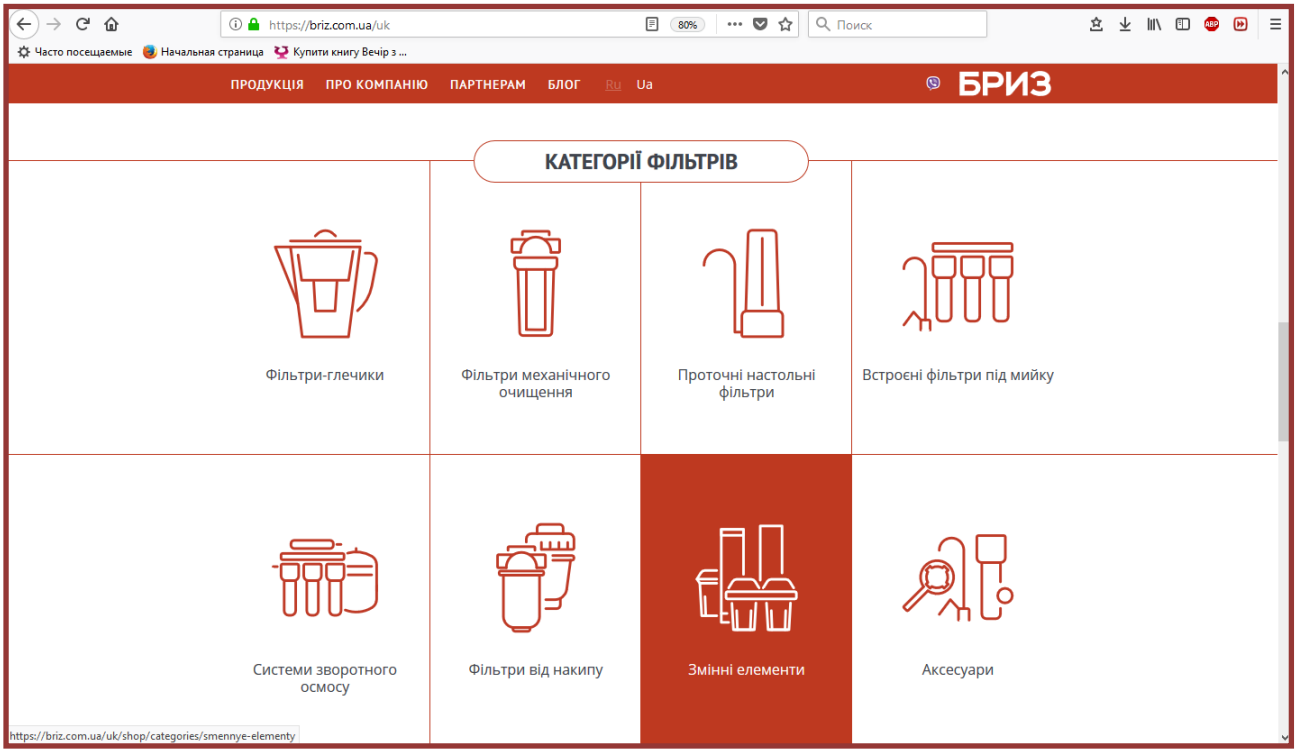

Рис. 2.8. Каталог за групами продукції

Перехід до каталогу конкретної групи товару відбувається шляхом натиснення на відповідний єлемент графічного меню. Розділ з переліком

товарних позицій обраної групи містить зображення продукції, її найменування, ціну, та кнопку «Купити», призначену для додавання товару до кошика (рис.2.9). Для переходу на сторінку товару необхідно клікнути курсором по його назві чи зображенню.

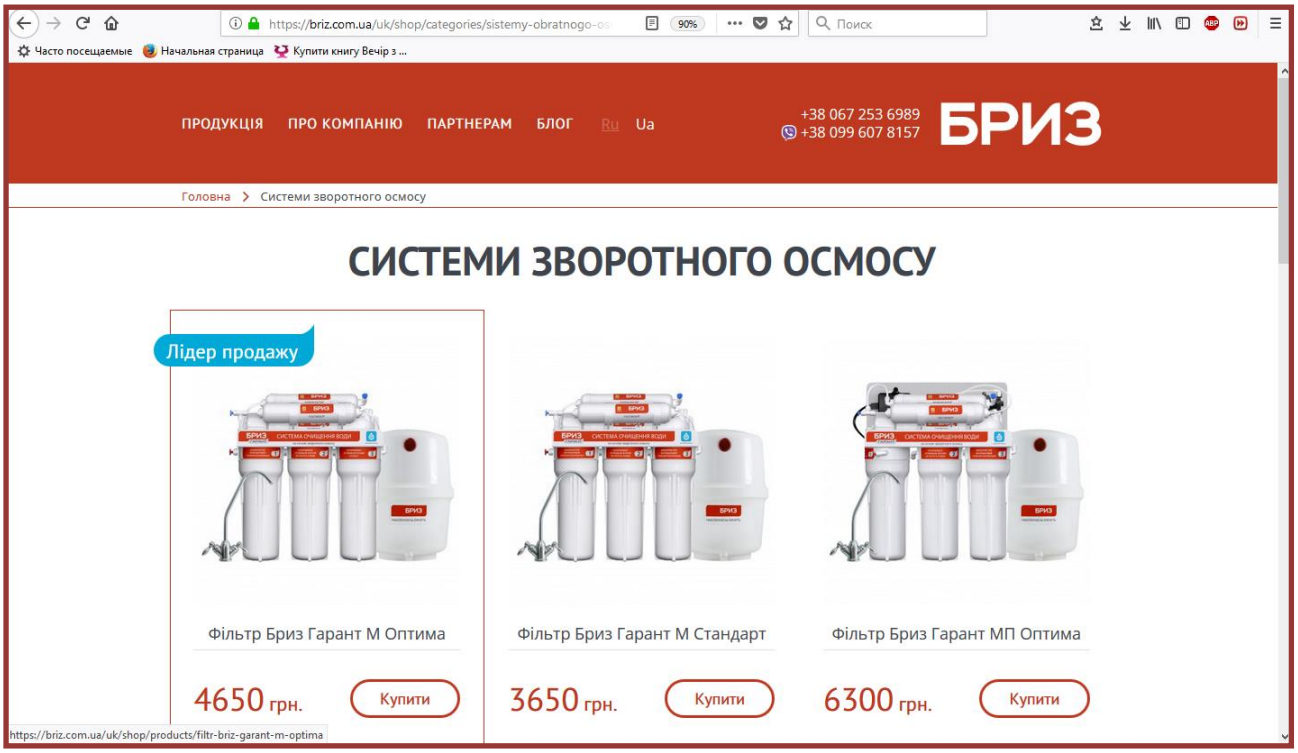

Рис. 2.9. Каталог товарних позицій

Сторінка товару містить детальний опис, ціну, галерею зображень товару, перелік його властивостей, та кнопку «Купити» (рис.2.10). За посиланням «Детальніше» користувач переходить до розгорнутого опису технічних характеристик виробу (рис.2.11).

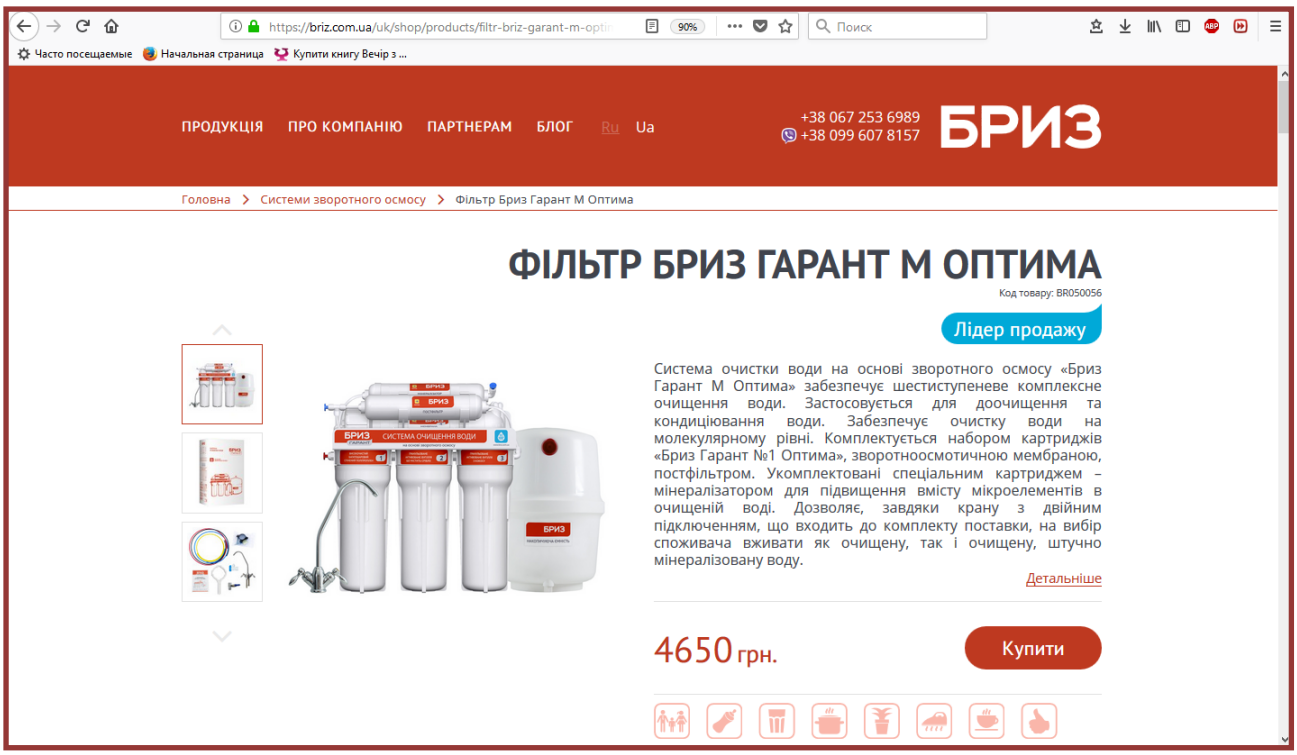

Рис. 2.10. Сторінка товару

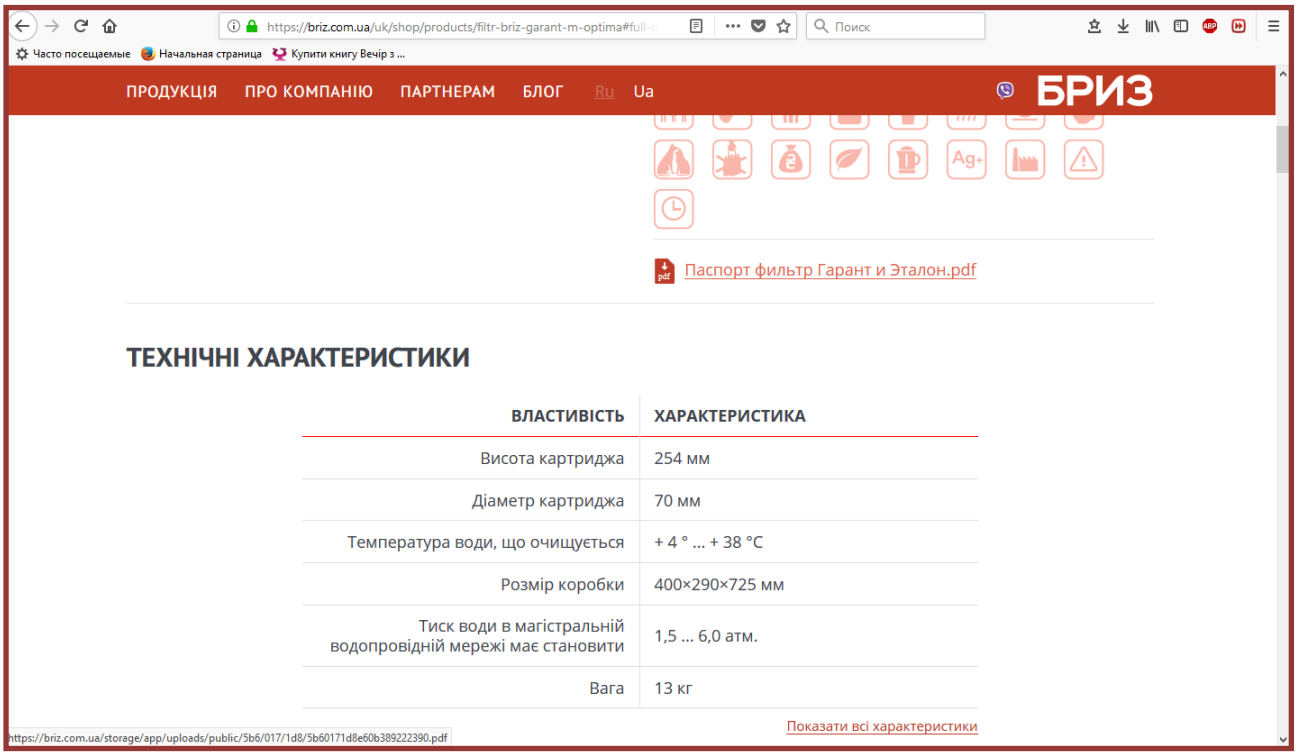

Рис. 2.11. Детальна інформація про товар

Завершує сторінку товару перелік супутніх пропозицій магазину, що доповнюють комплект обладнання або використовуються як витратні матеріали до нього (рис. 2. 12).

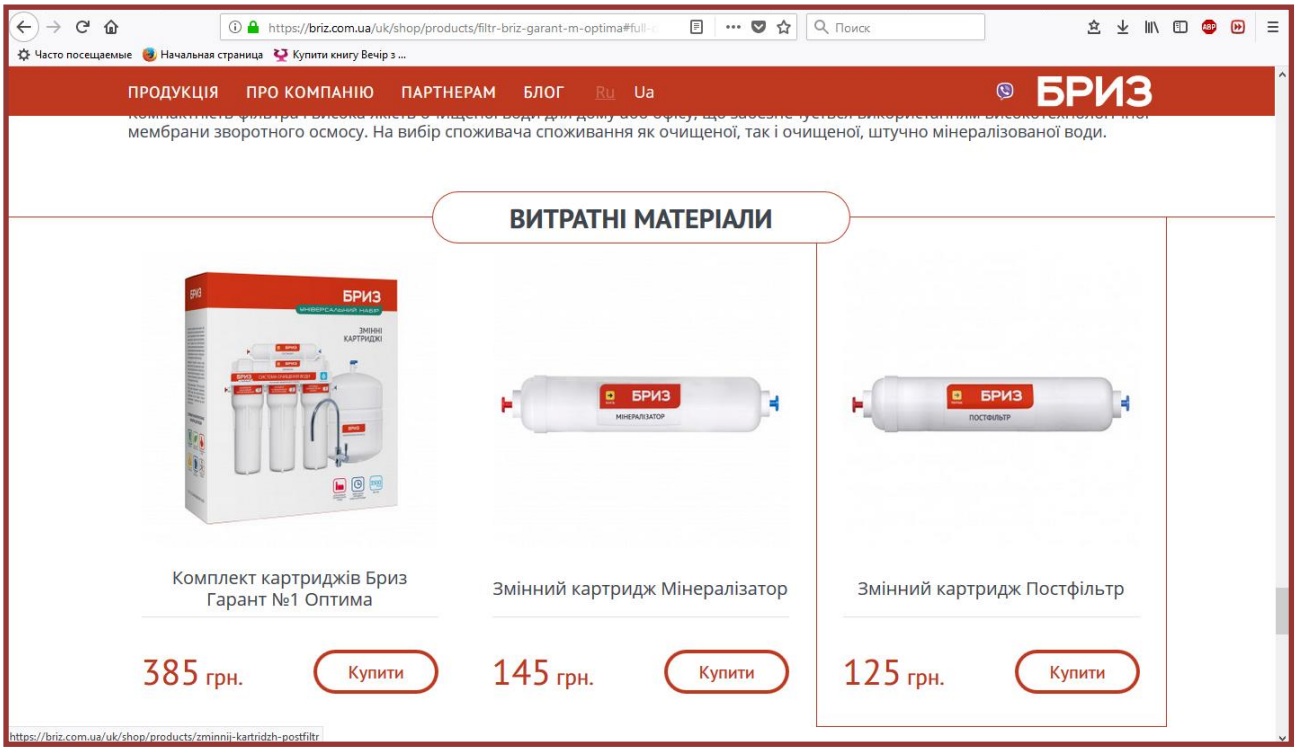

Рис. 2.12. Перелік супутніх пропозицій

Розділ сайту «Блог», до якого можна перейти через відповідний пункт головного меню, містить два підрозділи – «Новини» та «Статті». На сторінці новини публікується актуальна інформація про події компанії, здебільш у стислому вигляді (рис. 2.13).

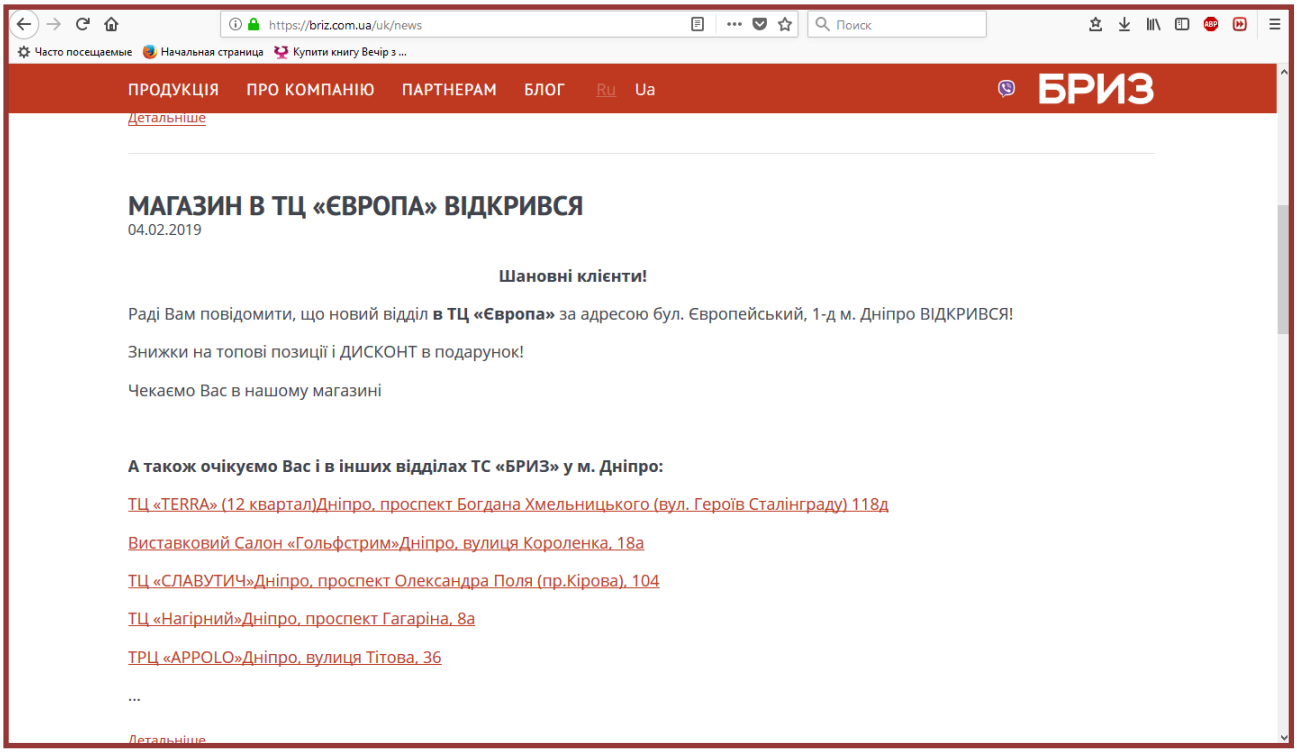

Рис. 2.13. Новини

У розділі «Статті» розміщена корисна інформація щодо вибору та експлуатації водоочисного обладнання, специфікації та відео-інструкції (рис. 2.14).

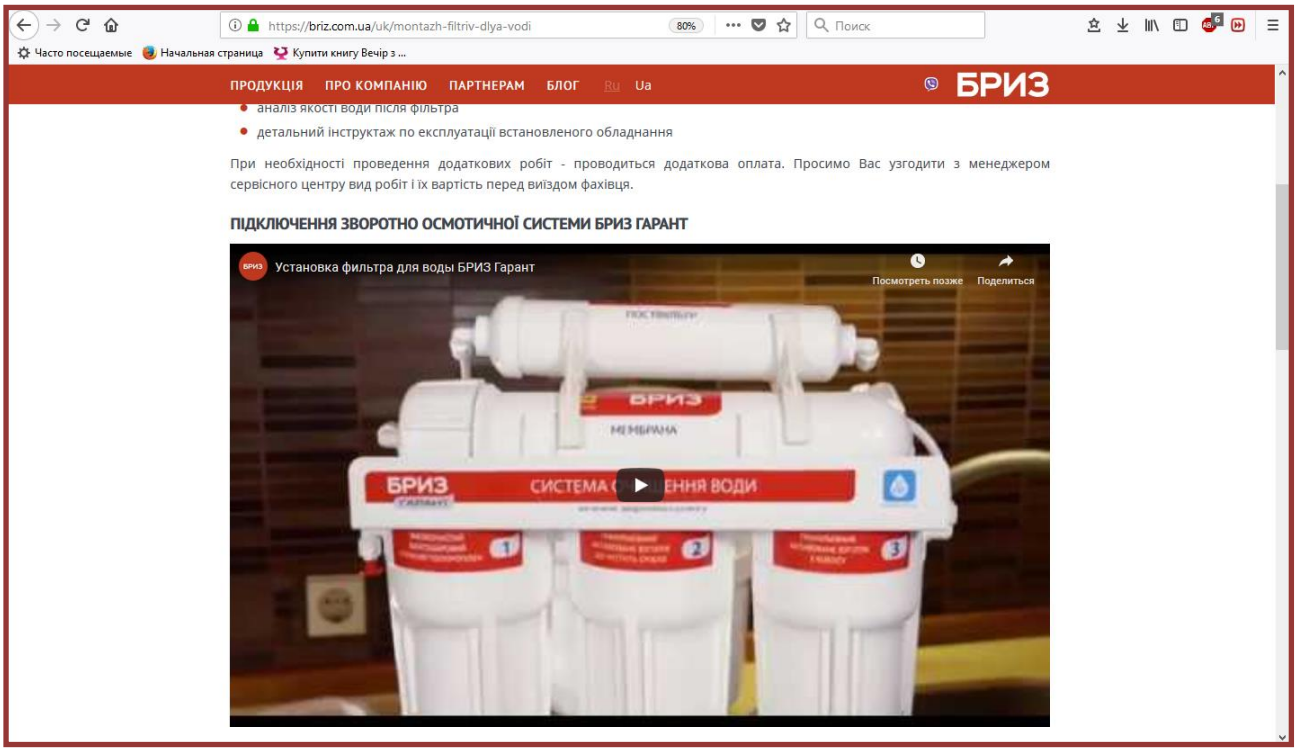

Рис. 2.14. Відео-інструкція з експлуатації товару

Щоб додати товар до кошика, необхідно натиснути на кнопку «Купити», розташовану на сторінці товару або у каталозі. На сторінці «Кошик» знаходиться перелік обраних покупцем товарів, включаючи їх найменування, ціну, кількість одиниць кожної позиції і загальна вартість покупки (рис.2.15).

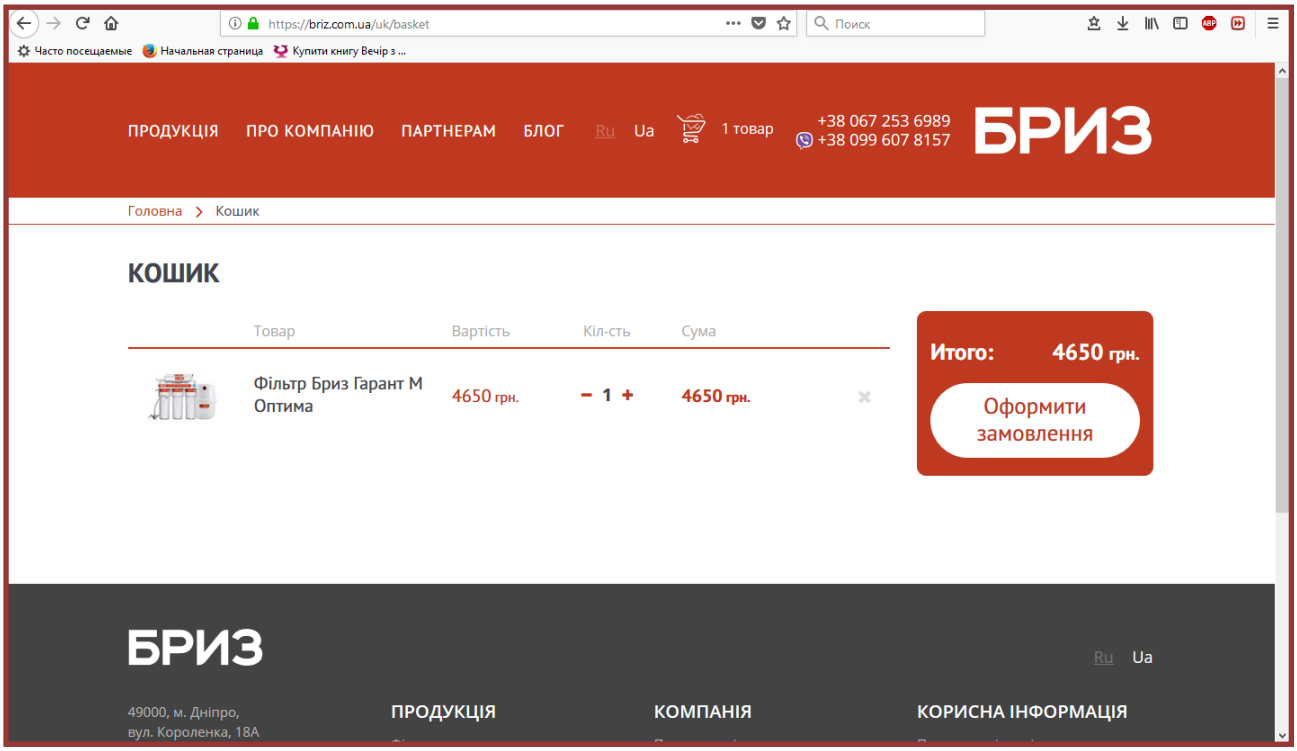

Рис. 2.15. Кошик

Кнопка «Оформити замовлення», розташована у правій частині сторінки кошику, перенаправлює користувача на сторінку оформлення замовлення (рис. 2.16).

Процедура оформлення замовлення включає, в першу чергу, внесення паспортних даних, контактного телефону та адреси покупця.

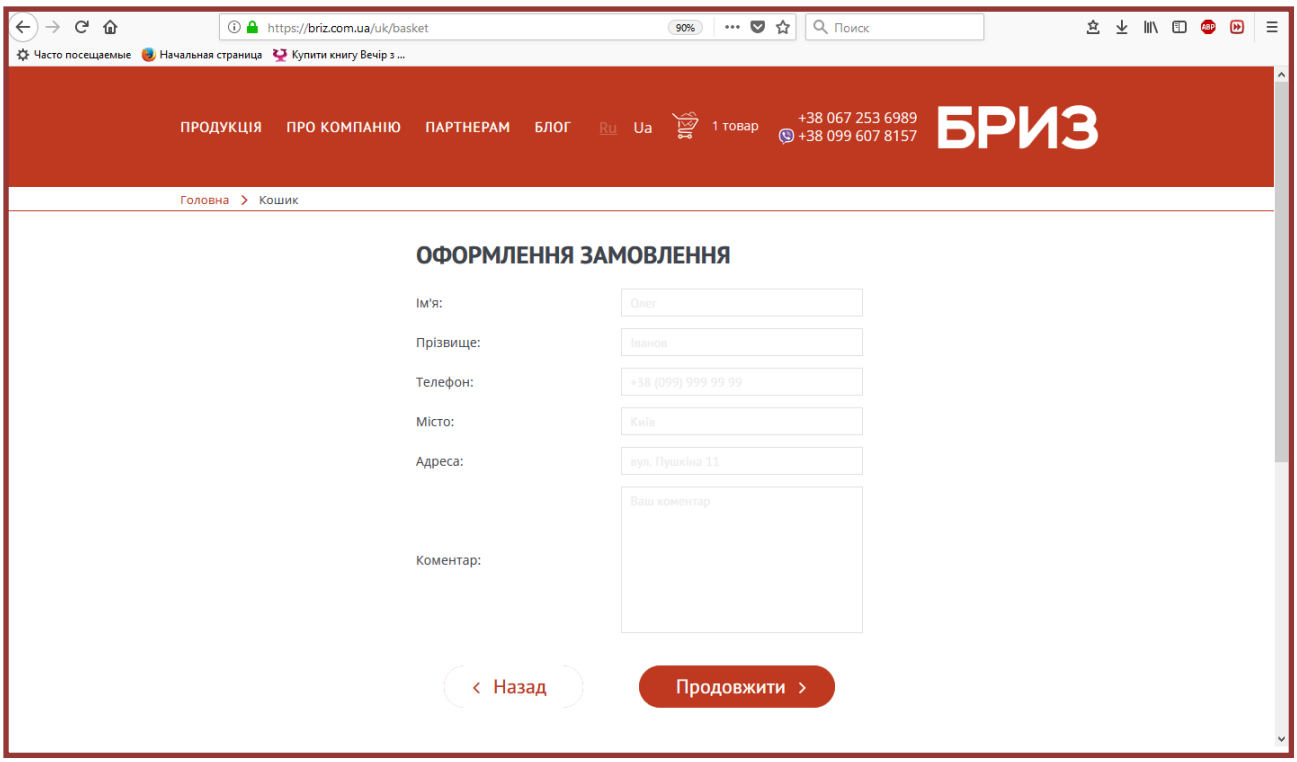

Рис. 2.16. Оформлення замовлення

Потім користувачеві необхідно вказати спосіб доставки і оплати замовлення, скориставшись запропонованим переліком варіантів надання даних послуг. Для переміщення між сторінками замовлення використовуються навігаційні кнопки «Назад» та «Продовжити».

РОЗДІЛ 3
ЕКОНОМІКА

3.1. Розрахунок вартості розробки програмного продукту

Початкові дані:

1. передбачуване число операторів програми – 1200;
2. коефіцієнт складності програми – 1,6;
3. коефіцієнт корекції програми в ході її розробки – 0,05;
4. годинна заробітна плата програміста – 90 грн/год;
5. коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,2;
6. коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1,2;
7. вартість машино-години ЕОМ – 13 грн/год.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д, \text{ людино-годин,} \quad (3.1)$$

де t_o - витрати праці на підготовку й опис поставленої задачі (приймається 50 людино-годин);

t_u - витрати праці на дослідження алгоритму рішення задачі;

t_a - витрати праці на розробку блок-схеми алгоритму;

t_n - витрати праці на програмування по готовій блок-схемі;

$t_{отл}$ - витрати праці на налагодження програми на ЕОМ;

$t_д$ - витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у програмногму забезпеченні, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \cdot C \cdot (1 + p),$$

де q - передбачуване число операторів (1200);

C - коефіцієнт складності програми (1,6);

p - коефіцієнт корекції програми в ході її розробки (0,05).

Звідси умовне число операторів в програмі:

$$Q = 1{,}6 \cdot 1200 \cdot (1 + 0{,}05) = 2016$$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \cdot B}{(75..85) \cdot k}\ \text{, людино-годин,}$$

де B - коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

k - коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності. При стажі роботи від 3 до 5 років він складає 1,2.

Приймемо збільшення витрат праці внаслідок недостатнього опису завдання не більше 50% (B = 1.2). З урахуванням коефіцієнта кваліфікації k = 1,2, отримуємо витрати праці на вивчення опису завдання:

$$t_u = (2016 \cdot 1{,}2) / (75 \cdot 1{,}2) = 26{,}88 \text{ людино-годин}$$

Витрати праці на розробку алгоритму рішення задачі визначаються за формулою:

$$t_a = \frac{Q}{(20\ldots25)\cdot k}\ \text{, людино-годин,}\quad (3.2)$$

де Q – умовне число операторів програми;

k – коефіцієнт кваліфікації програміста.

Підставивши відповідні значення в формулу (3.2), отримаємо:

$$t_a = 2016 / (20 \cdot 1{,}2) = 84 \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20\ldots25)\cdot k}\ \text{, людино-годин.}$$

$$t_n = 2016 / (25 \cdot 1{,}2) = 67{,}2 \text{ людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

- за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4..5)\cdot k}\ \text{, людино-годин.}$$

$$t_{отл} = 2016 / (5 \cdot 1{,}2) = 336 \text{ чел.-ч.}$$

- за умови комплексного налагодження завдання:

$$t_{отл}^{k} = 1{,}5 \cdot t_{отл}\ \text{, людино-годин.}$$

$$t^{k}_{отл} = 1,5 \cdot 336 = 504 \text{ людино-годин.}$$

Витрати праці на підготовку документації визначаються за формулою:

$$t_{д} = t_{др} + t_{до} \text{ , людино-годин,}$$

де $t_{др}$ - трудомісткість підготовки матеріалів і рукопису:

$$t_{др} = \frac{Q}{(15..20) \cdot k} \text{ , людино-годин,}$$

$t_{до}$ - трудомісткість редагування, печатки й оформлення документації:

$$t_{до} = 0,75 \cdot t_{др} \text{ , людино-годин.}$$

Підставляючи відповідні значення, отримаємо:

$$t_{др} = 2016 / (18 \cdot 1,2) = 93,33 \text{ людино-годин.}$$

$$t_{до} = 0,75 \cdot 93,33 = 70 \text{ людино-годин.}$$

$$t_{д} = 93,33 + 70 = 163,33 \text{ людино-годин.}$$

Повертаючись до формули (3.1), отримаємо повну оцінку трудомісткості розробки програмного забезпечення:

$$t = 50 + 26,88 + 84 + 67,2 + 336 + 163,33 = 727,41 \text{ людино-годин.}$$

3.2. Рахунок витрат на створення програми

Витрати на створення ПЗ $К_{ПО}$ включають витрати на заробітну плату виконавця програми $З_{ЗП}$ і витрат машинного часу, необхідного на налагодження програми на ЕОМ:

$$К_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн.}$$

Заробітна плата виконавців визначається за формулою:

$$З_{ЗП} = t \cdot С_{ПР}, \text{ грн,}$$

де: t - загальна трудомісткість, людино-годин;

$С_{ПР}$ - середня годинна заробітна плата програміста, грн/година

З урахуванням того, що середня годинна зарплата програміста становить 60 грн / год, отримуємо:

$$З_{ЗП} = 727{,}41 \cdot 90 = 44\,644{,}6 \text{ грн.}$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ, визначається за формулою:

$$З_{мв} = t_{отл} \cdot С_{мч}, \text{ грн,} \quad (3.3)$$

де $t_{отл}$ - трудомісткість налагодження програми на ЕОМ, год;

$С_{мч}$ - вартість машино-години ЕОМ, грн/год (13 грн/год).

Підставивши в формулу (3.3) відповідні значення, визначимо вартість необхідного для налагодження машинного часу:

$$З_{мв} = 336 \cdot 13 = 4368 \text{ грн.}$$

Звідси витрати на створення програмного продукту:

$$К_{ПО} = 44644{,}6 + 4368 = 49\,012{,}6 \text{ грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p}, \text{ міс.}$$

де B_k - число виконавців (дорівнює 1);

F_p - місячний фонд робочого часу (при 40 годинному робочому тижні F_p=176 годин).

Звідси витрати на створення програмного продукту:

$$T = 727{,}41 / 1 \cdot 176 \approx 4 \text{ міс.}$$

Висновок: Програмне забезпечення призначене для забезпечення швидкої та точної реакції інтерфейсу веб-сайту з продажу водоочисного обладнання на дії користувачів. Вартість даного програмного забезпечення становить 49 тис. грн. і не вимагає додаткових витрат як при розробці програми. Очікуваний час розробки становить 4 місяці. Цей термін пов'язаний зі значним числом операторів, і включає час на дослідження і розробку алгоритму вирішення поставленого завдання, програмування по готовому алгоритму, налагодження програми і підготовку документації.

ВИСНОВКИ

В даній кваліфікаційній роботі була розроблена підсистема маршрутизації для інтерактивного веб-сайту з продажу водоочисного обладнання.

Це програмне забезпечення призначене для централізованої обробки запитів до компонентів і сторінок інтерактивного веб-сайту. Практичне призначення даної підсистеми полягає в забезпеченні швидкої та точної реакції інтерфейсу веб-сайту на дії користувачів, що зробить процес покупки швидшим і зручнішим, і підвищить ефективність роботи інтернет-магазину.

Під час виконання даної кваліфікаційної роботи були виконані наступні задачі:

- вивчено предметну галузь розв'язуваної задачі;
- створено алгоритм для реалізації поставленого завдання;
- створено базу даних і клієнтську програму, що працює з нею.

Розроблене програмне забезпечення дозволяє:

- виконувати класифікацію і парсинг вхідного URL;
- викликати необхідні методи дочірніх контролерів в залежності від атрибутів URL;
- здійснювати переадресацію користувальницького запиту на відповідні сторінки в разі некоректного URL.

Програма реалізована на базі фреймворка Zend з використанням мови програмування PHP і СУБД MySQL.

Також у кваліфікаційній роботі було визначено трудомісткість розробленої підсистеми 727 люд-год, проведений підрахунок вартості роботи по створенню програми 49012 грн та розраховано час на його створення 4 міс.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бібліографічний запис. Бібліографічний опис. Загальні вимоги та правила складання : (ГОСТ 7.1-2003, IDT) : ДСТУ ГОСТ 7.1:2006. – Чинний з 2007–07–01. – К. : Держспоживстандарт України, 2007. – 47 с. – (Система стандартів з інформації, бібліотечної та видавничої справи) (Національний стандарт України).
2. Бибо Б. jQuery. Подробное руководство по продвинутому JavaScript / Б. Бибо, И. Кац. — Спб.: Символ-Плюс, 2009. — 384 с.
3. Вертайм К. Цифровой маркетинг. Как увеличить продажи с помощью социальных сетей, блогов, вики-ресурсов, мобильных телефонов и других современных технологий / К. Вертайм, Я. Фенвик. — Альпина Паблишер, 2010. - 384 с.
4. Гарсиа-Молина Г. Системы баз данных. Полный курс / Г. Гарсиа-Молина, Дж. Ульман, Дж. Уидом. — М.: Вильямс, 2003. – 1088 с.
5. Грабер М. Введение в SQL. — М.: Лори, 1996. — 479 с.
6. Дейт К. Дж. Введение в системы баз данных. 6-е изд. – СПб.: Вильямс, 2000. – 848с.
7. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення : ДСТУ 3008-95. – Чинний від 1996–01–01. – К. : Держстандарт України, 1996. – 39 с.
8. ДСТУ 2394-94 Інформація та документація. Комплектування фонду, бібліографічний опис, аналіз документів. Терміни та визначення. – Чинний від 01.01.1995. — Київ : Держстандарт України, 1994. – 88 с.
9. Зандстра М. PHP. Объекты, шаблоны и методики программирования – М.: Apress, 2015. – 576 с.
10. Колисниченко Д.Н. PHP и MySQL. Разработка Web-приложений. / Д.Н. Колисниченко. – СПб.: Питер, 2015. – 593 с.

11. Коннолли Т. Базы данных. Проектирование, реализация и сопровождение. Теория и практика. 3-е издание: Пер. с англ. / Т. Коннолли, К. Бегг - М.: Вильямс, 2003. — 1440 с.

12. Кривий С. Вступ до методів творення програмних продуктів. – Чернівці: Букрек, 2012. – 424 с.

13. Леки-Томпсон Э. PHP 5 для профессионалов / Э. Леки-Томпсон, А. Коув, С. Новицки, Х. Айде-Гудман. – М.: Диалектика, 2006. – 114 с.

14. Ленгсторф Дж.PHP и jQuery для профессионалов / Дж. Ленгсторф. — М.: Вильямс, 2010. — 352 с.

15. Локхарт Дж. Современный PHP. Новые возможности и передовой опыт. – М.: Символ, 2016. – 306 с.

16. Мальчук Е.В. HTML и CSS. Самоучитель / Е. В. Мальчук – М.: Вильямс, 2008. – 416 с.

17. Методичні вказівки з виконання економічного розділу в дипломних проектах студентів спеціальності «Комп'ютерні системи» / О.Г. Вагонова, О.Б. Нікітіна, Н.Н. Романюк; М-во освіти і науки України, ДВНЗ «Нац. гірн. ун-т». – Д.: НГУ, 2013. – 11 с.

18. Методичні рекомендації до виконання кваліфікаційних робіт бакалаврів напряму підготовки 6.050101 «Комп'ютерні науки / І.М. Удовик, Л.М. Коротенко, О.С. Шевцова; Нац. гірн. ун-т. – Д : НТУ «Дніпровська політехніка», 2018. – 65 с.

19. Никсон Р. Создаем динамические сайты с помощью PHP, MySql, JavaScript, CSS и HTML 5. – СПб.: Питер, 2016. – 768 с.

20. Перевозчикова О. Інформаційні системи і структури даних. – К.: Києво-Могилянська академія, 2007. – 288 с.

21. Пьюривал С. Основы разработки веб-приложений / С. Пьюривал. — СПб.: Питер, 2015. — 272 с.

22. Симдянов И. PHP 7. В подлиннике. / И. Симдянов, Д. Котеров. – СПб.: Питер, 2016. – 1073 с.

23. Скляр Д. Изучаем PHP 7. Руководство по созданию интерактивных веб-сайтов / Д. Скляр. — СПб.: Питер, 2017. — 382 с.

24. Скорочення слів в українській мові у бібліографічному описі. Загальні правила та вимоги : ДСТУ 3582-97. – Чинний від 1998–07–01. – К. : Держстандарт України, 1998. – 24 с. – (Державний стандарт України).

25. Скотт Б. Проектирование веб-интерфейсов / Б. Скотт, Т. Нейл. — Спб.: Символ-Плюс, 2010. — 352 с.

26. Столлингс В. Передача данных/ В. Столлингс. - СПб.: Питер, 2004. – 750 с.

27. Хеник Б. HTML и CSS. Путь к совершенству / Б. Хеник – СПб.: Питер, 2011. – 336 с.

28. Шварц Б. MySQL. Оптимизация производительности / Б. Шварц, П. Зайцев, В. Ткаченко. – М.: Символ, 2010. – 823 с.

29. Шкрыль А. А. Программируем для web-сайта / А.А. Шкрыль – М.: Диалектика, 2006. – 368с: ил.

ДОДАТОК А

КОД ПРОГРАМИ

CombineAssets.php // головний клас програми

```
<?php namespace System\Classes;

use App;
use Url;
use File;
use Lang;
use Event;
use Cache;
use Route;
use Config;
use Request;
use Response;
use Assetic\Asset\FileAsset;
use Assetic\Asset\GlobAsset;
use Assetic\Asset\AssetCache;
use Assetic\Asset\AssetCollection;
use Assetic\Factory\AssetFactory;
use October\Rain\Parse\Assetic\FilesystemCache;
use System\Helpers\Cache as CacheHelper;
use ApplicationException;
use DateTime;

class CombineAssets
{
    use \October\Rain\Support\Traits\Singleton;

    protected static $jsExtensions = ['js'];

    protected static $cssExtensions = ['css', 'less', 'scss', 'sass'];

    protected $aliases = [];

    protected $bundles = [];

    protected $filters = [];

    protected $localPath;

    protected $storagePath;

    public $useCache = false;

    public $useMinify = false;

    public $useDeepHashing = false;
```

```
private static $callbacks = [];

public function init()
{
    $this->useCache = Config::get('cms.enableAssetCache', false);
    $this->useMinify = Config::get('cms.enableAssetMinify', null);
    $this->useDeepHashing = Config::get('cms.enableAssetDeepHashing', null);

    if ($this->useMinify === null) {
        $this->useMinify = !Config::get('app.debug', false);
    }

    if ($this->useDeepHashing === null) {
        $this->useDeepHashing = Config::get('app.debug', false);
    }

    $this->registerFilter('js', new \October\Rain\Parse\Assetic\JavascriptImporter);

    $this->registerFilter('css', new \Assetic\Filter\CssImportFilter);
    $this->registerFilter(['css', 'less', 'scss'], new \Assetic\Filter\CssRewriteFilter);
    $this->registerFilter('less', new \October\Rain\Parse\Assetic\LessCompiler);
    $this->registerFilter('scss', new \October\Rain\Parse\Assetic\ScssCompiler);

    if ($this->useMinify) {
        $this->registerFilter('js', new \Assetic\Filter\JSMinFilter);
        $this->registerFilter(['css', 'less', 'scss'], new \October\Rain\Parse\Assetic\StylesheetMinify);
    }


    $this->registerAlias('jquery', '~/modules/backend/assets/js/vendor/jquery.min.js');
    $this->registerAlias('framework', '~/modules/system/assets/js/framework.js');
    $this->registerAlias('framework.extras', '~/modules/system/assets/js/framework.extras.js');
    $this->registerAlias('framework.extras', '~/modules/system/assets/css/framework.extras.css');

    /*
     * Deferred registration
     */
    foreach (static::$callbacks as $callback) {
        $callback($this);
    }
}

public static function combine($assets = [], $localPath = null)
{
    return self::instance()->prepareRequest($assets, $localPath);
}

public function combineToFile($assets = [], $destination, $localPath = null)
{
    // Disable cache always
    $this->storagePath = null;
```

```
        list($assets, $extension) = $this->prepareAssets($assets);

        $rewritePath = File::localToPublic(dirname($destination));
        $combiner = $this->prepareCombiner($assets, $rewritePath);

        $contents = $combiner->dump();
        File::put($destination, $contents);
    }

    public function getContents($cacheKey)
    {
        $cacheInfo = $this->getCache($cacheKey);
        if (!$cacheInfo) {
            throw new ApplicationException(Lang::get('system::lang.combiner.not_found',
['name'=>$cacheKey]));
        }

        $this->localPath = $cacheInfo['path'];
        $this->storagePath = storage_path('cms/combiner/assets');

        $lastModifiedTime = gmdate("D, d M Y H:i:s \G\M\T", array_get($cacheInfo, 'lastMod'));
        $etag = array_get($cacheInfo, 'etag');
        $mime = (array_get($cacheInfo, 'extension') == 'css')
            ? 'text/css'
            : 'application/javascript';

        header_remove();
        $response = Response::make();
        $response->header('Content-Type', $mime);
        $response->setLastModified(new DateTime($lastModifiedTime));
        $response->setEtag($etag);
        $response->setPublic();
        $modified = !$response->isNotModified(App::make('request'));

        if ($modified) {
            $this->setHashOnCombinerFilters($cacheKey);
            $combiner = $this->prepareCombiner($cacheInfo['files']);
            $contents = $combiner->dump();
            $response->setContent($contents);
        }

        return $response;
    }

    protected function prepareAssets(array $assets)
    {
        if (!is_array($assets)) {
            $assets = [$assets];
        }

        $combineJs = [];
        $combineCss = [];
```

```
    foreach ($assets as $asset) {
        if (substr($asset, 0, 1) == '@') {
            $combineJs[] = $asset;
            $combineCss[] = $asset;
            continue;
        }

        $extension = File::extension($asset);

        if (in_array($extension, self::$jsExtensions)) {
            $combineJs[] = $asset;
            continue;
        }

        if (in_array($extension, self::$cssExtensions)) {
            $combineCss[] = $asset;
            continue;
        }
    }

    if (count($combineCss) > count($combineJs)) {
        $extension = 'css';
        $assets = $combineCss;
    }
    else {
        $extension = 'js';
        $assets = $combineJs;
    }

    if ($aliasMap = $this->getAliases($extension)) {
        foreach ($assets as $key => $asset) {
            if (substr($asset, 0, 1) !== '@') {
                continue;
            }
            $_asset = substr($asset, 1);

            if (isset($aliasMap[$_asset])) {
                $assets[$key] = $aliasMap[$_asset];
            }
        }
    }

    return [$assets, $extension];
}

protected function prepareRequest(array $assets, $localPath = null)
{
    if (substr($localPath, -1) != '/') {
        $localPath = $localPath.'/';
    }
```

```
        $this->localPath = $localPath;
        $this->storagePath = storage_path('cms/combiner/assets');

        list($assets, $extension) = $this->prepareAssets($assets);

        /*
         * Cache and process
         */
        $cacheKey = $this->getCacheKey($assets);
        $cacheInfo = $this->useCache ? $this->getCache($cacheKey) : false;

        if (!$cacheInfo) {
            $this->setHashOnCombinerFilters($cacheKey);

            $combiner = $this->prepareCombiner($assets);

            if ($this->useDeepHashing) {
                $factory = new AssetFactory($this->localPath);
                $lastMod = $factory->getLastModified($combiner);
            }
            else {
                $lastMod = $combiner->getLastModified();
            }

            $cacheInfo = [
                'version'   => $cacheKey.'-'.$lastMod,
                'etag'      => $cacheKey,
                'lastMod'   => $lastMod,
                'files'     => $assets,
                'path'      => $this->localPath,
                'extension' => $extension
            ];

            $this->putCache($cacheKey, $cacheInfo);
        }

        return $this->getCombinedUrl($cacheInfo['version']);
    }

    protected function prepareCombiner(array $assets, $rewritePath = null)
    {
        Event::fire('cms.combiner.beforePrepare', [$this, $assets]);

        $files = [];
        $filesSalt = null;
        foreach ($assets as $asset) {
            $filters = $this->getFilters(File::extension($asset)) ?: [];
            $path = file_exists($asset) ? $asset : File::symbolizePath($asset, null) ?: $this->localPath .
$asset;
            $files[] = new FileAsset($path, $filters, public_path());
            $filesSalt .= $this->localPath . $asset;
        }
```

```
    $filesSalt = md5($filesSalt);

    $collection = new AssetCollection($files, [], $filesSalt);
    $collection->setTargetPath($this->getTargetPath($rewritePath));

    if ($this->storagePath === null) {
        return $collection;
    }

    if (!File::isDirectory($this->storagePath)) {
        @File::makeDirectory($this->storagePath);
    }

    $cache = new FilesystemCache($this->storagePath);

    $cachedFiles = [];
    foreach ($files as $file) {
        $cachedFiles[] = new AssetCache($file, $cache);
    }

    $cachedCollection = new AssetCollection($cachedFiles, [], $filesSalt);
    $cachedCollection->setTargetPath($this->getTargetPath($rewritePath));
    return $cachedCollection;
}

protected function setHashOnCombinerFilters($hash)
{
    $allFilters = call_user_func_array('array_merge', $this->getFilters());

    foreach ($allFilters as $filter) {
        if (method_exists($filter, 'setHash')) {
            $filter->setHash($hash);
        }
    }
}

protected function getDeepHashFromAssets($assets)
{
    $key = '';

    $assetFiles = array_map(function ($file) {
        return file_exists($file) ? $file : File::symbolizePath($file, null) ?: $this->localPath . $file;
    }, $assets);

    foreach ($assetFiles as $file) {
        $filters = $this->getFilters(File::extension($file));

        foreach ($filters as $filter) {
            if (method_exists($filter, 'hashAsset')) {
                $key .= $filter->hashAsset($file, $this->localPath);
            }
        }
```

```
    }

    return $key;
}

protected function getCombinedUrl($outputFilename = 'undefined.css')
{
    $combineAction = 'System\Classes\Controller@combine';
    $actionExists = Route::getRoutes()->getByAction($combineAction) !== null;

    if ($actionExists) {
        return Url::action($combineAction, [$outputFilename], false);
    }
    else {
        return '/combine/'.$outputFilename;
    }
}

protected function getTargetPath($path = null)
{
    if ($path === null) {
        $baseUri = substr(Request::getBaseUrl(), strlen(Request::getBasePath()));
        $path = $baseUri.'/combine';
    }

    if (strpos($path, '/') === 0) {
        $path = substr($path, 1);
    }

    $path = str_replace('.', '-', $path).'/';
    return $path;
}

//

public static function registerCallback(callable $callback)
{
    self::$callbacks[] = $callback;
}

public function registerFilter($extension, $filter)
{
    if (is_array($extension)) {
        foreach ($extension as $_extension) {
            $this->registerFilter($_extension, $filter);
        }
        return;
    }

    $extension = strtolower($extension);

    if (!isset($this->filters[$extension])) {
```

```
            $this->filters[$extension] = [];
        }

        if ($filter !== null) {
            $this->filters[$extension][] = $filter;
        }

        return $this;
    }

    public function resetFilters($extension = null)
    {
        if ($extension === null) {
            $this->filters = [];
        }
        else {
            $this->filters[$extension] = [];
        }

        return $this;
    }

    /**
     * Returns filters.
     * @param string $extension Extension name. Eg: css
     * @return self
     */
    public function getFilters($extension = null)
    {
        if ($extension === null) {
            return $this->filters;
        }
        elseif (isset($this->filters[$extension])) {
            return $this->filters[$extension];
        }
        else {
            return null;
        }
    }

    public function registerBundle($files, $destination = null, $extension = null)
    {
        if (!is_array($files)) {
            $files = [$files];
        }

        $firstFile = array_values($files)[0];

        if ($extension === null) {
            $extension = File::extension($firstFile);
        }
```

```
    $extension = strtolower(trim($extension));

    if ($destination === null) {
        $file = File::name($firstFile);
        $path = dirname($firstFile);
        $preprocessors = array_except(self::$cssExtensions, 'css');

        if (in_array($extension, $preprocessors)) {
            $cssPath = $path.'/../css';
            if (
                in_array(strtolower(basename($path)), $preprocessors) &&
                File::isDirectory(File::symbolizePath($cssPath))
            ) {
                $path = $cssPath;
            }
            $destination = $path.'/'.$file.'.css';
        }
        else {
            $destination = $path.'/'.$file.'-min.'.$extension;
        }
    }

    $this->bundles[$extension][$destination] = $files;

    return $this;
}

public function getBundles($extension = null)
{
    if ($extension === null) {
        return $this->bundles;
    }
    elseif (isset($this->bundles[$extension])) {
        return $this->bundles[$extension];
    }
    else {
        return null;
    }
}


public function registerAlias($alias, $file, $extension = null)
{
    if ($extension === null) {
        $extension = File::extension($file);
    }

    $extension = strtolower($extension);

    if (!isset($this->aliases[$extension])) {
        $this->aliases[$extension] = [];
    }
```

```
    $this->aliases[$extension][$alias] = $file;

    return $this;
}

public function resetAliases($extension = null)
{
    if ($extension === null) {
        $this->aliases = [];
    }
    else {
        $this->aliases[$extension] = [];
    }

    return $this;
}

public function getAliases($extension = null)
{
    if ($extension === null) {
        return $this->aliases;
    }
    elseif (isset($this->aliases[$extension])) {
        return $this->aliases[$extension];
    }
    else {
        return null;
    }
}


protected function putCache($cacheKey, array $cacheInfo)
{
    $cacheKey = 'combiner.'.$cacheKey;

    if (Cache::has($cacheKey)) {
        return false;
    }

    $this->putCacheIndex($cacheKey);
    Cache::forever($cacheKey, base64_encode(serialize($cacheInfo)));
    return true;
}

protected function getCache($cacheKey)
{
    $cacheKey = 'combiner.'.$cacheKey;

    if (!Cache::has($cacheKey)) {
        return false;
    }
```

```
    return @unserialize(@base64_decode(Cache::get($cacheKey)));
}

protected function getCacheKey(array $assets)
{
    $cacheKey = $this->localPath . implode('|', $assets);

    if ($this->useDeepHashing) {
        $cacheKey .= $this->getDeepHashFromAssets($assets);
    }

    $dataHolder = (object) ['key' => $cacheKey];
    Event::fire('cms.combiner.getCacheKey', [$this, $dataHolder]);
    $cacheKey = $dataHolder->key;

    return md5($cacheKey);
}

public static function resetCache()
{
    if (Cache::has('combiner.index')) {
        $index = (array) @unserialize(@base64_decode(Cache::get('combiner.index'))) ?: [];

        foreach ($index as $cacheKey) {
            Cache::forget($cacheKey);
        }

        Cache::forget('combiner.index');
    }

    CacheHelper::instance()->clearCombiner();
}

protected function putCacheIndex($cacheKey)
{
    $index = [];

    if (Cache::has('combiner.index')) {
        $index = (array) @unserialize(@base64_decode(Cache::get('combiner.index'))) ?: [];
    }

    if (in_array($cacheKey, $index)) {
        return false;
    }

    $index[] = $cacheKey;

    Cache::forever('combiner.index', base64_encode(serialize($index)));

    return true;
}
```

```
}
```

RequestLogs.php //Клас для логування запитів

```
<?php namespace System\Controllers;

use Str;
use Lang;
use File;
use Flash;
use Backend;
use Redirect;
use BackendMenu;
use Backend\Classes\Controller;
use ApplicationException;
use System\Classes\SettingsManager;
use System\Models\RequestLog;
use Exception;

class RequestLogs extends Controller
{
    public $implement = [
        \Backend\Behaviors\FormController::class,
        \Backend\Behaviors\ListController::class
    ];
    public $formConfig = 'config_form.yaml';

    public $listConfig = 'config_list.yaml';

    public $requiredPermissions = ['system.access_logs'];

    public function __construct()
    {
        parent::__construct();

        BackendMenu::setContext('October.System', 'system', 'settings');
        SettingsManager::setContext('October.System', 'request_logs');
    }

    public function index_onRefresh()
    {
        return $this->listRefresh();
    }

    public function index_onEmptyLog()
    {
        RequestLog::truncate();
        Flash::success(Lang::get('system::lang.request_log.empty_success'));
        return $this->listRefresh();
    }
```

```
    public function index_onDelete()
    {
        if (($checkedIds = post('checked')) && is_array($checkedIds) && count($checkedIds)) {

            foreach ($checkedIds as $recordId) {
                if (!$record = RequestLog::find($recordId)) continue;
                $record->delete();
            }

            Flash::success(Lang::get('backend::lang.list.delete_selected_success'));
        }
        else {
            Flash::error(Lang::get('backend::lang.list.delete_selected_empty'));
        }

        return $this->listRefresh();
    }
}
```

SystemController.php //Головний контролер підсистеми

```
<?php namespace System\Classes;

use Lang;
use ApplicationException;
use System\Classes\CombineAssets;
use Illuminate\Routing\Controller as ControllerBase;
use Exception;

class SystemController extends ControllerBase
{
    public function combine($name)
    {
        try {

            if (!strpos($name, '-')) {
                throw new ApplicationException(Lang::get('system::lang.combiner.not_found', ['name'
=> $name]));
            }

            $parts = explode('-', $name);
            $cacheId = $parts[0];

            $combiner = CombineAssets::instance();
            return $combiner->getContents($cacheId);

        }
        catch (Exception $ex) {
            return '/* '.e($ex->getMessage()).' */';
        }
    }
```

```
}
```

PluginBase.php //Клас для координації функцій плагінів

```
<?php namespace System\Classes;

use Illuminate\Support\ServiceProvider as ServiceProviderBase;
use ReflectionClass;
use SystemException;
use Yaml;
use Backend;

class PluginBase extends ServiceProviderBase
{
    protected $loadedYamlConfiguration = false;

    public $require = [];

    public $elevated = false;

    public $disabled = false;

    public function pluginDetails()
    {
        $thisClass = get_class($this);

        $configuration = $this->getConfigurationFromYaml(sprintf('Plugin configuration file
plugin.yaml is not '.
            'found for the plugin class %s. Create the file or override pluginDetails() '.
            'method in the plugin class.', $thisClass));

        if (!array_key_exists('plugin', $configuration)) {
            throw new SystemException(sprintf(
                'The plugin configuration file plugin.yaml should contain the "plugin" section: %s.',
$thisClass)
            );
        }

        return $configuration['plugin'];
    }

    public function register()
    {
    }

    public function boot()
    {
    }

    public function registerMarkupTags()
    {
```

```
    return [];
}

public function registerComponents()
{
    return [];
}

public function registerNavigation()
{
    $configuration = $this->getConfigurationFromYaml();
    if (array_key_exists('navigation', $configuration)) {
        $navigation = $configuration['navigation'];

        if (is_array($navigation)) {
            array_walk_recursive($navigation, function (&$item, $key) {
                if ($key === 'url') {
                    $item = Backend::url($item);
                }
            });
        }

        return $navigation;
    }
}

public function registerPermissions()
{
    $configuration = $this->getConfigurationFromYaml();
    if (array_key_exists('permissions', $configuration)) {
        return $configuration['permissions'];
    }
}

public function registerSettings()
{
    $configuration = $this->getConfigurationFromYaml();
    if (array_key_exists('settings', $configuration)) {
        return $configuration['settings'];
    }
}

public function registerSchedule($schedule)
{
}

public function registerReportWidgets()
{
    return [];
}

public function registerFormWidgets()
```

```
    {
        return [];
    }

    public function registerListColumnTypes()
    {
        return [];
    }

    public function registerMailTemplates()
    {
        return [];
    }

    public function registerConsoleCommand($key, $class)
    {
        $key = 'command.'.$key;

        $this->app->singleton($key, function ($app) use ($class) {
            return new $class;
        });

        $this->commands($key);
    }

    protected function getConfigurationFromYaml($exceptionMessage = null)
    {
        if ($this->loadedYamlConfiguration !== false) {
            return $this->loadedYamlConfiguration;
        }

        $reflection = new ReflectionClass(get_class($this));
        $yamlFilePath = dirname($reflection->getFileName()).'/plugin.yaml';

        if (!file_exists($yamlFilePath)) {
            if ($exceptionMessage) {
                throw new SystemException($exceptionMessage);
            }
            else {
                $this->loadedYamlConfiguration = [];
            }
        }
        else {
            $this->loadedYamlConfiguration = Yaml::parse(file_get_contents($yamlFilePath));
            if (!is_array($this->loadedYamlConfiguration)) {
                throw new SystemException(sprintf('Invalid format of the plugin configuration file: %s.
The file should define an array.', $yamlFilePath));
            }
        }

        return $this->loadedYamlConfiguration;
    }
```

```
}
```

PluginManager.php ///клас для управління плагінами підсистеми

```
<?php namespace System\Classes;

use App;
use Str;
use File;
use Lang;
use View;
use Config;
use RecursiveIteratorIterator;
use RecursiveDirectoryIterator;
use ApplicationException;

class PluginManager
{
    use \October\Rain\Support\Traits\Singleton;

    protected $app;

    protected $plugins;

    protected $pathMap = [];

    protected $registered = false;

    protected $booted = false;

    protected $metaFile;

    protected $disabledPlugins = [];

    protected $registrationMethodCache = [];

    public static $noInit = false;

    protected function init()
    {
        $this->bindContainerObjects();
        $this->metaFile = storage_path('cms/disabled.json');
        $this->loadDisabled();
        $this->loadPlugins();

        if ($this->app->runningInBackend()) {
            $this->loadDependencies();
        }
    }

    public function bindContainerObjects()
    {
        $this->app = App::make('app');
```

```
}

public function loadPlugins()
{
    $this->plugins = [];

    foreach ($this->getPluginNamespaces() as $namespace => $path) {
        $this->loadPlugin($namespace, $path);
    }

    return $this->plugins;
}

public function loadPlugin($namespace, $path)
{
    $className = $namespace.'\Plugin';
    $classPath = $path.'/Plugin.php';

    // Autoloader failed?
    if (!class_exists($className)) {
        include_once $classPath;
    }

    // Not a valid plugin!
    if (!class_exists($className)) {
        return;
    }

    $classObj = new $className($this->app);
    $classId = $this->getIdentifier($classObj);

    if ($this->isDisabled($classId)) {
        $classObj->disabled = true;
    }

    $this->plugins[$classId] = $classObj;
    $this->pathMap[$classId] = $path;

    return $classObj;
}

public function registerAll($force = false)
{
    if ($this->registered && !$force) {
        return;
    }

    foreach ($this->plugins as $pluginId => $plugin) {
        $this->registerPlugin($plugin, $pluginId);
    }

    $this->registered = true;
```

```
}

public function unregisterAll()
{
    $this->registered = false;
    $this->plugins = [];
}

public function registerPlugin($plugin, $pluginId = null)
{
    if (!$pluginId) {
        $pluginId = $this->getIdentifier($plugin);
    }

    if (!$plugin) {
        return;
    }

    $pluginPath = $this->getPluginPath($plugin);
    $pluginNamespace = strtolower($pluginId);

    $langPath = $pluginPath . '/lang';
    if (File::isDirectory($langPath)) {
        Lang::addNamespace($pluginNamespace, $langPath);
    }

    if ($plugin->disabled) {
        return;
    }

    $autoloadPath = $pluginPath . '/vendor/autoload.php';
    if (File::isFile($autoloadPath)) {
        ComposerManager::instance()->autoload($pluginPath . '/vendor');
    }

    if (!self::$noInit || $plugin->elevated) {
        $plugin->register();
    }

    $configPath = $pluginPath . '/config';
    if (File::isDirectory($configPath)) {
        Config::package($pluginNamespace, $configPath, $pluginNamespace);
    }

    /*
     * Register views path
     */
    $viewsPath = $pluginPath . '/views';
    if (File::isDirectory($viewsPath)) {
        View::addNamespace($pluginNamespace, $viewsPath);
    }
```

```
    $initFile = $pluginPath . '/init.php';
    if (!self::$noInit && File::exists($initFile)) {
        require $initFile;
    }

    $routesFile = $pluginPath . '/routes.php';
    if (File::exists($routesFile)) {
        require $routesFile;
    }
}

public function bootAll($force = false)
{
    if ($this->booted && !$force) {
        return;
    }

    foreach ($this->plugins as $plugin) {
        $this->bootPlugin($plugin);
    }

    $this->booted = true;
}

public function bootPlugin($plugin)
{
    if (!$plugin || $plugin->disabled) {
        return;
    }

    if (!self::$noInit || $plugin->elevated) {
        $plugin->boot();
    }
}

public function getPluginPath($id)
{
    $classId = $this->getIdentifier($id);
    if (!isset($this->pathMap[$classId])) {
        return null;
    }

    return File::normalizePath($this->pathMap[$classId]);
}

public function exists($id)
{
    return (!$this->findByIdentifier($id) || $this->isDisabled($id))
        ? false
        : true;
}
```

```
public function getPlugins()
{
   return array_diff_key($this->plugins, $this->disabledPlugins);
}

public function findByNamespace($namespace)
{
   if (!$this->hasPlugin($namespace)) {
      return null;
   }

   $classId = $this->getIdentifier($namespace);

   return $this->plugins[$classId];
}

public function findByIdentifier($identifier)
{
   if (!isset($this->plugins[$identifier])) {
      $identifier = $this->normalizeIdentifier($identifier);
   }

   if (!isset($this->plugins[$identifier])) {
      return null;
   }

   return $this->plugins[$identifier];
}


public function hasPlugin($namespace)
{
   $classId = $this->getIdentifier($namespace);

   return isset($this->plugins[$classId]);
}

public function getPluginNamespaces()
{
   $classNames = [];

   foreach ($this->getVendorAndPluginNames() as $vendorName => $vendorList) {
      foreach ($vendorList as $pluginName => $pluginPath) {
         $namespace = '\\'.$vendorName.'\\'.$pluginName;
         $namespace = Str::normalizeClassName($namespace);
         $classNames[$namespace] = $pluginPath;
      }
   }

   return $classNames;
}
```

```
    public function getVendorAndPluginNames()
    {
        $plugins = [];

        $dirPath = plugins_path();
        if (!File::isDirectory($dirPath)) {
            return $plugins;
        }

        $it = new RecursiveIteratorIterator(
            new RecursiveDirectoryIterator($dirPath,
RecursiveDirectoryIterator::FOLLOW_SYMLINKS)
        );
        $it->setMaxDepth(2);
        $it->rewind();

        while ($it->valid()) {
            if (($it->getDepth() > 1) && $it->isFile() && (strtolower($it->getFilename()) ==
"plugin.php")) {
                $filePath = dirname($it->getPathname());
                $pluginName = basename($filePath);
                $vendorName = basename(dirname($filePath));
                $plugins[$vendorName][$pluginName] = $filePath;
            }

            $it->next();
        }

        return $plugins;
    }

    public function getIdentifier($namespace)
    {
        $namespace = Str::normalizeClassName($namespace);
        if (strpos($namespace, '\\') === null) {
            return $namespace;
        }

        $parts = explode('\\', $namespace);
        $slice = array_slice($parts, 1, 2);
        $namespace = implode('.', $slice);
        return $namespace;
    }

    public function normalizeIdentifier($identifier)
    {
        foreach ($this->plugins as $id => $object) {
            if (strtolower($id) == strtolower($identifier)) {
                return $id;
            }
        }
```

```
    return $identifier;
}

public function getRegistrationMethodValues($methodName)
{
    if (isset($this->registrationMethodCache[$methodName])) {
        return $this->registrationMethodCache[$methodName];
    }

    $results = [];
    $plugins = $this->getPlugins();

    foreach ($plugins as $id => $plugin) {
        if (!method_exists($plugin, $methodName)) {
            continue;
        }

        $results[$id] = $plugin->{$methodName}();
    }

    return $this->registrationMethodCache[$methodName] = $results;
}


public function clearDisabledCache()
{
    File::delete($this->metaFile);
    $this->disabledPlugins = [];
}

protected function loadDisabled()
{
    $path = $this->metaFile;

    if (($configDisabled = Config::get('cms.disablePlugins')) && is_array($configDisabled)) {
        foreach ($configDisabled as $disabled) {
            $this->disabledPlugins[$disabled] = true;
        }
    }

    if (File::exists($path)) {
        $disabled = json_decode(File::get($path), true) ?: [];
        $this->disabledPlugins = array_merge($this->disabledPlugins, $disabled);
    }
    else {
        $this->writeDisabled();
    }
}

public function isDisabled($id)
{
    $code = $this->getIdentifier($id);
```

```
    if (array_key_exists($code, $this->disabledPlugins)) {
        return true;
    }

    return false;
}

protected function writeDisabled()
{
    File::put($this->metaFile, json_encode($this->disabledPlugins));
}

public function disablePlugin($id, $isUser = false)
{
    $code = $this->getIdentifier($id);
    if (array_key_exists($code, $this->disabledPlugins)) {
        return false;
    }

    $this->disabledPlugins[$code] = $isUser;
    $this->writeDisabled();

    if ($pluginObj = $this->findByIdentifier($code)) {
        $pluginObj->disabled = true;
    }

    return true;
}

public function enablePlugin($id, $isUser = false)
{
    $code = $this->getIdentifier($id);
    if (!array_key_exists($code, $this->disabledPlugins)) {
        return false;
    }

    // Prevent system from enabling plugins disabled by the user
    if (!$isUser && $this->disabledPlugins[$code] === true) {
        return false;
    }

    unset($this->disabledPlugins[$code]);
    $this->writeDisabled();

    if ($pluginObj = $this->findByIdentifier($code)) {
        $pluginObj->disabled = false;
    }

    return true;
}
```

```
public function findMissingDependencies()
{
    $missing = [];

    foreach ($this->plugins as $id => $plugin) {
        if (!$required = $this->getDependencies($plugin)) {
            continue;
        }

        foreach ($required as $require) {
            if ($this->hasPlugin($require)) {
                continue;
            }

            $missing[] = $require;
        }
    }

    return $missing;
}

    foreach ($this->plugins as $id => $plugin) {
        if (!$required = $this->getDependencies($plugin)) {
            continue;
        }

        $disable = false;

        foreach ($required as $require) {
            if (!$pluginObj = $this->findByIdentifier($require)) {
                $disable = true;
            }
            elseif ($pluginObj->disabled) {
                $disable = true;
            }
        }

        if ($disable) {
            $this->disablePlugin($id);
        }
        else {
            $this->enablePlugin($id);
        }
    }
}

public function getDependencies($plugin)
{
    if (is_string($plugin) && (!$plugin = $this->findByIdentifier($plugin))) {
        return false;
    }
```

```
    if (!isset($plugin->require) || !$plugin->require) {
        return null;
    }

    return is_array($plugin->require) ? $plugin->require : [$plugin->require];
}

public function sortByDependencies($plugins = null)
{
    if (!is_array($plugins)) {
        $plugins = $this->getPlugins();
    }

    $result = [];
    $checklist = $plugins;

    $loopCount = 0;
    while (count($checklist)) {

        if (++$loopCount > 999) {
            throw new ApplicationException('Too much recursion');
        }

        foreach ($checklist as $code => $plugin) {

            $depends = $this->getDependencies($plugin) ?: [];
            $depends = array_filter($depends, function ($pluginCode) use ($plugins) {
                return isset($plugins[$pluginCode]);
            });

            if (!$depends) {
                array_push($result, $code);
                unset($checklist[$code]);
                continue;
            }

            $depends = array_diff($depends, $result);
            if (count($depends) > 0) {
                continue;
            }

            array_push($result, $code);
            unset($checklist[$code]);
        }

    }

    return $result;
}
```

```
    public function deletePlugin($id)
    {
        UpdateManager::instance()->rollbackPlugin($id);

        if ($pluginPath = PluginManager::instance()->getPluginPath($id)) {
            File::deleteDirectory($pluginPath);
        }
    }


    public function refreshPlugin($id)
    {
        $manager = UpdateManager::instance();
        $manager->rollbackPlugin($id);
        $manager->updatePlugin($id);
    }
}
```

ErrorHandler.php // клас для обробки некоректних URL

```
<?php namespace System\Classes;

use Log;
use View;
use Config;
use Cms\Classes\Theme;
use Cms\Classes\Router;
use Cms\Classes\Controller as CmsController;
use October\Rain\Exception\ErrorHandler as ErrorHandlerBase;
use October\Rain\Exception\ApplicationException;

class ErrorHandler extends ErrorHandlerBase
{
    public function beforeHandleError($exception)
    {
        if ($exception instanceof ApplicationException) {
            Log::error($exception);
        }
    }

    public function handleCustomError()
    {
        if (Config::get('app.debug', false)) {
            return null;
        }

        $theme = Theme::getActiveTheme();
        $router = new Router($theme);

        // Use the default view if no "/error" URL is found.
        if (!$router->findByUrl('/error')) {
            return View::make('cms::error');
        }
```

```
        // Route to the CMS error page.
        $controller = new CmsController($theme);
        $result = $controller->run('/error');

        // Extract content from response object
        if ($result instanceof \Symfony\Component\HttpFoundation\Response) {
            $result = $result->getContent();
        }

        return $result;
    }

    public function handleDetailedError($exception)
    {
        // Ensure System view path is registered
        View::addNamespace('system', base_path().'/modules/system/views');

        return View::make('system::exception', ['exception' => $exception]);
    }
}
```

ДОДАТОК Б

ВІДГУК
керівника економічного розділу
на кваліфікаційну роботу бакалавра
на тему:
«Розробка веб-ориентованого додатку для
продажу побутового обладнання для очистки води на мові програмування PHP
з використанням фреймворку Zend
студента групи 122-18ск-1 Дубика Дениса Павловича

Керівник економічного розділу
Зав. каф. ПЕП та ПУ, д.е.н. **О. Г. Вагонова**

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	**Опис**
Пояснювальні документи	
Диплом Дубик.doc	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Диплом Дубик.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF.
Програма	
diplom Дубик.zip	Архів. Містить коди програми і откомпільовану програму.
Презентація	
Презентація Дубик.ppt	Презентація кваліфікаційної роботи.