Міністерство освіти і науки України
Сумський державний університет
Навчально-науковий інститут бізнес-технологій «УАБС»
Кафедра економічної кібернетики

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

на тему «Розробка системи управління контентом Інтернет-магазину»

Виконав студент 2 курсу, групи ЕК.м-61а
(номер курсу) (шифр групи)
Спеціальності 051«Економіка («Економічна кібернетика»)»
Грабар А.П.
(прізвище, ініціали студента)
Керівник к.т.н., доцент Яценко В.В.
(посада, науковий ступінь, прізвище, ініціали)

Суми – 2018 рік

ЗМІСТ

ВСТУП

Постановка проблеми. Інтернет, як глобальна комп'ютерна мережа, зв'язує десятки мільйонів абонентів у більш як 160 країнах світу. Щомісяця поширення зростає на 10–14%. Інтернет утворює ядро, яке забезпечує взаємодію інформаційних мереж, що належать різним установам у всьому світі. Якщо раніше вона використовувалася виключно як середовище для передачі файлів і повідомлень електронної пошти, то сьогодні вирішуються складніші завдання, які підтримують функції мережного пошуку та доступу до розподілених інформаційних ресурсів й електронних архівів. Отже, Інтернет можна розглядати як деякий глобальний інформаційний простір. Інтернет дав можливість впровадити електронну торгівлю до комплексної мережі комерційної діяльності. Електронна торгівля не обмежується сферою бізнесу. Спонтанне формування груп за інтересами серед користувачів Інтернету скоротило розрив між виробником і споживачем, підвищивши тим самим ефективність економіки. Для Інтернету не існує кордонів, він забезпечує можливість ділових контактів для людей у будь-якій точці світу, де є доступ до мережі. Інтернет дає можливість компаніям вийти із своїм товаром на величезний світовий ринок, значно знижувати витрати в побудованих ними ланцюжках попиту та пропозиції. Ставши одним з інструментів ведення бізнесу, Інтернет істотно підвищує швидкість і динаміку взаємовідносин бізнес-партнерів.

Успіх будь-якого бізнесу багато в чому залежить від того, наскільки ефективно компанія доводить свою інформацію до клієнтів і партнерів. Створення інтернет-сайту та розміщення його в Інтернеті – один з альтернативних методів позиціонування компанії та інформування цільової аудиторії. Саме в Інтернеті багато хто шукає докладну, і свіжу інформацію, на основі якої можна отримати уявлення про новини компанії, товари і послуги. Зараз важко уявити велику компанію без інтернет-сайту. Наявність у компанії гарного мережевого представництва не тільки підсилює позитивний образ фірми, а й говорить на користь надійності і ґрунтовності компанії, створює певний імідж і враження. В даний час найбільш популярним видом віртуальної торгівлі є Інтернет-магазин.

Інтернет-магазин зазвичай містить наочний і барвистий каталог наданих товарів, з їх достатнім описом і зазначенням ціни, що дозволяє зацікавити потенційного покупця, допомогти йому зробити вибір і, в підсумку, зробити покупку. У багатьох компаніях зустрічаються проблеми збуту, які заважають ефективно працювати відділу продажів, і не зникають навіть з підбором хороших продавців. Вирішити їх можна лише шляхом автоматизації процесу продажів.

Актуальність теми. Система управління контентом (CMS) – це комплексна програма для створення та підтримки Інтернет-проектів будь-якої складності. CMS забезпечує користувача необхідними візуальними засобами для створення інтерактивних сторінок сайту, вміст яких можна змінювати, використовуючи вбудований у систему текстовий редактор. Крім цього, CMS надають користувачеві широкий спектр додаткових послуг, таких як менеджмент користувачів сайту, публікація новин або статей, , Інтернет-магазини, галереї, дошки оголошень тощо. Написання CMS зводиться до двох випадків: під певний проект та універсальними для будь-якої системи. Система управління контентом – призначена для роботи на хостингу, наданому провайдером послуг. Використання CMS для ведення бізнесу стає невід’ємною частиною сучасного світу. Розвиток інформаційних технологій привів до того, що в сучасній світовій економіці контент став ключовим поняттям у процесах розвитку і провадження бізнесу в різних галузях.

Загалом проблематика даної роботи має не велике висвітлення у вітчизняній наукової літературі. Більш актуальними є іноземні публікації наукових досліджень та технічна інформація від компаній виробників систем управління вмістом Інтернет-магазину, але все ж слід зазначити такі наукові публікації: Н. Кабір «Переваги ведення бізнесу в мережі Інтернет», Д. Козьє «Електронна комерція», Я.Кісь «Реалізація Інтернет-магазину з використанням інтелектуальної компоненти», А.Ю.Берко «Системи електронної контент комерції», А.Ю.Берко, В.М.Дорош, Л.В.Чирун «Інтелектуальна система управління контентом сайтів електронного бізнесу».

Метою роботи є аналіз, проектування та розробка системи управління контентом Інтернет-магазину.

Об’єктом дослідження є процес розробки веб-орієнтованої системи для Інтернет-магазину.

Предметом дослідження є інформаційна система керування вмістом веб-ресурсу для електронної комерції.

Мета роботи досягається розв’язанням таких задач:

– охарактеризувати системи керування контентом;

– проаналізувати основні переваги та проблеми CMS;

– сформувати вимоги до системи керування контентом Інтернет-магазину;

– виконати проектування інформаційної системи керування контентом інтернет-магазину;

– дослідити підсистеми забезпечення функціональної частини;

– обрати архітектуру та технологію для розробки;

– визначити програмне та апаратне забезпечення;

– розробити прототип CMS Інтернет-магазину.

Ступінь вирішення проблеми. В ході виконання роботи реалізовано прототип системи керування контентом для використання в електронної комерції.

РОЗДІЛ 1. АНАЛІЗ СИСТЕМ УПРАВЛІННЯ КОНТЕНТОМ ДЛЯ ЕЛЕКТРОННОЇ КОМЕРЦІЇ

1.1. Загальна характеристика системи керування контентом.

Багато років минуло з тих пір, як з'явилася Всесвітня мережа Інтернет. З плином часу змінилося багато чого: технічні можливості мережі розширилися, аудиторія виросла, цілі і завдання, для вирішення яких була створена мережа, перестали бути виключно військовими або науковими.

Сучасний Інтернет – це величезне сховище, в якому кожен по своєму бажанню може знайти потрібний текст, музику, навчальні матеріали, а з деякого часу і просто купити все, що необхідно – від продуктів до програмного забезпечення.[2]

Віртуальний магазин – це реалізоване в мережі Інтернет представництво шляхом створення Web-сервера для продажу товарів і послуг іншим користувачам мережі Інтернет. Віртуальний магазин називають також Інтернет - магазином. До нього повністю підходить визначення віртуального підприємства. Інакше кажучи, віртуальний магазин - це співтовариство територіально роз'єднаних співробітників магазина (продавців, касирів) і покупців, які можуть спілкуватися і обмінюватися інформацією через електронні засоби зв'язку при повній (або мінімальній) відсутності особистого прямого контакту.

Інтернет-магазини все більше набирають популярність, оскільки вигідні як для продавця, так і для покупця - продавець не переплачує за оренду приміщення і не знаходиться за прилавками цілий день в очікуванні покупців, роблячи завищену ціну, а покупець має можливість придбати товар дешевше, ніж в магазині і в зручний для Вас час з будь-якого місця, де має доступ до інтернету.

У 1990 році Тім Бернерс-Лі створив перший веб-сервер і браузер. Він був відкритий для комерційного використання в 1991 році. У 1994 році відбулися інші досягнення, наприклад, онлайн-банкінг та відкриття Інтернет-магазину піци «Pizza Hut». У тому ж році Netscape представила SSL-шифрування даних, переданих в

мережі, яке стало необхідним для безпеки Інтернет-магазинів. Крім того, в 1994 році німецька компанія Intershop представила свою першу систему Інтернет-магазинів. У 1995 році Amazon запустила свій Інтернет-магазин, а в 1996 році з'явився eBay.

Переваги Інтернет-магазину:

– економія часу. Завдяки навігації і пошукових системам можна швидко знайти потрібний товар;

– доставка товарів додому. Користувачам Інтернет-магазинів більше немає потреби бігати магазинами, тягати сумки і коробки. Усе це швидко і легко роблять співробітники служби доставки;

– можливість уникнути черг традиційних магазинів – проблема, яка викликала найбільше обурення в їхніх попередників;

– цілодобове обслуговування. Це правда важливо і зручно сучасній людині, зайнятий вдень і вночі, яка має ненормований робочого дня і який встигає «до закриття магазину».

Сегмент онлайн-продажів росте практично у всіх країнах світу і особливо у Східній Європі. В Україні показник доступності високошвидкісного доступу до Інтернету все ще знаходиться далеко від бажаного показника, а це означає, що разом зі збільшенням числа активних користувачів Мережі буде рости і відсоток покупців в Інтернет-магазинах.

Електронна комерція (e-commerce) - термін, який використовується для позначення комерційної активності в мережі Інтернет. Забезпечує можливість здійснення покупок, продажів, сервісного обслуговування, проведення маркетингових заходів шляхом використання комп'ютерних мереж. Електронна комерція (в широкому сенсі) - підприємницька діяльність по здійсненню комерційних операцій з використанням електронних засобів обміну даними. Об'єктами електронної комерції є те, на що спрямована діяльність систем електронної комерції. До них можна віднести різні товари, послуги та інформацію. Об'єм електронної торгівлі зростає по експоненті з часу появи глобальної мережі.

Електронна комерція дозволяє замовляти продукцію та послуги прямо на сайті, а потім отримувати їх традиційним шляхом: за допомогою транспортної організації або для інформаційних продуктів – пересилкою каналами Інтернету[9].

З'явилася електронна комерція завдяки стрімкому розвитку технологій автоматизації продажів , впровадженню на підприємствах автоматизованих систем управління ресурсами, зростанню кількості активних інтернет-користувачів. Останнім часом електронна комерція охоплює все більш широкі сфери діяльності людини.

На сьогоднішній день звичайному інтернет-користувачеві дуже зручно і швидко оплачувати товари або послуги за допомогою інтернету, отримувати і переводити гроші зі свого електронного рахунку, користуватися послугами банків через Інтернет – все це електронна комерція.

Ринок електронної комерції розвивається швидкими темпами і досяг колосальних обсягів на Заході. Широкосмуговий доступ в інтернет сьогодні така ж необхідність, як і телебачення або стільниковий зв'язок.

За даними Світового Банку, кожен п'ятий житель Китаю має доступ в Мережу, в США кількість активних користувачів інтернету перевищила позначку в 80% від загального населення. У Фінляндії право на швидкісний Інтернет приписали до основних прав людини – це поправка до Конституції.

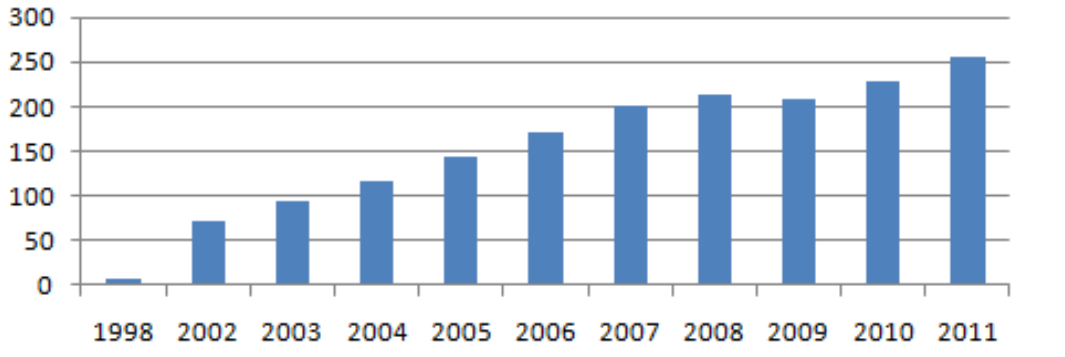


Рисунок 1.1. – Зростання інтернет-торгівлі в США за 13 річний період, в млрд. дол.

При загальної інтернетизації, користувачі Мережі, крім роботи і розваг, все частіше купують товари і послуги онлайн. Темпи зростання Інтернет-торгівлі в США – найбільш розвиненому ринку – дещо уповільнилися, але як і раніше недосяжні для інших країн (рис. 1.1).

Європейський ринок, який включає в себе 17 найбільш розвинених країн, практично вдвічі поступається ринку електронної комерції в США.

До трійки найбільш відвідуваних e-commerce сайтів України входять:

– OLX.ua (40,4% користувачів);
– Rozetka.com.ua (32,5%);
– Prom.ua (26,5%).

У порівнянні з першою половиною минулого року, в першому півріччі 2017 року товарообіг Prom.ua виріс на 100%, і склав 4 мільярди гривень. Більше половини інтернет-покупців України – люди у віці від 14 до 34 років. Трохи більше 36% з них – жителі міст з населенням понад 500 000 чоловік, 20,5% – проживають в сільській місцевості.

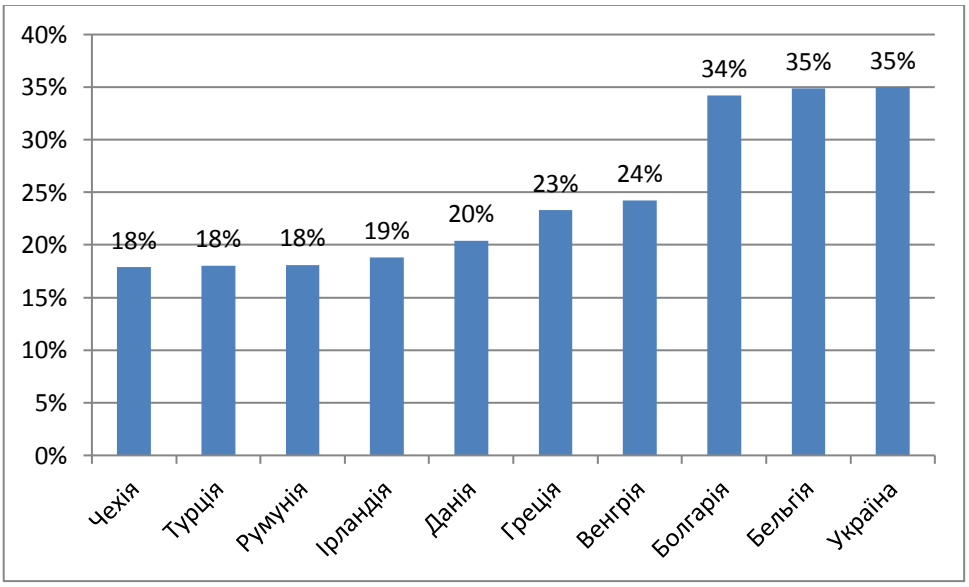


Рисунок 1.2. – Темпи зростання продажів в Інтернеті

За підсумками минулого року, Україна випередила всі інші європейські країни за таким показником як темпи зростання продажів в Інтернеті – для нашої країни цей показник склав 35% (рис. 1.2).

За результатами досліджень, проведених Українським фінансовим порталом, відсотковий розподіл операцій з купівлі/продажу товарів чи послуг в Україні на сьогоднішній день можна проілюструвати наступними діаграмами (рис. 1.3).

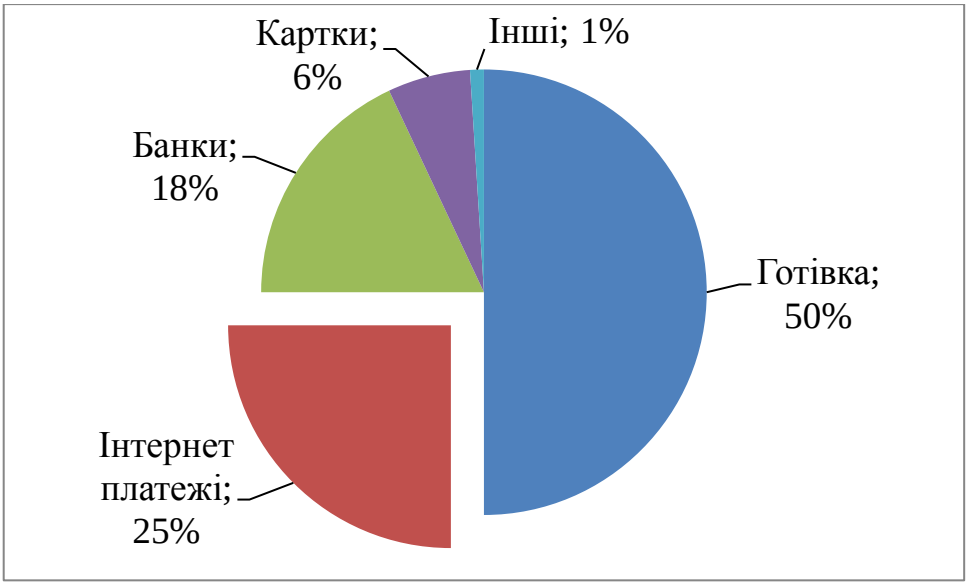


Рисунок 1.3. – Розподіл операцій з купівлі/продажу товарів

Як видно, в Україні поширення банківських Інтернет-послуг просувається досить повільно. Крім об'єктивних причин, таких як недосконалість законодавчої бази, недостача професіоналів у бізнесі для використання нових технологій та недовіра до можливостей Інтернет-бізнесу, активному розвитку електронної комерції заважає ще й психологічний фактор. Ще одна проблема на шляху розвитку Інтернет-банкінгу в нашій країні - це рівень захисту. Часто фіксуються випадки шахрайства при використанні послуг, які надаються електронною комерцією.

Підтримка перших веб-сайтів потребувала знань мови гіпертекстової розмітки (HTML) і навиків у роботі із графічними зображеннями . Однак сайти, які містять лише декілька сторінок із текстом та малюнками залишилися в минулому. На даному етапі програмування, майже кожний веб-сайт містить щонайменше кілька

різних веб-сторінок та підрозділів, багато графіки, анімації, музики й відео файлів. Керування такими відмінностями потребує багато часу та знань [1].

Безсумнівно, що для цього необхідний такий інструмент, завдяки якому завдання керування сайтом значно полегшується. Таким інструментом є система керування контентом (вмістом сайту) або CMS.

Таким чином, CMS — це програмне забезпечення, що призначене для конструювання і керування веб-сайтами. Система управління контентом дає можливість створювати динамічні сайти, оновлення яких можна робити навіть без знань html чи css.

Системи мають доступний для розуміння інтерфейс, що є легким для освоєння, а щоб додавати матеріали, новини на сайт досить вміти користуватись текстовим редактором. Такі CMS стали улюбленими як серед новачків так і професіоналів, оскільки вимагають мінімум часу і зусиль для написання власних сайтів [3].

Оцінювати придатність CMS для використання в тій чи іншій сфері можна, тільки спираючись на їх характеристики. Нижче наведені описи основних характеристик систем управління контентом.

Оптимізація процесу. Щоб контент потрапив на сайт, необхідне схвалення багатьох осіб: керівника, контент-менеджера, коректора, редактора. Оптимізовані CMS дозволяють максимально скоротити час на переміщення попереднього варіанту документа між різними відділами компанії, відразу після завершення роботи одним фахівцем відправляючи контент в інший підрозділ.

Складність процесів. Системи управління контентом здійснюють набагато більш складні процеси, ніж якби запит на доступ відбувався б до статичної сторінці, і слабкі сервери можуть не витримати пікового навантаження. Щоб сайт працював без падінь, час обробки запиту з боку CMS не повинна перевищувати декількох мілісекунд.

Прив'язка до платформи. Деякі CMS (особливо ранні продукти) володіли таким істотним недоліком, як можливість функціонування лише на одній платформі. При придбанні CMS керівництву підприємства необхідно зробити акцент на

можливості роботи системи на всіх платформах, які використовуються організацією[17].

Використання CMS – досить простий шлях побудувати власний сайт. Вибравши один із видів такої системи, потрібно:

Можливості адміністрування CMS:

– створення необмеженої кількості сторінок;

– чітка організація структури сайту;

– для кожної динамічної сторінки створення унікального опису і ключового слова з метою підвищення рейтингу в пошукових системах;

– програмування за календарем терміну початку і закінчення публікації будь-яких матеріалів;

– обмеження доступу до певних розділів сайту тільки для зареєстрованих користувачів;

– закачування зображень за допомогою браузера у власну бібліотеку - для подальшого використання з будь-якого місця сайту;

– опитування та голосування для ефективного зворотного зв'язку;

– самостійний вибір різних модулів, таких як - останні новини, лічильник відвідувань, докладна статистика відвідувань, гостьова книга, форум і т.д.;

– створення не однієї, а декількох форм зворотного зв'язку для кожного контакту;

– зміна порядку об'єктів, включаючи новини, питання, статті і т.д.;

– генератор показу випадкової новини;

– модуль прийому від авторів новин, статей і посилань;

– ієрархія об'єктів - кількість секцій, розділів, підрозділів і сторінок;

– зберігання за допомогою бібліотеки зображень GIF і JPEG-файлів для легкого доступу;

– менеджер розсилання новин;

– менеджер архіву;

– роздруківка або відправка на e-mail будь-якої статті із сайту;

– вибір з візуальних редакторів, що спрощує редагування матеріалів до рівня редагування тексту в програмі Word;

– попередній перегляд перед остаточним розміщенням;

– легка зміна дизайну;

– ощадливе використання місця на сервері за рахунок використання бази даних MYSQL;

– використання адрес сторінок адаптованих для кращої індексації всіма пошуковими системами.

CMS підходить як для невеликого, так і для великого корпоративного сайту або інформаційного проекта.

За допомогою системи управління веб-сайтом створюється :

– онлайн-магазини;

– корпоративні веб-сайти та портали;

– невеликі за обсягом змісту сайти;

– сайти для некомерційних організацій, шкіл і т.д;

– персональні сторінки;

– сайти співтовариств;

– онлайн-журнали, газети та ін.

Системи поділяються на різні види, які призначені для широко кола потреб. Існують платні і безкоштовні CMS. Оскільки створення сайтів за допомогою CMS є досить швидким і не потребує спеціальних навичок, цей інструмент стає все більш популярним. На сьогоднішній день, навіть сайти-візитки часто розробляють на CMS, хоча редагування такого сайту у візуальному редакторі типу Macromedia Dreamveawer не представляє особливих складнощів [2]. Взагалі, однією з головних функцій CMS є можливість встановлення додаткових модулів або плагінів, які значно розширюють початковий функціонал. Краще за все, для створення сайтів такого типу, використовувати спеціалізовану CMS. Так можна отримати більшість важливих функцій для реалізації проекту без необхідності додаткових плагінів, які можуть призвести до виникнення помилок [5].

Незважаючи на тип ресурсу, який використовується, при створенні сайту на CMS завжди недостатньо основних можливостей, особливо це стосується безкоштовних CMS.

Якщо потрібно створити сайт для новин, портал або блог, то оформлення не так важливо. Якщо необхідно створити Інтернет-магазин або ігровий сайт, то графічне оформлення повинно стояти на одному з перших місць при проектуванні ресурсу.

Для оформлення сайту відповідно до обраної тематики призначені графічні шаблони і теми оформлення. Пошук їх, як правило, не складає труднощів, оскільки вони доступні на просторах Інтернету. Можна навіть замовити його у програмістів індивідуально.

CMS ідеально підходить для власників веб-ресурсів, які бажають ефективно, зручно і швидко управляти змістом сайту і його структурою. Власник сайту зможе самостійно додавати, редагувати, видаляти сторінки і розділи на сайті, без знання спеціальних технологій. Всі ці дії здійснюються в захищеній адміністративній частині сайту, у вікні звичайного браузера[36].

Що до переваг системи керування сайтом, по-перше заощаджує гроші – замовивши один раз створення сайту з системою управління контентом, потім не потрібно буде оплачувати послуги вебмайстра, тому що користуватися системою управління може практично будь-який співробітник. Також простота використання, для зміни чого не будь потрібні базові знання ПК, і ніяких мов програмування. В будь-який час можна підключити додаткові модулі, не блокуючи роботу сайту і швидкий редизайн сайту, в якому можна робити зміни зовнішнього вигляду (дизайну), без втрат наповнення і зупинки ресурсу.

Щоб сайт працював, він повинен постійно оновлюватись. Щоб сайт оновлювати потрібно або утримувати співробітника - фахівця в цій галузі, або постійно платити гроші веб-дизайнеру, що буде підтримувати (оновлювати) цей сайт. Але, замовивши створення веб-сайту на системі керування контентом, одержується незалежність від фахівців, заощаджуються гроші і сайт працює стабільно.

До основних функцій систем управління контентом відносять створення контенту (CMS мають в своєму функціоналі кошти, що дозволяють генерувати вміст сайту за допомогою звичних методів).

Управління контентом (адміністратор CMS може делегувати іншим користувачам право редагування певних розділів і стежити, щоб ці права дотримувалися, також в управління контентом входить перегляд даних про внесення останніх змін і інтеграцію з застосовуваними компанією IT-системами);

Публікація контенту (відразу після створення новий контент з'являється на сайті, його зовнішній вигляд стандартизується відповідними інструментами за зразком головною веб-сторінки). Придбати або створити власноруч шаблон сайту.

1.2. Переваги використання, класифікація та проблеми уніфікації CMS.

З появою нових інформаційних технологій спостерігається постійне зростання вимог до інтерактивності і зручностей користування веб-сайтів. На зміну сайтам-сторінкам і сайтам-візиток приходять так звані «керовані» сайти, коли користувачеві надають можливість редагувати структуру сайту незалежно від його наповнення, розміщувати новий контент (інформацію), не вдаючись до допомоги розробника, керувати вмістом – будь-яким інформаційно значущим наповненням інформаційного ресурсу (наприклад, веб-сайту) - текстові описи, графіка, мультимедіа, новини компанії, спеціальні пропозиції або акції, каталог продукції або послуг компанії та інше. Зазнає великі зміни й Інтернет-торгівля.

Сайти-каталоги (веб-вітрини), Інтернет-магазини з традиційними кошиком покупок і формою оформлення замовлення on-line витісняють повноцінні CRM системи (Customer Relationship Management - управління взаємовідносинами з клієнтами) на сучасних Інтернет ресурсах. Для таких систем характерна робота з кожним клієнтом індивідуально. Це і «Особисті кошики», що дозволяють набирати товари, які ви купували раніше, «Історія покупок», що ведеться з початку вашої реєстрації в Інтернет-магазині, «Система накопичувальних знижок», «Система підтримки декількох адрес доставки» і т.д.

В даний час в мережі з'являється все більше багато яких on-line сервісів де можна здійснити оплату кредитною карткою, оплатити послуги доступу Інтернет, поповнити рахунок на мобільному телефоні, зробити електронну фотографію, змонтувати відео-ролик і т.д. Все перераховане призводить до того що сучасні веб-розробки це повноцінні програмні продукти (дистанційні інформаційні системи), що вимагають для свого створення повноцінної програмної середовища або програмного фреймворка[37].

Використання CMS надає багато переваг.

Оперативне оновлення інформації – інформацію публікує співробітник, що володіє інформацією, без додаткових посередників у вигляді технічних фахівців.

CMS призначені для автоматизації процесу публікації інформації на web-сайті, надаючи користувачам можливість самим публікувати матеріали в мережі і визначати їх візуальне подання, використовуючи для цього стандартні засоби, що не вимагають знання мови HTML і складних процедур.

CMS дає змогу створювати і модифікувати інформаційне наповнення сайтів.

Зниження вартості підтримки – оновлює інформацію користувач. Вартість знижується за рахунок зниження втрат часу на пошуки документів, виключення дублювання і помилок, збільшення швидкості зв'язку з партнерами та клієнтами.

Додаткові сервіси – пошук, форуми, голосування і т.д., вимагають інтерактивної взаємодії з користувачем. Вони вже реалізовані в рамках CMS.

Зменшення термінів і вартості розробки – потрібна функціональність вже реалізована в CMS і може бути використана відразу. Фірми-виробники пропонують універсальні системи управління контентом, які можуть реалізувати будь-які Інтернет-проекти.

Підвищення якості розробки – під час розроблення повністю або частково використовуються готові модулі, які вже пройшли неодноразове тестування. Також можливе підключення вже готових розроблених модулів, які не надаються в стартовому наборі.

Зниження вартості подальших модифікацій – CMS дають змогу розділити дані та їх подання. Це дає змогу набагато простіше змінити зовнішній вигляд сайту, ніж у випадку зі статичним сайтом [5].

Існує класифікація CMS, яка побудована на моделі подання даних: модульна, об'єктна і мережева.(рис. 1.4).

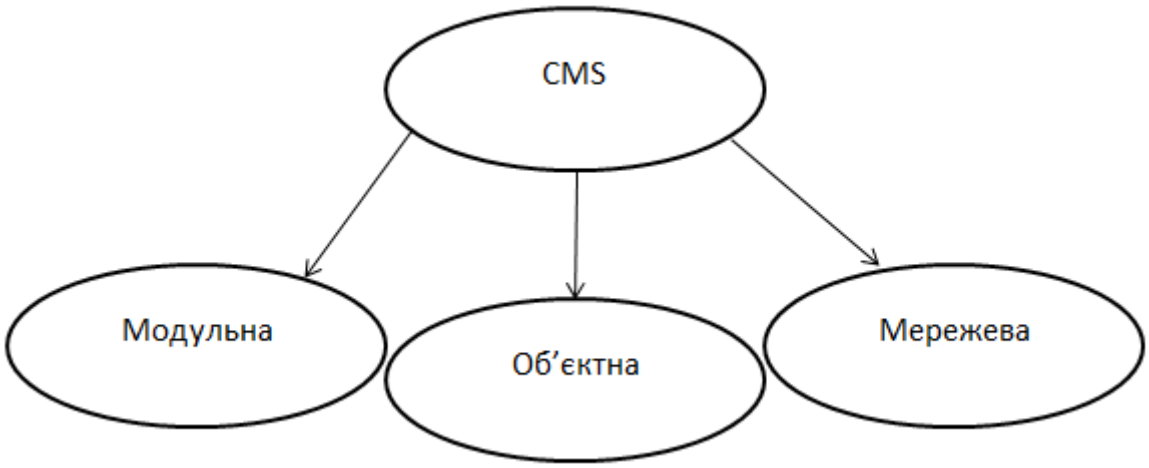


Рисунок 1.4. Класифікація CMS

Модульною моделлю подання даних є розділення вмісту сайта (контенту) на окремі модулі, які поділяють за типом вмісту. Кожен модуль відповідає тільки за свою частину контенту. Структура даних залежить від модуля, і вся робота з контентом зосереджена всередині модуля. Розширювати функціональність можна за рахунок додавання нового модуля, заміни або редагування існуючого коду. Незважаючи на очевидну обмеженість моделі даних, системи на її основі найпопулярніші завдяки своїй простоті. У модульних CMS-систем є один спільний недолік – строго фіксована в межах модуля структура вмісту, але за необхідності для розширення їх функціональності можна скористатися зовнішніми модулями. Очевидна перевага цих систем – можливість одержання майже повністю готового до використання порталу за короткий час[38].

Для розуміння об'єктної моделі подання даних треба оперувати такими поняттями, як клас і об'єкт. Об’єкт і клас є основою цієї моделі. Класи представляють побудову даних і являють собою набір атрибутів (рядок тексту,

число, зображення і т.д.). Екземпляри класу (об'єкти) мають певну структуру і можуть містити інші об'єкти, утворюючи довільну ієрархічну структуру. Об'єкти класу можуть наслідувати властивості, сутність і функцію об'єктів, які в них розміщуються. Клас контенту не зберігає реальних даних – таку інформацію містять об'єкти. Визначивши один клас, можна створити безліч його представників (контент-об'єктів). У CMS-системах дані зазвичай зберігаються за допомогою реляційної або об'єктної бази даних. Зазвичай системи, засновані на об'єктно-орієнтовній моделі даних, найбільш функціональні та гнучкі, але водночас і найскладніші.

Мережева модель подання даних в CMS-системах ґрунтується на теорії графів: побудова інформації представляється у вигляді вузлів зі зв'язками між ними.

Фундаментом системи може слугувати як мережева, так і традиційна реляційна СКБД, на якій основується мережева модель опису даних. У реляційних таблицях зберігається інформація про вузли, їх атрибути і зв'язки між ними. Зв'язок відрізняється від атрибуту тим, що в ньому зберігається посилання на інший вузол, а в атрибуті – власне значення. Для отримання даних із спрямованого графу зазвичай використовують рекурсивні процедури обробки, такі як складання списків вузлів, визначення атрибутів вузла за атрибутами батька та інше [5].

Класифікувати CMS-системи можна за різними критеріями.

Основним з них є ціна, простота використання, рівень безпеки, функціональні можливості[32].

Створити найпростішу CMS систему може навіть початківець прорамміст. Найпростіший скрипт, що виконує завантаження інформації в базу даних і відображення її за запитом користувача на певній сторінці сайту в першому наближенні є CMS-системою. Але різниця цього скрипта і реальної ситеми управління сайтом приблизно така ж, як між велосипедом «Україна» і сучасним авіалайнером.

Перш за все, CMS бувають комерційні і з відкритим кодом, тобто безкоштовні. Різниця між ними, напевно, не стільки в грошах, скільки в

витрачаються на розробку сайту зусиллях. Безкоштовна система вимагає набагато більше уваги при розробці проекту[29].

Безкоштовні системи досить часто накладають суттєві обмеження на дизайн сайту і дуже рідко створюються як уніфікований продукт. Як правило, вони є чиїмось рішенням якихось приватних завдань. Очевидно, що народжене в таких умовах ПО може добре вирішувати завдання створення такого ж роду сайтів, але може бути абсолютно не пристосоване для вирішення завдань іншого плану, скажімо, електронної комерції[31].

У безкоштовних систем і свої переваги. Вони цілком можуть підійти для невеликих проектів зі стандартним набором функцій. Якщо в запропонованому переліку можливостей немає будь-якої, то можна завжди звернутися до ентузіастам цієї системи за порадою. Як правило, для найпопулярніших систем розроблена достатня кількість додаткових модулів самих різних видів. Ці ж ентузіасти завжди допоможуть порадами по налаштуванні системи. Нарешті, безкоштовні системи менш вимогливі до умов хостингу проекту.

Комерційні ж системи сильні, перш за все, своєю оперативної тех-підтримкою, своєчасними безкоштовне оновлення, можливістю поступового, по мірі виникнення потреб, придбання додаткових модулів і впровадження їх в інтернет-проект. Перед тим як поцікавитися функціоналом і вартістю, подумайте про інтуїтивно зрозумілий інтерфейс адміністративного розділу і зручність роботи з системою для некваліфікованого користувача.

Архітектура системи управління контентом визначає її тривалий життєвий цикл, можливість легкого розширення функціональності, високий рівень швидкодії і безпеки в процесі робочої експлуатації. Класична схема побудови багатьох CMS-систем ґрунтується на поділі візуального дизайну сайту і його інформаційного наповнення. При створенні сайту за допомогою такої системи розробляється набір шаблонів сторінок, в яких згодом розміщується контент, а також бази даних для його зберігання. Роль кожного з учасників схеми функціонування CMS розподіляються наступним чином:

– розробник (група впровадження, компанія-розробник) обмежується лише створенням “початкової” інформаційної системи на основі CMS;

– користувач публікує необхідну інформацію і визначає її подання;

– адміністратор керує користувачами;

– функціями авторів контенту є створення потрібного наповнення.

Для розроблення або вибору систем управління контентом виникає проблема для замовника (компанія розробник, клієнт), яка CMS-система потрібна. Компанії-виробники пропонують набір функціональності, які, на перший погляд, задовольняють потреби користувача. Але при детальному використанні може виникнути низка недоліків.

Виділяють основні недоліки CMS-систем:

– універсальність – негативно впливає на швидкість роботи конкретного сайта тому, що конкретний сайт ніколи не потребує всіх можливостей CMS;

– розширення функціональності – сайт стає проблематичним, коли завдання виходить за межі можливостей CMS. Тобто в кожної CMS існує свій набір функціональності, який обмежує роботу конкретного сайта;

– однотипний дизайн – практично всі системи допускають застосування різних шаблонів дизайну, але вони не можуть кардинально вплинути на структуру сторінки;

– архітектура – системи управління контентом використовують різні моделі подання даних та різні архітектурні рішення для побудови структури CMS;

– міграція даних – для роботи з базою даних використовуються різноманітні засоби СУБД, тому неможливо переносити контент з однієї системи до іншої;

– контроль версій – створення нових версій зводиться до створення нових модулів, а не до удосконалення CMS.

Ієрархія в CMS будується переважно за допомогою категорій, до яких належить контент. Якщо під час розроблення виникає питання додавання дописів, для яких немає створеної категорії контенту, розробник створює нові модулі системи, а також змінює структуру та створює нову версію CMS.

Варто згадати проблему, що між існуючими системами управління контентом немає спільності. Відсутність уніфікації накладає певні обмеження під час переходу з однієї системи на іншу. По суті, міграція з однієї системи на іншу передбачає повне перероблення сайту або навіть замовлення нового сайту.

Уніфікація може стати необхідним кроком при розумінні CMS-систем. Класифікація CMS за потребами для замовника (соціальні мережі, для комерційної діяльності, особистого використання та CMS-державних установ) та подальша уніфікація цих категорій повинна сприяти вирішенню проблеми вибору CMS[15].

Уніфікація – це приведення будь-чого до єдиної системи, форми, одноманітності.

Подоланням недоліків CMS-систем є уніфікація, яку необхідно здійснювати у таких напрямках:

1. Потрібно розділити CMS-системи за категоріями. Приведення систем управління контентом до якогось єдиного стандарту поділу покращить їхнє сприйняття та вибірку. Основними категоріями можуть стати: ·

CMS для соціальних мереж. Відмінністю від інших категорій є повний набір функціональності для роботи як соціальної мережі (підтримка java-додатків, пошук, форуми, голосування, розширений багатоканальний режим доступу до системи і т.д.);

CMS для комерційної діяльності. Прикладами таких CMS-систем є Інтернет-магазин, Інтернет-аукціон, Інтернет-біржа, Інтернет-портал і т.д. Особливості: установлення двостороннього зв'язку з відвідувачами ресурсу, налагодження чіткої автоматизації відносин “клієнт–продавець”, єдиний підхід до ієрархічної організації контенту, додавання нових категорій товарів тощо;

CMS для особистого використання. Прикладами таких CMS є персональні блоги, персональні Інтернет-сторінки, Інтернет-щоденики, Інтернет-записники. Особливості: різнотипний дизайн, можливість розширення тощо;

CMS для державних установ. Прикладом таких CMS-систем є сайти міських та обласних рад, основною складовою яких є безпека доступу до даних та можливість додавання даних без втрат.

2. Визначити єдину архітектуру побудови ядра та доступу до даних. Архітектурне рішення є невід’ємною складовою вибору CMS-систем. Тому доцільно визначити таку побудову CMS, яка б об’єднувала все найкраще.

3. Сумісність СКБД. Для зручного обміну між системами управління контентом потрібно визначити не єдину СКБД, а CMS, що підтримують міжнародний стандарт мови SQL. Реалізація стандарту призведе до використання мови SQL під час розроблення CMS та взаємодії з системою БД при використанні тільки цієї мови. Це істотно обмежує можливий набір СКБД, але подолає проблему сумісності СКБД.

4. Уніфікувати будову схеми бази даних. Схема баз даних побудована залежно від потреб конкретної задачі, що призводить до неможливості подальшої зміни або розширення. Вирішенням стане стандартизований набір правил побудови бази даних в залежності від категорії та призначення CMS.

5. Правила нотації елементів системи і БД. Однією з проблем несумісності CMS-систем є нестандартизованість назв файлів, модулів, лінків і т.д. Для взаємодії системи необхідно привести всі правила написання до єдиного стандарту, що дасть змогу під час переходу з однієї системи на іншу подолати проблему пристосування.

6. Організувати правильність ролей та рівнів доступу. Багато проблем у безпеці CMS пов'язано з контролем прав доступу на рівні програмних модулів, але не на рівні СКБД і БД. Вирішенням стане організація розширеної багаторівневої системи доступу до системних програмних модулів та СКБД і БД.

7. Версійність (контроль версій) контенту. Необхідно: розташовувати документи у багаторівневих навігаційних меню за ступенем їх логічної взаємозалежності; визначити єдиний підхід до ієрархічної організації контенту; забезпечити можливість додавання дописів до контенту без створення нових модулів та зміни структури CMS;

8. Багатомовність. Критерій, з яким стикається користувач, вже при виборі CMS-системи. Для реалізації потрібно застосувати модульний підхід. Кожен модуль повинен містити можливість перекладу контенту будь-яку мовою за бажанням

користувача. Щоб перекласти сайт іншою мовою, достатньо підключити модуль перекладу.

9. Можливість імпорту даних з офісних додатків. Одна з важливих функцій для користування контентом – це перенесення контенту з офісних засобів, таких як MS Office з подальшим перетворенням в потрібний формат. Також забезпечити конвертування тексту із збереженням розмітки та таблиць.

10. Підтримка декількох сайтів та різнотипний дизайн. Більшість систем управління контентом дають змогу змінювати зовнішній вигляд ресурсу без зміни інформаційного наповнення. Однак дизайнерські шаблони є одноманітними через прив'язку до структури створеного за допомогою контент-системи ресурсу сайта. Необхідно організувати можливість різнотипного дизайну та неприв’язаність до стартового набору.

1.3. Формування вимог до розробки системи керування контентом.

Список вимог до CMS:

– створення контенту;
– управління контентом;
– публікація;
– презентація (дизайн, шаблони, лейаути);
– комунікація (інтерактивні модулі);
– управління користувачами (користувачі, групи, профілі, workflow);
– настройки CMS (оновлення, підключення додаткових модулів);
– статистика (моніторинг за використанням системи - хто і коли який контент завантажував, статистика відвідувань сайту);
– електронний магазин.

Контент сайту – це текстове наповнення сайту в поєднанні з графічною інформацією, а так же інформацією у вигляді таблиць, форм і т.п. Якісний контент є запорукою успіху будь-якого сайту. Досить часто, та більш того, майже завжди інформаційна складова сайту набагато важливіше його дизайну. Текстовий контент

є своєрідним інтерфейсом між власниками та відвідувачами сайту. Автор повинен вільно публікувати контент і мати повний доступ до всього функціоналу CMS. Щоб CMS була ефективною, вона повинна бути швидкою, інтуїтивно зрозумілою і зручною[28].

Ще одним пунктом з вимог являється презентація. Добре б, щоб опубліковані сторінки відповідали очікуванням ваших користувачів. Особливо важливо визначити ці вимоги, якщо перед виконавцем ви ставите завдання не тільки розміщення сайту, але і розробки дизайну або лейаутов сторінок.

Ключовим модулем у всіх CMS є модуль управління контентом. Щодо публікації, то цей модуль з контенту формує остаточний вигляд сторінки. Остаточне відображення сторінки має управлятися через таблиці стилів, що забезпечить гнучкість і розширюваність. Загальний шаблон сторінок вже повинен бути заданий, і управління ним повинно бути простим і зрозумілим.

Модульність, розширюваність і простота в управлінні – це основні вимоги до будь-якого веб-проекту. Сама CMS повинна забезпечувати лише базовий функціонал (управління сторінками, структурою сайту і редагування інформації на ньому), який в міру розвитку спроможний розширюватися [9].

Основна вимога - це гнучка конфігурація сайту за допомогою функціональних модулів. Вони повинні розширювати функціонал сайту в будь-яких межах, - від сайту-візитки, до Інтернет магазину.

Дуже важливо робити адміністрування сайту максимально простим і зрозумілим, щоб клієнт вже через півгодини самостійно міг додавати сторінки, редагувати інформацію, управляти розділами і меню на сайті. Тому, треба максимально спростити процес адміністрування, залишивши лише необхідні функції, які б знадобилися недосвідченому власнику сайту[14].

Список базових функцій (операцій) адмін-панелі CMS:

– загальні налаштування сайту;
– створення сторінок;
– управління сторінками (редагування властивостей і змісту, видалення);
– управління розділами (додавання, редагування властивостей);

– управління меню (додавання, редагування посилань);
– редактор дизайну (візуальний редактор для шаблонів HTML);
– робота з модулями (управління настройками модулів).

Цей функціонал повинен задовольнити більшість користувачів (адміністраторів), тому, варто сфокусуватися на зручному інтерфейсі і ергономіці, не навантажуючи її зайвими елементами.

Ключовою інформаційною сутністю систем керування контентом сайтів є сторінка або елемент контенту. Як правило, в базі даних CMS-системи існує спеціальна таблиця, що представляє дану сутність, характеризуючи її певним набором полів. Основними полями загального елемента контенту є заголовок, html-текст, анотація та ін. В залежності від системи по-різному можуть бути реалізовані ієрархічні та структурні зв’язки між елементами контенту. Для реалізації CMS пропонується створити два базові класи – клас системи (ядро) та клас сторінки. Уся робота із сторінками сайту відбувається за допомогою класу сторінки. Клас ядра відповідає за загальну логіку роботи системи та обробку ключових запитів[5].

Незважаючи на велике розмаїття програмних систем, що застосовуються для керування контентом, актуальним завданням залишається дослідження, розробка та вдосконалення засобів універсального керування інформаційними об’єктами Web-ресурсів, що дозволить спростити створення нових ресурсів різного призначення, а також забезпечить ефективні механізми їх супроводження та налаштування.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ КЕРУВАННЯ КОНТЕНТОМ ІНТЕРНЕТ-МАГАЗИНУ

2.1. Архітектура веб-орієнтованої інформаційної системи.

Архітектуру інформаційної системи можна описати як концепцію, що визначає модель, структуру, виконувані функції й взаємозв'язок компонентів інформаційної системи[19].

У найзагальнішому вигляді архітектуру систем управління Web-контентом можна представити таким чином (рис. 2.5).

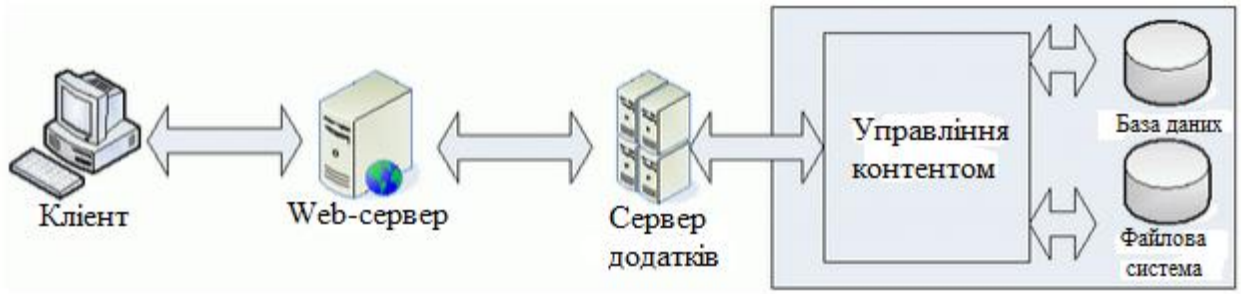


Рисунок 2.5. – Архітектура CMS

PHP (Hypertext Preprocessor (Препроцесор гіпертексту) – це широко використовувана мова сценаріїв загального призначення з відкритим вихідним кодом. Саме на цій мові буде написана CMS Інтернет магазину.

Синтаксис мови бере початок з C, Java і Perl. PHP досить простий для вивчення. Перевагою PHP є надання web-розробникам можливості швидкого створення динамічних web-сторінок[6].

Робота по керуванню спеціалізованими інформаційними об'єктами в контексті CMS передбачає опис нового типу контенту як нової сутності в інформаційній системі, що повинно включати:

– загальний опис створюваної сутності, що включає такі характеристики як ім'я для програмного опрацювання та підпис для відображення користувачам;

– опис набору полів, які повинні характеризувати відповідні інформаційні об’єкти, що включає власне перелік полів, а також опис кожного окремого поля із зазначенням його імені, підпису, типу даних та ін. інформації;

– створення інфраструктури для збереження екземплярів новостворюваної сутності, тобто безпосередніх об’єктів із значеннями їх полів;

– налаштування способу відображення об’єктів на сайті, їх адміністрування, а також будь-яких інших засобів роботи з ними.

Таким чином спеціалізовані об’єкти в БД CMS декомпонуються на такі сутності, які повинні певним чином відображатися в системі:

– тип об’єкту;

– поля об’єкту;

– об’єкти(або екземпляри);

– значення полів об’єктів.

У випадку, коли для створення об’єкта застосовується спосіб приєднання до контенту, роль «об’єкту» виконуватиме сторінка загального типу, до якої буде приєднано спеціалізовані «значення полів» відповідно до «типу об’єкту». У випадку розмежування спеціалізованих об’єктів та контенту, «об’єкт» буде представлено окремою сутністю в базі даних.

Існують три основні області, де використовується PHP.(рис. 2.6).

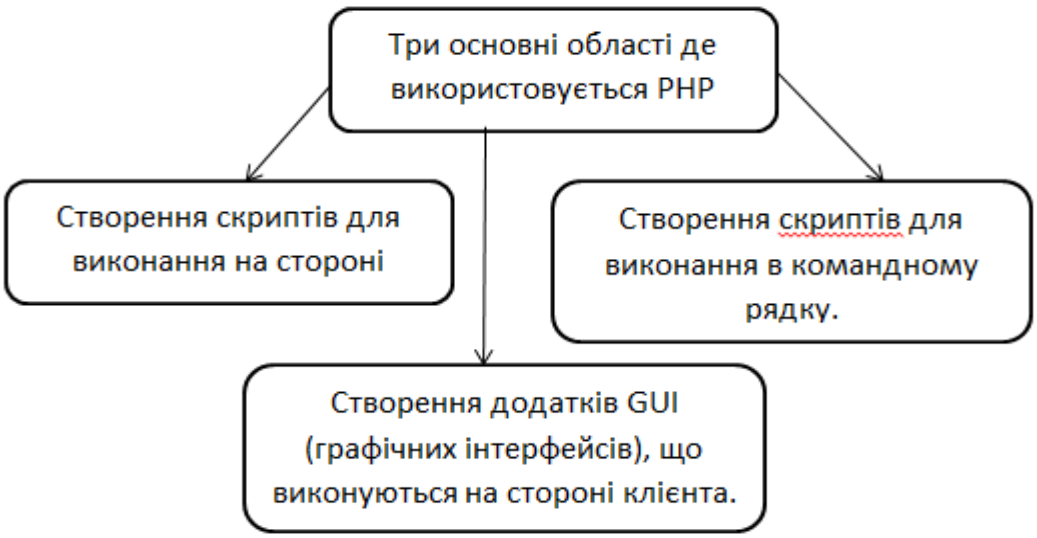


Рисунок 2.6. – Основні області, де використовується PHP

1.Створення скриптів для виконання на стороні сервера. PHP найбільш широко використовується саме таким чином. Все, що знадобиться, це парсер PHP (у вигляді програми CGI або серверного модуля), веб-сервер і браузер. Можливість переглядати результати виконання PHP-скриптів в браузері надають працюючий веб-сервер і встановлений PHP.

CGI (від англ. Common Gateway Interface - «загальний інтерфейс шлюзу») - стандарт інтерфейсу, використовуваного для зв'язку зовнішньої програми з веб-сервером. Програму, яка працює за таким інтерфейсу спільно з веб-сервером, прийнято називати шлюзом, хоча багато хто воліє назви «скрипт» (сценарій) або «CGI-програма».

Оскільки гіпертекст є статичним за своєю природою, веб-сторінка не може безпосередньо взаємодіяти з користувачем. До появи JavaScript, не було іншої можливості відреагувати на дії користувача, крім як передати введені їм дані на веб-сервер для подальшої обробки. У разі CGI ця обробка здійснюється за допомогою зовнішніх програм і скриптів, звернення до яких виконується через стандартизований інтерфейс - загальний шлюз [30].

Узагальнений алгоритм роботи через CGI можна представити в наступному вигляді:

1) клієнт запитує CGI-додаток за його URI;

2) веб-сервер приймає запит і встановлює змінні оточення, через них з додатком передаються дані та службова інформація;

3) веб-сервер перенаправляє запити через стандартний потік введення (stdin) на вхід спричиненої програми;

4) CGI-додаток виконує всі необхідні операції і формує результати у вигляді HTML;

5) сформований гіпертекст повертається веб-сервера через стандартний потік виводу (stdout). Повідомлення про помилки передаються через stderr;

6) веб-сервер передає результати запиту клієнта.

2. Створення скриптів для виконання в командному рядку. Створюється PHP-скрипт, здатний запускатися незалежно від веб-сервера та браузера. Все, що буде

потрібно - парсер PHP. Такий спосіб використання PHP ідеально підходить для скриптів, які повинні виконуватися регулярно, наприклад, за допомогою утиліти cron [планувальник завдань] (на платформах Unix або Linux) або за допомогою планувальника завдань (Task Scheduler) на платформах Windows. Ці скрипти також можуть бути використані в задачах простої обробки текстів.

3. Створення додатків GUI (графічних інтерфейсів), що виконуються на стороні клієнта.

Графічний інтерфейс (зазвичай вимовлений GOO-ee) - це графічний (а не суто текстовий) інтерфейс для комп'ютера. Коли ви читаєте це, ви дивитеся на графічний інтерфейс або графічний інтерфейс вашого конкретного веб-браузера. Термін набув чинності тому, що перші інтерактивні інтерфейси користувача з комп'ютерами не були графічними; вони були орієнтованими на текстову клавіатуру і складалися, як правило, з команд, які вам доводилося запам'ятати, та відповідь на комп'ютері, яка була назівально короткою. Командний інтерфейс операційної системи DOS (який ви все ще можете отримати з операційної системи Windows) є прикладом типового інтерфейсу користувача-комп'ютера, перш ніж графічні інтерфейси з'являться. Проміжний крок у користувальницьких інтерфейсах між інтерфейсом командного рядка та графічним інтерфейсом був неграфічним інтерфейсом на основі меню, який дозволив вам взаємодіяти за допомогою миші, а не за допомогою введення команд клавіатури.

MVC описує простий спосіб побудови структури додатка (рис. 2.7), метою якого є відділення бізнес-логіки від призначеного для користувача інтерфейсу. В результаті, додаток легше масштабується, тестується, супроводжується і звичайно ж реалізується.

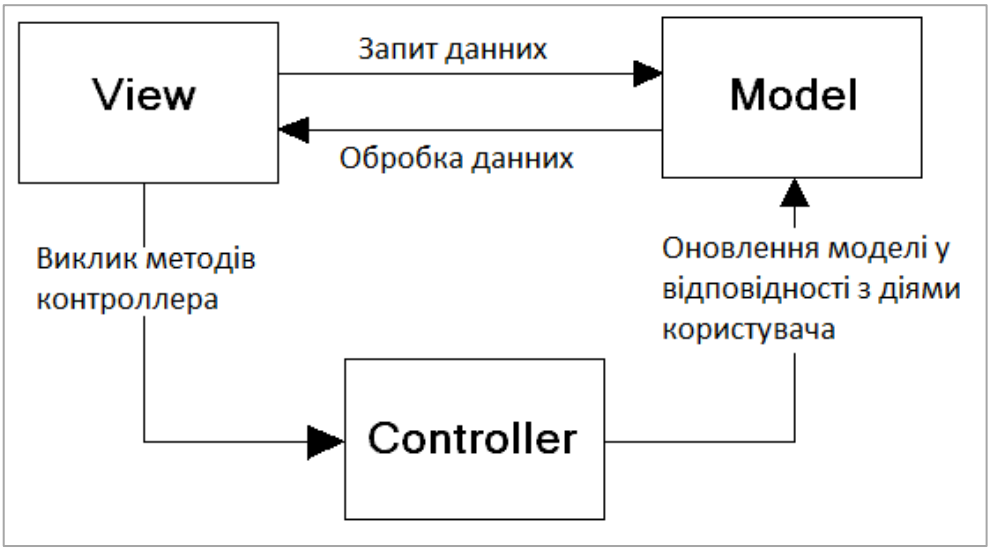


Рисунок 2.7. – Спосіб побудови структури додатка

В архітектурі MVC модель надає дані і правила бізнес-логіки, уявлення відповідає за користувальницький інтерфейс, а контролер забезпечує взаємодію між моделлю і представленням.

Модель – містить бізнес-логіку додатка і включає методи вибірки (це можуть бути методи ORM), обробки (наприклад, правила валідації) і надання конкретних даних, що часто робить її дуже товстою, що цілком нормально.

Модель не повинна безпосередньо взаємодіяти з користувачем. Всі змінні, що відносяться до запиту користувача повинні оброблятися в контролері.

Модель не повинна генерувати HTML або інший код відображення, який може змінюватися в залежності від потреб користувача. Такий код повинен оброблятися в видах.

Одна і та ж модель, наприклад: модель аутентифікації користувачів може використовуватися як в призначеній для користувача, так і в адміністративній частині програми. В такому випадку можна винести загальний код в окремий. Вид - використовується для завдання зовнішнього відображення даних, отриманих з контролера і моделі.

Види містять HTML-розмітку і невеликі вставки PHP-коду для обходу, форматування і відображення даних.

Моделі повинні звертатися до бази даних, а контролер в свою чергу має працювати з даними, отриманими із запиту користувача.

Може безпосередньо звертатися до властивостей і методів контролера або моделей, для отримання готових до висновку даних.

Види зазвичай поділяють на загальний шаблон, що містить розмітку, загальну для всіх сторінок (наприклад, шапку і підвал) і частини шаблону, які використовують для відображення даних, що виводяться з моделі або відображення форм введення даних.

Контролер – сполучна ланка, що з'єднує моделі, види і інші компоненти в робоче додаток. Контролер відповідає за обробку запитів користувача. Контролер не повинен містити SQL-запитів. Їх краще тримати в моделях. Контролер не повинен містити HTML і інший розмітки. Її варто виносити в види.

У добре спроектованому MVC-додатку контролери зазвичай дуже тонкі і містять тільки кілька десятків рядків коду. Чого, не скажеш про Stupid Fat Controllers (SFC) в CMS Joomla. Логіка контролера досить типова і велика її частина виноситься в базові класи.

Моделі, навпаки, дуже товсті і містять велику частину коду, пов'язану з обробкою даних, тому що структура даних і бізнес-логіка, що міститься в них, зазвичай досить специфічна для конкретного додатка[41].

Для розробки веб-додатку використовуємо операційну систему Windows 10.

Щоб почати розробку веб-орієнтованої системи потрібно встановити таке програмне забезпечення: Apache HHTP Server, MySql, PHP, PHPMyAdmin, PhpStorm.

В якості БД веб-програми будемо використовувати MySql. MySQL - найпопулярніша у світі база даних з відкритим кодом. Завдяки своїй перевіреній продуктивності, надійності та простоті використання, MySQL став провідним вибором бази даних для будь-яких веб-застосунків, починаючи від персональних веб-сайтів та невеликих Інтернет-магазинів до великомасштабних, високопрофесійних веб-операцій, таких як Facebook, Twitter і YouTube. MySQL, як

будь-який інший СУБД є программу-сервер, яка зараз переживає пам'яті комп'ютера та обслуговує TCP порт[24].

Проведемо аналіз існуючих CMS.

Drupal, Joomla та WordPress – безкоштовні системи управління, на основі яких можна створювати функціональні і легко керовані сайти без серйозних фінансових витрат. Спробуємо провести аналіз і порівняти ці системи між собою. Ми вибрали найбільш важливі аспекти, які швидше за все зацікавлять розробників, які планують побудувати програмний продукт на базі cms / cmf рішень. Таблиця 2.1.

Таблиця 2.1. Порівняння найвідоміших CMS

№	CMS	Офіційний сайт CMS	Простота наповнення (1-10)	Захист (1-10)	Вартість розробки (1-10)	Простота доопрацювання функціоналу (1-10)	Вбудоване SEO (1-10)	Можливості доопрацювання SEO
1	Joomla(2.5-3.0)	Joomla.com	3	+/-	5	7	+/-	+
2	Wordpress	Wordpress.org	2	+/-	5	7	+	+
3	Drupal	www.drupal.com	7	+	9	3	+/-	+
4	OpenCart	www.opencart.com	3	+/-	8	4	+/-	+
5	MODx	Modx.com	1	+/-	7	6	+	+
6	PrestaShop	Prestashop.com	8	+	10	8	-	+
7	Magneto	Magnetocommerce.com	5	+	10	10	-	+

Преваги CMS Joomla:

– проста і зручна в адмініструванні

– універсальна, підходить для вирішення безлічі завдань

– існує дуже багато плагінів, модулів, розширень

– чи не складне додавання своїх модулів

– багатий вибір безкоштовних і платних шаблонів

– часті оновлення самого движка і додаткових модулів для функціонування сайту

– початківець користувач зможе самостійно і швидко освоїти адміністративну панель сайту

– Недоліки CMS Joomla:

– Падає швидкість при дуже великих розмірах сайту (наприклад якщо використовувати сайт в ролі Інтернет-магазину або блогу)

– Часто піддається до атак хакерів, вимагає додаткової настройки безпеки

– При просуванні сайту потрібна ретельна і тонка настройка

WordPress – це найкраща CMS для ведення своїх особистих блогів і для новинних сайтів. На ній також можна створювати і ресурсовитратні сайти, такі як: Інтернет-магазини, онлайн каталоги, бізнес сайти. З легкістю може впоратися з простеньким сайтом-візиткою. Аналіз cms сайту на WordPress допоможе зробити розуміння плюсів і мінусів цього движка.

Переваги CMS WordPress:

– широкий вибір безкоштовних і платних шаблонів, що дозволити підібрати унікальний стиль для Вашого сайту.

– зручний графічний інтерфейс управління сайтом. Після короткого навчання Ви будете себе почувати як риба в воді.

– широкий спектр готових і перевірених часом модулів

– відмінна продуктивність, природно, при невеликих обсягах наповнення сайту.

Недоліки CMS WordPress:

– повільна швидкість переходу по посиланнях, довге завантаження фотографій. На хороших хостингах цього можна і не побачити, тому що цей "движок" дуже ресурсномісткий і на поганому хостингу проблема стане помітнішою.

– за неофіційними даними – злом системи не становить особливих труднощів. Хороші розробники знають як боротися з подібними нападами хакерів.

– SEO компонент забезпечить хороші позиції у видачі Google або Yandex.

– Продовжує аналіз cms систем движок для сайту Drupal. Система управління контентом Drupal відрізняється від інших найпотужнішим двигуном, який здатний

витримати великі навантаження. Більш детальне порівняння cms можна виконати тільки при розумінні всіх позитивних і негативних якостей движка.

Переваги CMS Drupal:

– мультизадачність, яка забезпечується цією системою просто вражає. Відкриває можливість для створення сайтів самих різних тематик від сайтів-візиток до повноцінних гігантних Інтернет-магазинів.

– ком'юніті - це те, без чого не зможе розвиватися ніяка CMS. Кожен викладений модуль або доповнення, крім того, що мають офіційну документацію ще й проходять випробування перед тим як опублікуватися на сайті.

– вважається одним з кращих серед своїх братів CMS, якщо Ваш сайт буде відрізнятися високою завантаженістю користувачів вибирайте Drupal.

– Недоліки CMS Drupal:

– дуже складна для адміністрування, без читання документації звичайний користувач не зможе впоратися навіть з банальними завданнями.

– на порядок вище вартість розробки сайтів на Drupal'e

– колосальне навантаження на сервер. Вимагає дуже потужного устаткування від хостингу.

– відсутні вбудовані модулі для SEO просування.

– вимагає наявності хорошої технічної підтримки.

Крім того, що drupal є CMS він одночасно є платформою для розробки нового функціоналу і передбачає легку адаптацію під уже створений функціонал. Отже є CMF. Drupal (друпал) - система управління сайтом, написана на мові PHP і використовує як сховище даних реляційну базу даних (підтримуються MySQL, PostgreSQL та інші). Drupal є вільним програмним забезпеченням, захищеним ліцензією GPL, і розвивається зусиллями ентузіастів зі всього світу[35].

Drupal відкрита система, що дозволяє легко і невимушено розширювати можливості, що має великий репозитарій створеного фахівцями функціоналу, що масштабується для побудови будь-якого ресурсу. Рекомендована кращими фахівцями в області розробки і збірки повномасштабних сайтів.

Що стосується MySQL, номером порту буде число 3306. А клієнтська програма, чи це CGI-додаток на Perl або програмний продукт, сполучається з СУБД у цій порту і посилає йому рядки на SQL. Той, у своє чергу їх інтерпретує, виконуючи необхідне, і відсилає результати запиту назад клієнту. У такий спосіб відбувається спілкування серверу баз даних із клієнтськими програмами[24].

Apache - це найпоширеніше програмне забезпечення для веб-серверів. Розроблений та підтримується Apache Software Foundation, Apache - це програмне забезпечення з відкритим кодом, доступне безкоштовно. Він працює на 67% всіх веб-серверів у світі. Він може бути надзвичайно індивідуальним для задоволення потреб багатьох різних середовищ, використовуючи розширення та модулі. Більшість постачальників хостингу WordPress використовують Apache як програмне забезпечення для веб-серверів. Тим не менше, WordPress також може працювати на іншому веб-серверному програмному забезпеченні.

Apache HTTP Server підтримує модульність. Існує більше 500 модулів, що виконують різні функції. Частина з них розроблюється командою Apache Software Foundation, але основне число - окремими відкритими вихідними кодами-розробниками. Для того, щоб Apache краще шукав файли PHP при запиті директорії, а не файли HTML (на даний час в першу чергу він буде шукати файл з назвою index.html).

Модулі включаються до складу сервера в момент компонування, так і завантажені динамічно, через директиви конфігураційного файлу.

В модулях реалізуються такі речі, як:

– підтримка мов програмування.
– додавання функцій.
– виправлення помилок або модифікація основних функцій.
– посилення безпеки.

PHPMyAdmin - це веб-додаток, який розповсюджується з відкритим кодом, написаний мовою PHP-web-програмування і представляє собою веб-інтерфейс для адміністрування MySQL-база даних. PHPMyAdmin для роботи з базою даних

потрібен браузер, який і буде передавати на сервер всі команди. В якості мови роботи з БД широко використовуються знання SQL.

PHPMyAdmin широко розповсюджується по всьому світу і є одним з провідних у роботі СУБД. Інтерфейс доступний більш ніж на 60 мовах світу.

За допомогою phpmyadmin:

– створюються і коригуються бази даних, таблиці, записи;

– створюються користувачі;

– з'являється можливість виконувати SQL-команди;

– працює система пошуку по базі даних.

Всі операції з осередками і їх вмістом - перегляд, копіювання, видалення, вставка робляться в один клік. Найчастіше веб-розробниками та адміністраторами сайту використовується можливість миттєвого створення так званого дампа бази (копії) для перенесення сайту з одного хоста на інший. При бажанні його можна отримати одразу архівованим.

Існує можливість – після виконання запиту програма показує не тільки детальну інформацію про нього (час обробки, кількість порушених рядів), а й пропонує показати розширену інформацію про використаних даних при побудові запиту (індексах та інше, що може бути корисно при побудові складних запитів, налагодженні і ручної оптимізації).

Ще одна функція дозволяє по введеному запиту автоматично будувати код на PHP. Хоча результуючий код дуже простий - всього лише змінна sql, яка містить код запиту, оптимізований під синтаксис PHP, але це дуже корисна функція, особливо якщо спочатку налагоджувати складний запит через інтерфейс phpMyAdmin-а, а потім, переконавшись в конкретній видачі результатів, перенести запит в скрипт.

PhpStorm – це інтегроване середовище розробки на PHP з інтелектуальним редактором, який глибоко розуміє код, підтримує PHP 7.2-5.3 для сучасних і класичних проектів, забезпечує краще в індустрії автодоповнення коду, рефакторинг, запобігання помилок нальоту і підтримує змішування мов.

Ключові можливості:

– підтримує PHP 7.2-5.3, генератори, співпрограми і все синтаксичні поліпшення;

– PHP рефакторингом, code (re) arranger, детектор дубльованого коду

– підтримка Vagrant, Composer, вбудований REST клієнт, Command Line Tools, SSH консоль;

– підтримка фреймворків (MVC view для Symfony2, Yii) і спеціалізовані плагіни для провідних PHP фреймворків (Symfony, Magento, Drupal, Yii, CakePHP, WordPress, Joomla! І багато інших);

– візуальний відладчик для PHP додатків, валідація конфігурації відладчика, PHPUnit з покриттям коду (підтримка PHPUnit 6), а також інтеграція з профілювальником HTML, CSS, JavaScript редактор. Налагодження і модульне тестування для JS. Підтримка HTML5, CSS, Sass, SCSS, Less, Stylus, Compass, CoffeeScript, TypeScript, ECMAScript Harmony, Emmet і інших передових технологій веб-розробки;

– повний набір інструментів для фронтенд-розробки[39].

Рівень розвитку сучасних технологій настільки високий, що дозволяє побудувати інформаційну систему будь-якого масштабу, складності та функціональності. Однак, з огляду на вимоги бізнесу, засновані на показниках різних бізнес оцінок, виникають додаткові складності, вирішення яких зводиться до забезпечення раціонального підходу до процесу проектування, реалізації і подальшій експлуатації інформаційних систем. Виходячи з цього, можна однозначно вважати вибрану архітектуру одним із основних показників ефективності створюваної інформаційної системи, а, отже, і успішності бізнесу.

2.2. Склад функціональної частини.

Автоматизована інформаційна система, як і будь-яка інша складна система, складається з окремих елементів і організації їх взаємозв’язку. В автоматизованій інформаційній системі виділяють функціональну частину та частину забезпечення,

які, у свою чергу, поділяються на простіші елементи-підсистеми, котрі також припускають подальший поділ.

Розвиток комп'ютерної інформаційної технології нерозривно пов'язаний з розвитком інформаційних систем, які в економіці використовуються для автоматизованого (людино-машинного) розв'язування економічних задач. Для розв'язування будь-якої задачі за допомогою комп'ютера необхідно створити інформаційне забезпечення.

Функціональна підсистема – це відносно самостійна частина системи, виділена за спільністю функціональних ознак управління. В автоматизованих інформаційних системах функціональні підсистеми, як правило, виокремлюють за такими ознаками:

– стадіями управління (прогнозування, планування, облік, звітність, аналіз, контроль тощо);

– видами основної діяльності (доходи, видатки, трансферти тощо);

– організаційною структурою (структурні підрозділи); - функціональною ознакою (виконувані функції).

Функціональна ознака декомпозиції АІС визначає призначення підсистеми, тобто для якої сфери діяльності вона призначена і які основні цілі, задачі та функції виконує.

Здійснювати декомпозицію функціональної частини АІС — поділяти цю систему на підсистеми, комплекси задач і окремі задачі — можна як за окремими ознаками, так і за їх сукупністю.

Загальна схема інтернет-магазину

Через мережу Інтернет покупець за допомогою браузера заходить на web-сайт інтернет-магазину. Web-сайт містить електронну вітрину, на якій представлені каталог товарів (з можливістю пошуку) і необхідні інтерфейсні елементи для введення реєстраційної інформації, формування замовлення, проведення платежів через інтернет, оформлення доставки, отримання інформації про компанії-продавця і online допомоги.

Реєстрація покупця виробляється або при оформленні замовлення, або при вході в магазин. Після вибору товару від покупця потрібно заповнити форму, в якій вказується, яким чином буде здійснено оплату і доставка. Для захисту персональної інформації взаємодія має здійснюватися по захищеному каналу. По завершенню формування замовлення і реєстрації вся зібрана інформація про покупця надходить з електронної вітрини в торгову систему інтернет-магазину. У торговельній системі здійснюється перевірка наявності затребуваного товару на складі, ініціюється запит до платіжної системи. При відсутності товару на складі направляється запит постачальнику, а покупцю повідомляється про час затримки.

У тому випадку, якщо оплата здійснюється при передачі товару покупцеві: кур'єром або післяплатою - необхідно підтвердження факту замовлення. Найчастіше це відбувається за допомогою електронної пошти або по телефону.

При можливості оплати через інтернет, підключається платіжна система (PayPal, EasyPay, WebMoney і т.п.). Після повідомлення про проведення online платежу торговельною системою формується замовлення для служби доставки.

Торгові системи електронних магазинів на практиці рідко бувають повністю автоматизованими. Повністю автоматизованими вони бувають лише в тих випадках, коли інтернет-магазин зроблений професійно і коли в ньому відсутні будь-які помилки. Легкість здійснення покупки часто призводить до збільшення кількості помилок користувача (особливо при низькоякісному дизайні електронного магазину і відсутності online допомоги) - а це суттєві втрати для магазину. Тому потрібна перевірка менеджером кожного окремого факту замовлення. Виняток становить лише продаж інформаційного продукту (файли, архіви і т.п.), який можна доставити з мінімальними витратами безпосередньо через інтернет.

Можна стверджувати, що основні проблеми електронної комерції лежать на стиках інтернету і реальної діяльності. У звичайній торгівлі покупець звик до того, що є можливість оцінити товар візуально, визначити його якість і характеристики. В електронній торгівлі він такої можливості позбавлений. Максимум, на що він може розраховувати, це фотографія товару і перерахування його характеристик. Найчастіше цієї інформації достатньо, але тут вступають в дію емоційні і

психологічні чинники. Більшість електронних магазинів мають проблеми з доставкою товарів, особливо якщо ціна товару невелика. Проблеми також виникають при необхідності оплатити товар в електронному магазині. Для цього є безліч причин: недовіру громадян по відношенню до банківської системи в цілому і безготівковим платежам зокрема, неврегульованість організаційних і правових питань електронних платежів, невпевненість в безпеці проведення транзакцій через інтернет.

У загальному випадку технічну сторону будь-якого інтернет-магазину можна розглядати як сукупність електронної вітрини і торгової системи.

Структуру системи управління контентом для веб-сайту магазину, що розробляється, можна подати у вигляді інтелект-карти (рис. 2.8).

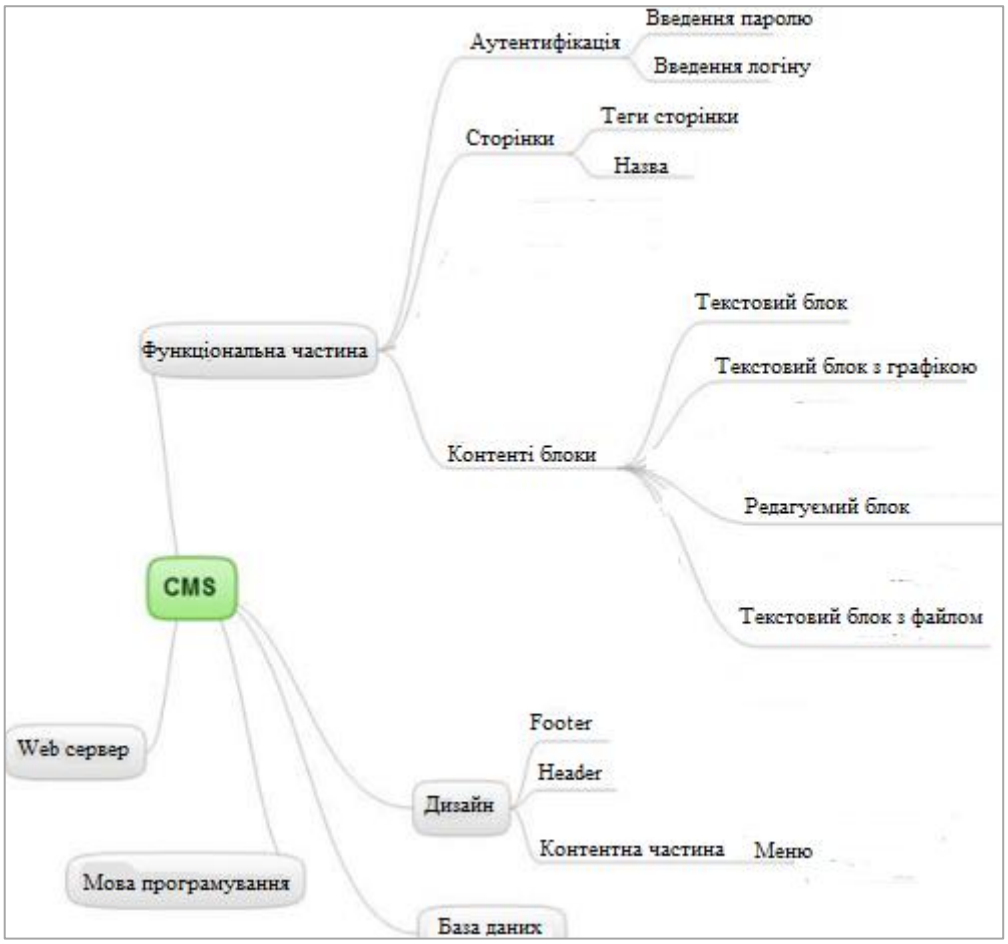


Рисунок 2.8 – Структура системи управлінням контентом

Електронна вітрина призначена для виконання наступних завдань:

– надання інтерфейсу до бази даних товарів, що продаються у вигляді каталогу або прайс-листа;

– робота з електронною "кошиком" покупця;

– реєстрація покупців;

– оформлення замовлень з вибором методу оплати і доставки;

– надання online допомоги покупцеві;

– збір маркетингової інформації;

– забезпечення безпеки особистої інформації покупців;

– автоматична передача інформації в торговельну систему.

Вітрина електронного магазину розташовується на інтернет-сервері і являє собою Web-сайт з активним вмістом.

Основа вітрини електронного магазину - каталог товарів із зазначенням цін, який може бути структурований різними способами (за категоріями товарів, по виробникам), містити повну інформацію про характеристики кожного товару, а також його зображення. Вибравши товар, який сподобався, користувач поміщає його в "кошик". У будь-який момент до остаточного оформлення замовлення покупець може відредагувати вміст кошика і кількість товарів кожного виду.

Процес реєстрації може ініціюватися системою до або після вибору товарів з каталогу. Введення реєстраційних даних після вибору товарів дозволяє покупцеві заощадити час в тому випадку, якщо він не прийняв рішення що-небудь купити в цьому електронному магазині.

Найчастіше електронна вітрина і є власне інтернет-магази-ном, а друга важлива частина, електронна торговельна система, просто відсутня. Всі запити покупців надходять не в автоматизовану систему обробки замовлень, а до менеджерів з продажу. Далі бізнес-процеси

Для розв’язування будь-якої задачі з допомогою комп’ютера необхідно створити інформаційне забезпечення (забезпечити розрахунки потрібними даними) і математичне забезпечення (створити математичну модель розв’язування задачі, за якою складається програма для ПК). Спрощену схему автоматизованого

розв’язування економічної задачі (наприклад, розрахунок оптимальної виробничої програми) зображено на (рис. 2.9).

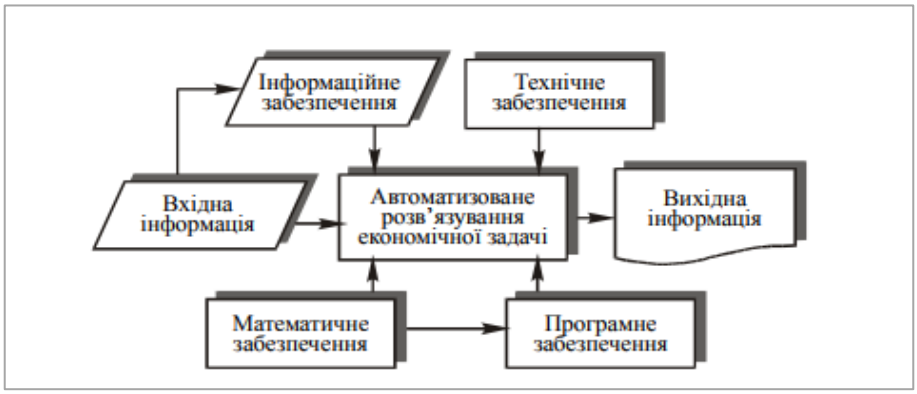


Рисунок 2.9. – Схема автоматизованого розв’язування економічних задач

У системах обробки інформації головними її компонентами є дані та обчислення. Більшість інформаційних систем управління інформаційними ресурсами в організаціях містять і багато інших компонентів, таких як вимоги, запити, трігери і звіти. І всі вони, зокрема, містять великі описи свого власного змісту в тій чи іншій формі. Ці описи необхідні для інтерпретації і для коректного використання наданої інформації.

Відповідно до виділених функціональних підсистем та з урахуванням вимог управління визначається склад задач, що розв'язуються. У функціональних задачах, для автоматизації яких призначена підсистема, відображаються специфічні особливості кожної конкретної підсистеми певного об'єкта. Саме задача є об'єктом розробки, впровадження та експлуатації кінцевим користувачем.

Найбільшою мірою функціональність АІС забезпечується за допомогою принципу модульності. Модуль являє собою цілісну групу елементів системи, яка описана тільки своїми входами і виходами. Модуль як частина системи відображає зв'язок між елементами, тобто обмін інформацією.

IDEF0 - Function Modeling - методологія функціонального моделювання та графічна нотація, призначена для формалізації і опису бізнес-процесів. Відмінною особливістю IDEF0 є її акцент на підпорядкованість об'єктів. В IDEF0 розглядається

логічні відносини між роботами, а не їх тимчасова послідовність. IDEF0-модель дає можливість отримати точну специфікацію всіх операцій і дій, здійснюваних в бізнес-процесі, а також характер взаємозв'язків між ними.

Методологія IDEF0 може використовуватися для моделювання широкого кола автоматизованих або неавтоматизованих «систем» або предметних областей, включаючи будь-які можливі комбінації апаратного та програмного забезпечення, машин, процесів і людей. При створенні нових систем IDEF0 можна використовувати спочатку для завдання вимог до системи і її функцій, а потім для розробки власне системи, яка відповідає заданим вимогам і виконує задані функції. При роботі з існуючими системами за допомогою IDEF0 можна аналізувати виконувані системою функції і документувати механізми (засоби), якими це досягається.

На рисунку 2.10. приведена структурно-функціональна модель системи керуванням контенту сайтом.

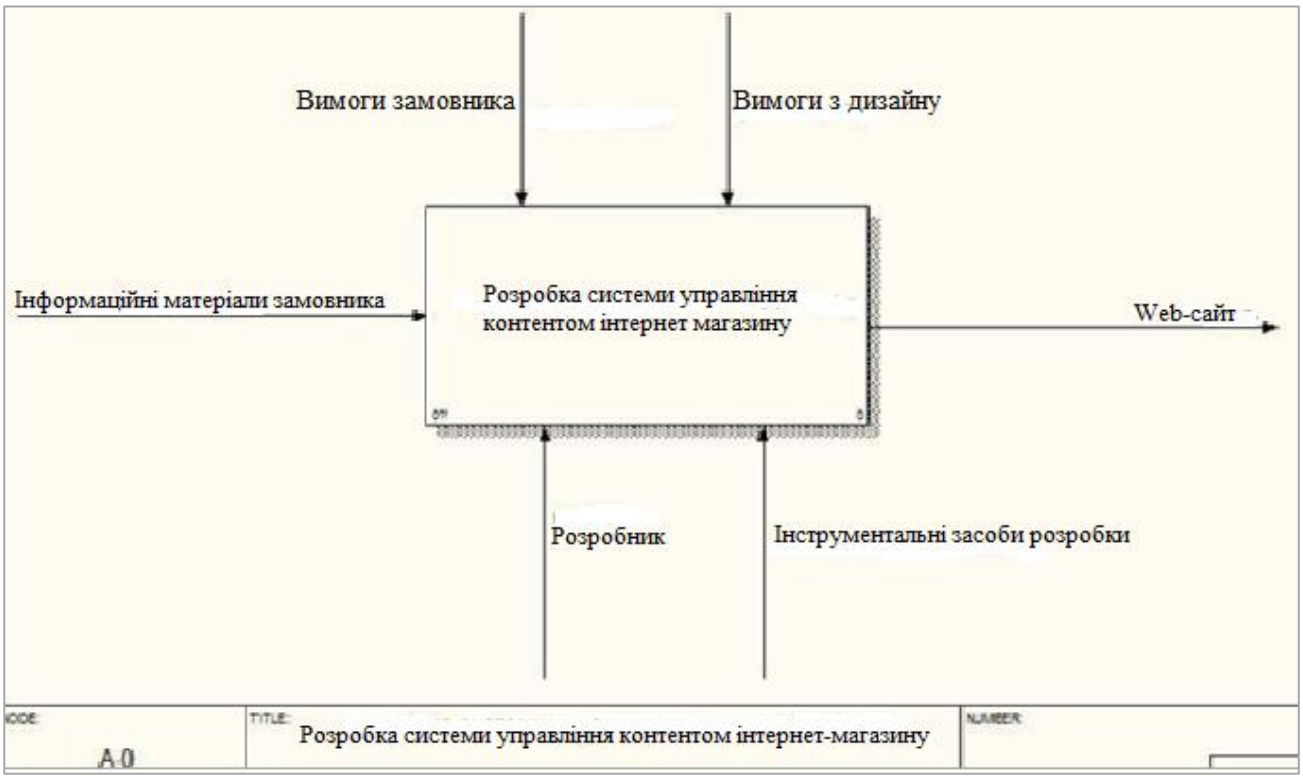


Рисунок 2.10 – Модель системи управління контентом сайту в нотації IDEF0

Таблиця 2.2. Опис основних елементів діаграми

Назва стрілки	Опис
Вимоги замовника	Функціонал, який потрібен для сайту.
Вимоги з дизайну	Вигляд інтерфейсу
Інформаційні метеріали замовника	Інформаційні матеріали що будуть розміщені на web-ресурсі
Інструментальні засоби розробки	Задіяні програми для створення
Розробник	Розробник системи управління контентом

На рисунку 2.11 зображена декомпозиція першого рівня діаграми IDEF0.

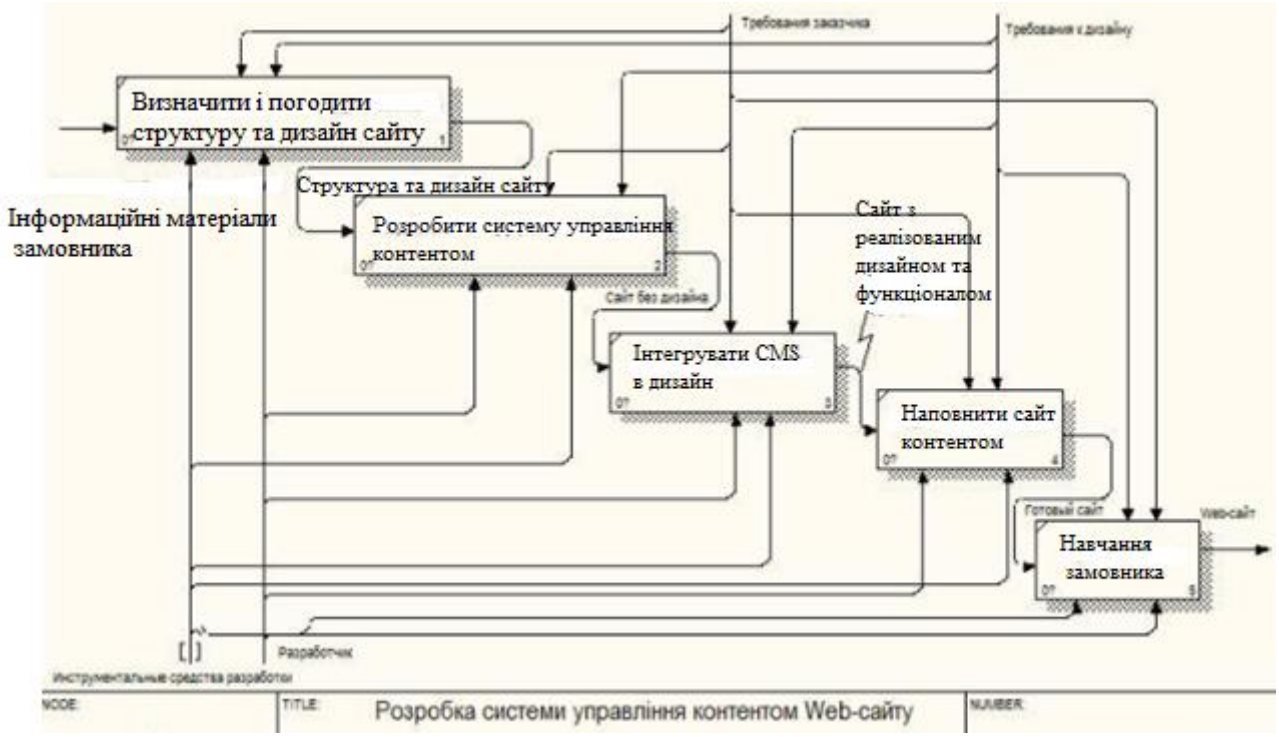


Рисунок 2.11 – Декомпозиція першого рівня

На рисунку 2.12 зображена декомпозиція другого рівня діаграми IDEF0.

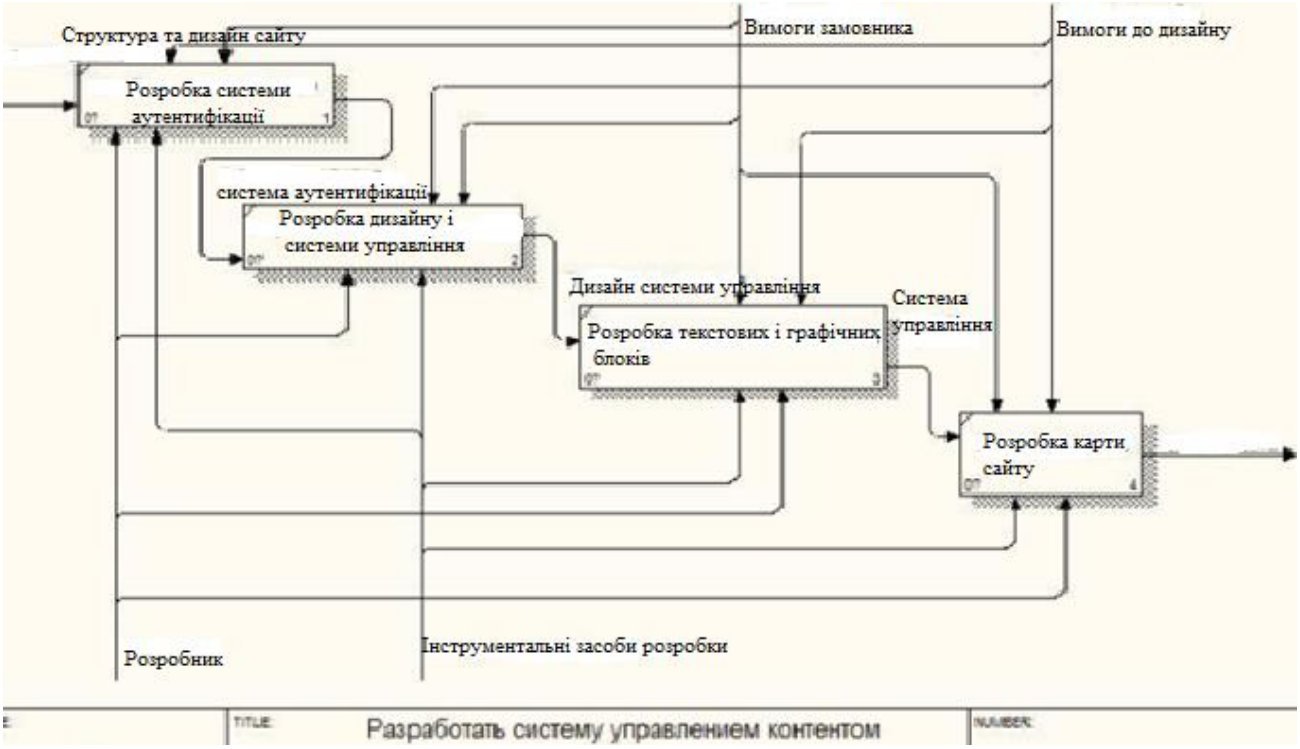


Рисунок 2.12 Декомпозиція другого рівня

У мові UML взаємодія елементів розглядається в інформаційному аспекті їх комунікації, т. е. взаємодіючі об'єкти обмінюються між собою деякою інформацією. При цьому інформація приймає форму закінчених повідомлень. Іншими словами, хоча повідомлення і має інформаційний зміст, воно набуває додаткового властивість надавати спрямований вплив на свого одержувача[40].

Мова UML є общецелевого мову візуального моделювання, який розроблений для специфікації, візуалізації, проектування та документування компонентів програмного забезпечення, бізнес-процесів і інших систем. Мова UML одночасно є простим і потужним засобом моделювання, який може бути ефективно використаний для побудови концептуальних, логічних і графічних моделей складних систем самого різного цільового призначення. Ця мова вібрала в себе найкращі якості методів програмної інженерії, які з успіхом використовувалися впродовж останніх років при моделюванні великих і складних систем[40].

На рисунку 2.13 зображена діаграма взаємодій системи управління контентом сайту.

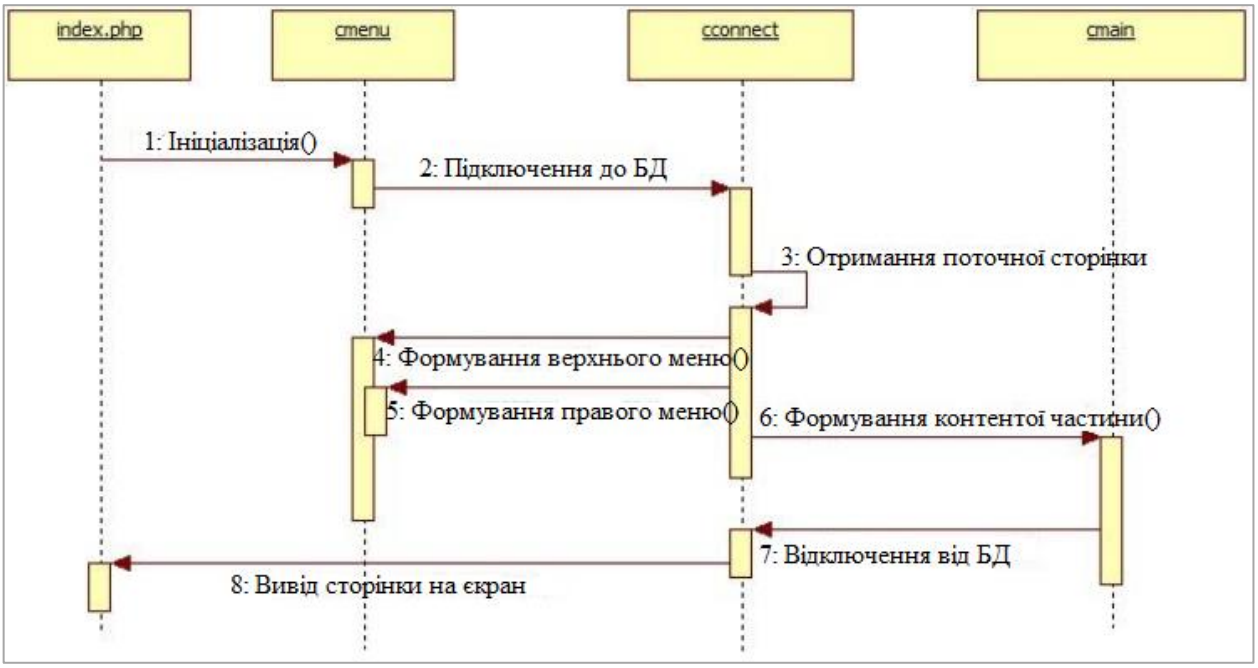


Рисунок 2.13 Діаграма взаємодій

Мова UML заснована на деякому числі базових понять, які можуть бути вивчені і застосовані більшістю програмістів і розробників, знайомих з методами об'єктно-орієнтованого аналізу і проектування. При цьому базові поняття можуть комбінуватися і розширюватися таким чином, що фахівці об'єктного моделювання отримують можливість самостійно розробляти моделі великих і складних систем в самих різних областях додатків.

Конструктивне використання мови UML ґрунтується на розумінні загальних принципів моделювання складних систем і особливостей процесу об'єктно-орієнтованого аналізу і проектування зокрема. Вибір засобів вираження для побудови моделей складних систем зумовлює ті завдання, які можуть бути вирішені з використанням даних моделей. При цьому одним з основних принципів побудови моделей складних систем є принцип абстрагування, який наказує включати в модель тільки ті аспекти проектованої системи, які мають безпосереднє відношення до виконання системою своїх функцій або свого цільового призначення. При цьому всі другорядні деталі опускаються, щоб надмірно не ускладнювати процес аналізу та дослідження отриманої моделі.

Діаграма класів (class diagram) служить для представлення статичної структури моделі системи в термінології класів об'єктно-орієнтованого програмування. Діаграма класів може відбивати, зокрема, різні взаємозв'язки між окремими сутностями предметної області, такими як об'єкти і підсистеми, а також описує їх внутрішню структуру і типи відносин.

2.3 Підсистеми забезпечення функціональної частини.

2.3.1. Програмне забезпечення.

Програмне забезпечення (*software*) – загальне поняття, що вказує на набір кодованих інструкцій (програм) для керування процесором (CPU) комп'ютера. Процесор CPU комп'ютера зчитує такі кодовані інструкції та виконує їх. Виконання програмного забезпечення комп'ютером полягає у маніпулюванні інформацією та керуванні апаратними компонентами комп'ютера. Наприклад, типовим для персональних комп'ютерів є відображення інформації на екран та прийом її з клавіатури.

Програмне забезпечення та апаратне забезпечення є дві комплементарні компоненти комп'ютера, причому межа між ними нечітка: деякі фрагменти програмного забезпечення на практиці реалізуються суто апаратурою мікросхем комп'ютера, а програмне забезпечення, в свою чергу, здатне виконувати функції електронної апаратури. Та по суті призначення програмного забезпечення полягає в керуванні комп'ютером так і іншими програмами та маніпулюванні інформацією [2].

Основні функції операційних систем (ОС) полягають в управлінні ресурсами (фізичними та логічними) і процесами обчислювальних систем. Фізичними ресурсами є: оперативна пам'ять, процесор, монітор, принтер, магнітні та оптичні диски. До логічних ресурсів можна віднести програми, файли, події тощо. Під процесом розуміється деяка послідовність дій, предписана відповідною програмою і використовуваними нею даними.

Початком PHP можна вважати осінь 1994 року, коли Расмус Лердорф (Rasmus Lerdorf) вирішив розширити можливості своєї Home-page (домашньої сторінки) і написати невеликий движок для виконання найпростіших завдань. Такий движок був готовий до початку 1995 року і називався Personal Home Page Tools.

PHP є однією з найбільш широко використовуваних і впізнаваних технологій, що використовуються в Інтернеті. PHP - мова програмування, що використовується на стороні WEB-сервера для динамічної генерації HTML-сторінок. Про це говорить і розшифровка її назви: PHP – Personal HyperText Processor.

PHP – одна з небагатьох мов програмування, створених спеціально для розробки веб-додатків. Тому вона включає в себе всі функції, необхідні саме для роботи на веб-сервері, і при цьому позбавлена надмірності, властивої багатьом її конкурентам.

В декількох словах – на PHP можна зробити усе, що можна зробити за допомогою CGI–програм. Наприклад: обробляти дані з форм, генерувати динамічні сторінки, отримувати та відправляти кукі (cookies)[8].

Крім цього, в PHP включена підтримка багатьох баз даних (databases), неповний перелік вказаний на таблиці 2.3., що робить написання Web-додатків з використанням БД простим..

Таблиця 2.3. – Неповний перелік підтримуваних БД

Adabas D	InterBase	Solid
dBase	mSQL	Syba.se
Empress	MySQL	Velocis
FilePro	Oracle	Unix dbm
Informix	PostgreSQL	

Найпоширенішим використанням PHP є доступ до бази даних, аналіз результатів з цієї бази даних та відображення результатів на веб-сторінці. Ось чому PHP - остання частина загального абревіатуру "LAMP", що означає "Linux, Apache, MySQL і PHP". Встановлення LAMP - це одна з найпоширеніших конфігурацій веб-

сервера і поєднує в собі потужний веб-сервер Apache з PHP та MySQL, що дозволяє створювати вражаючі надійні веб-сторінки та керувати даними[39].

2.3.2 Апаратне забезпечення.

Апаратне забезпечення (англ. hardware) — комплекс технічних засобів, який включає електронний пристрій і, зокрема, ПК: зовнішні пристрої, термінали, абонентські пункти, тощо, які необхідні для функціонування тієї чи іншої системи; фізична частина комп'ютера[27].

Згідно обраних технологій, архітектури, платформи, структури функціональної частини та програмного забезпечення до апаратного забезпечення висуваються вимоги підтримки таких систем:

– PHP-5.3;
– Apache-2.2;
– MySQL-5.5;
– PHPMyAdmin;
– JavaScript.

Для виконання скриптів та їх налагодження необхідно Web-сервер, який зміг би приймати запити користувача і відправляти йому відповідні відповіді. Для того щоб Web-сервер міг виконувати скрипти, написані на PHP, необхідно встановити PHP, а так як ми будемо працювати з базою даних необхідно встановити СУБД (Систему Управління Базами Даних). Одним з найбільш поширених Web-серверів є Apache, а однією з найбільш розповсюджених СУБД є MySQL дивитися таблицю 2.4.

Таблиця. 2.4. – Дистрибутиви для установки Web-сервера з підтримкою PHP і MySQL

Назва программи	Офіційний сайт (звідки можно скачати дистрибутив)	Розмір дистрибутиву
Apache(Web-сервер)	www.apache.org	4,9 М байт
PHP	www.php.net	8,9 М байт
СУБД MySQL	www.mysql.com	16,9 М байт
phpMyAdmin	www.phpmyadmin.net	3,4 М байт
	Разом:	34,1 М байт

Apache – це HTTP веб-сервер. Веб сервер Apache – кросплатформенне програмне забезпечення. Завдяки великій документації і хорошій інтеграції з стороннім ПЗ, Apache отримав величезне поширення. Підтримує наступні ОС – Linux, BSD, Mac OS, Microsoft Windows, Novell NetWare, BeOS.

У Apache 2.0 вводиться поняття мультипроцесорних модулів (MPM). Серед модулів, що надаються з пакетом Apache 2.0, присутня експериментальний модуль perchild, що дозволяє здійснювати віртуальний хостинг при безлічі користувальницьких ідентифікаторів шляхом призначення distributor-потоку для комбінації IP адреса / порт та передачі запитів на satellite-потоки, що виконуються під індивідуальними користувацькими мандатами. На жаль, perchild залишався експериментальним і функціонував тільки у випадку удачі і врешті-решт був видалений з офіційного Apache, починаючи з версії 2.2. Перед цим, розуміючи, що необхідність в наявності стабільного функціонуючого perchild-подібного модуля зберігається, співтовариство Apache розпочало роботу зі створення модулів MPM, здатних заповнити утворився пробіл. Розробка MetuxMPM і його підпроцесу, створеного через fork, peruser, стала продовженням робіт по досягненню цієї мети.

Переваги веб-сервера Apache: підтримка мов програмування PHP, Python, Ruby, Perl, ASP, Tcl; легкість в підключенні зовнішніх модулів; підтримка технологій PHP, CGI і FastCGI; наявність механізмів, які забезпечують безпечність і розмежування доступу даних; можливість використовувати СУБД для аутентифікації користувачів; гнучка і надійна конфігурація системи; підходить для додатків, яким потрібен потужний криптографічний захист даних; можливість створення користувацьких директорій для веб-сайту; можливість налаштування віртуальних хостів, за допомогою яких на одному фізичному сервері можна створити кілька віртуальних; записує протоколи того, що відбувається на вашому сервері; активний зворотний зв'язок з розробниками і своєчасне вирішення виниклих помилок в ПЗ.

Веб-додаток phpMyAdmin – це набір скриптів, написаних на PHP, який надає практично всі необхідні функції по роботі з базами даних MySQL. Додатково до можливостей самого сервера MySQL він надає додаткові функціонал, який дозволяє

більш ефективно і легко працювати з даними. Причому всі функції доступні прямо з браузера, навіть перезавантаження віддаленого сервера (якщо ця можливість дозволена обліковим записом користувача.

Можливості PHPMyAdmin:

– редагування, створення, видалення, перегляд: баз даних, таблиць, їх записів і користувачів;

– ручний і легкий експорт та імпорт БД, таблиць і записів;

– ручний пошук;

– адміністрування БД і користувачів;

– при обробці SQL з'являються підказки та підсвічування синтаксису;

– інформування про дії (в процесі, змінено і тп.).

Ця програма, залишаючись незмінно зручною і практично не змінюючи свого інтерфейсу, постійно модифікується і поліпшується слідом за появою нових версій PHP і MySQL. На сьогоднішній день актуальною версією цього веб-додатку є PHPMyAdmin 4.4.14, що працює спільно з MySQL 5 і PHP 5.2.

2.3.3. Інші засоби забезпечення.

До засобів забезпечення автоматизованих інформаційних систем відносять програмні, технічні (апаратно-програмні), лінгвістичні, правові (законодавчі) та організаційні (адміністративні) засоби.

Програмне забезпечення (ПО) підрозділяється на системне і прикладне ПЗ. Системне ПЗ – це машинно-орієнтоване ПО. Воно реалізовано у вигляді операційної системи, мережевого ПЗ, сервісних програм і систем програмування. Прикладне ПО є проблемно-орієнтованим і реалізуються у вигляді комплексів програм вирішення конкретних завдань.

Організаційні процеси використовуються для управління проектами, створенням інфраструктури та визначенням і оцінкою проекту. Розробка включає в себе всі роботи по створенню ПЗ і його компонентів відповідно до заданих вимог, включаючи оформлення та підготовку матеріалів, необхідних для перевірки

працездатності та відповідної якості програмних продуктів, матеріалів, необхідних для організації.

Технічне забезпечення – комплекс технічних засобів (технічні засоби збору, реєстрації, передачі, обробки, відображення, розмноження інформації, оргтехніки та ін.), Призначених для роботи інформаційної системи, і документація на ці засоби і технологічні процеси.

Управління проектом пов'язано з питаннями планування і організації робіт, створення контролю за строками та якістю виконуваних робіт. Технічне і організаційне забезпечення проекту включає вибір методів і інструментальних засобів для реалізації проекту визначення методів опису проміжних станів розробки, розробку методів і засобів.

Організаційне та методичне забезпечення – це сукупність засобів і методів організації виробництва та управління ним в умовах впровадження інформаційної системи. Воно включає в себе методики проведення робіт, вимоги до оформлення документів, посадові інструкції і т.д.

Інтерфейс мережевий ІТ надає користувачеві засоби теледоступ до територіально розподілених інформаційних та обчислювальних ресурсів завдяки розвинутим засобам зв'язку, що робить інформаційні технології широковживаних і багатофункціональними.

На етапі функціонування інформаційної системи організаційна підтримка вирішує такі завдання:

– впровадження методів управлінських завдань;

– організація персоналу та ансамблю технічних засобів інформаційної системи;

– контроль та аналіз ефективності управління;

– формування пропозицій щодо вдосконалення та розвитку інформаційної системи.

Лінгвістичне забезпечення (ЛЗ) об'єднує сукупність мовних засобів для формалізації природної мови, побудови і поєднання інформаційних одиниць у процесі спілкування персоналу АСУ із засобами обчислювальної техніки. За

допомогою лінгвістичного забезпечення здійснюється спілкування людини з машиною. ЛЗ включає інформаційні мови для описання структури одиниць інформаційної бази АСУ; мови управління та маніпулювання даними інформаційної бази АСУ; мовні засоби інформаційно-пошукових систем; мовні засоби автоматизації проектування АСУ; діалогові мови спеціального призначення та інші мови; систему термінів і визначень, які використовуються в процесі розробки й функціонування АСУ.

Компоненти лінгвістичного забезпечення повинні бути узгодженими з компонентами забезпечення інших видів, бути відносно інваріантними до конкретного вмісту баз даних, надавати в компактній формі засоби для опису всіх об’єктів і процесів заданого для системи класу з необхідним степенем деталізації і без суттєвих обмежень на об’єкт опису, бути розрахований в основному на діалоговий режим їх використання.

Правове забезпечення – це сукупність правових норм, які визначають юридичний статус і функціонування інформаційних систем, регламентують порядок одержання, опрацювання та використання інформації. Будується на базі юридичного підходу, який розглядає соціальне управління, як організаційну, в тому числі виконавчо-розпорядницьку діяльність державних органів, спрямовану на виконання законів та інших нормативних актів.

Правове забезпечення етапів функціонування ІС - інформаційних систем - включає:

– статус ІС;
– права, обов'язки і відповідальність персоналу;
– правові положення окремих видів процесу управління.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ПРОТОТИПУ СИСТЕМИ КЕРУВАННЯ КОНТЕНТОМ ІНТЕРНЕТ - МАГАЗИНУ

3.1 Структура та особливості розробки серверної частини

Для розробки прототипу системи керування контентом Інтернет-магазину по-перше встановлюється Open Server.

Open Server – це портативний локальний WAMP / WNMP сервер, який має багатофункціональну керуючу програму і великий вибір компонентів , що підключаються. Це перший повноцінний професійний інструмент, створений спеціально для веб-розробників з урахуванням їх рекомендацій і побажань.

Для налагодження скриптів в середовищі Open Server пропонує на вибір відразу два види HTTP серверів, різні версії PHP і СУБД модулів, а так само можливість швидкого перемикання між ними.

Дистрибутив Basic – це урізана версія Open Server Basic: базова, мінімальна за розміром, версія Open Server. В Basic версії, так само як і в Premium, відсутній пакет додаткових програм, але нам вистачить і цього, збірці немає модулів Git, ImageMagick, MongoDB, Rockmongo, PostgreSQL і PhpPgAdmin.

Для завантаження необхідно зайти в розділ завантажень сайту Open Server (http://open-server.ru/) і вибрати дистрибутив для завантаження.

Після того, як завантажено інсталяційний файл Open Server на комп'ютер, необхідно вибрати місце для вашого майбутнього локального сервера, запустити скачаний інсталяційний файл і вибирати диск для установки (рис. 3.14). Всі майбутні локальні сайти будуть розташовані в папці domains.

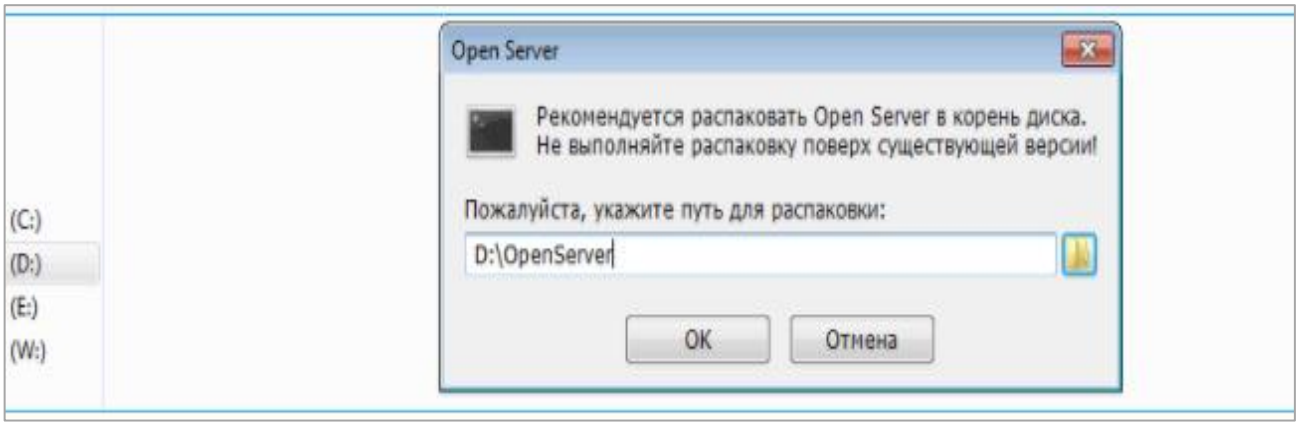


Рисунок 3.14. – Вказаний шлях розпаковки програми

Встановлення продовжується відкриттям папки та обиранням в залежності від розрядності операційної системи версії " Open Server x64.exe " для 64 бітної системи, або ж " Open Server \x86.exe " для 32 бітної системи(рис. 3.15). Також якщо програма запускається вперше, то потрібно встановити Microsoft Visual C++.

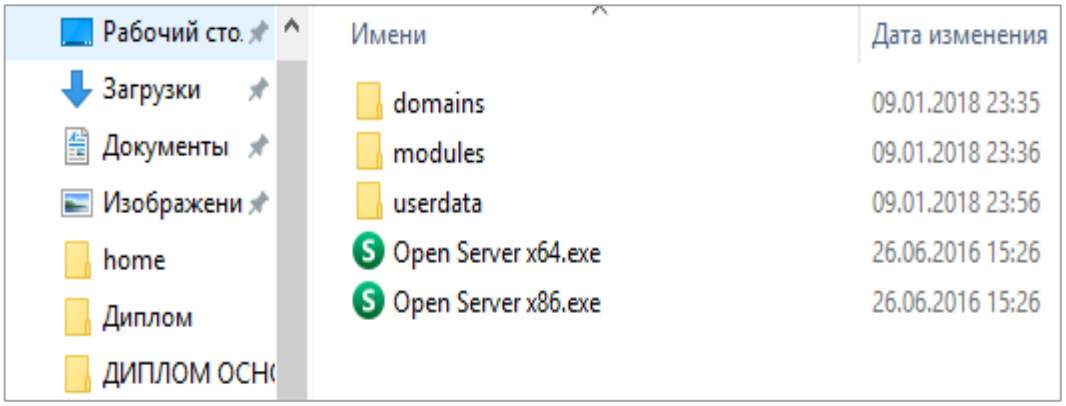


Рисунок 3.15. – Вибір версії програми у відповідності до розрядності системи

Після цього в правому нижньому кутку панелі задач з'явиться значок у вигляді червоного прапора (рис. 3.16).

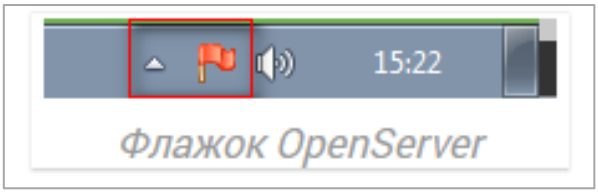


Рисунок 3.16 – Значок встановленої програми Open Server

Запуск системи здійснюється за допомогою вказаного значка (рис. 3.17).

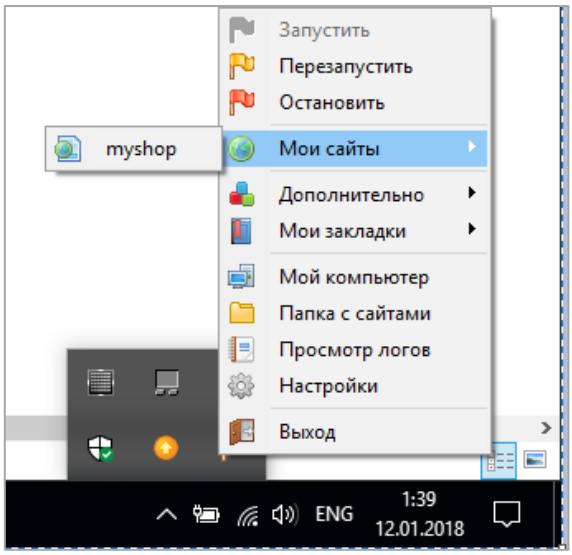


Рисунок 3.17. – Запуск програми Open Server

Створювати базу даних будемо через PhpMyAdmin.(рис. 3.18).

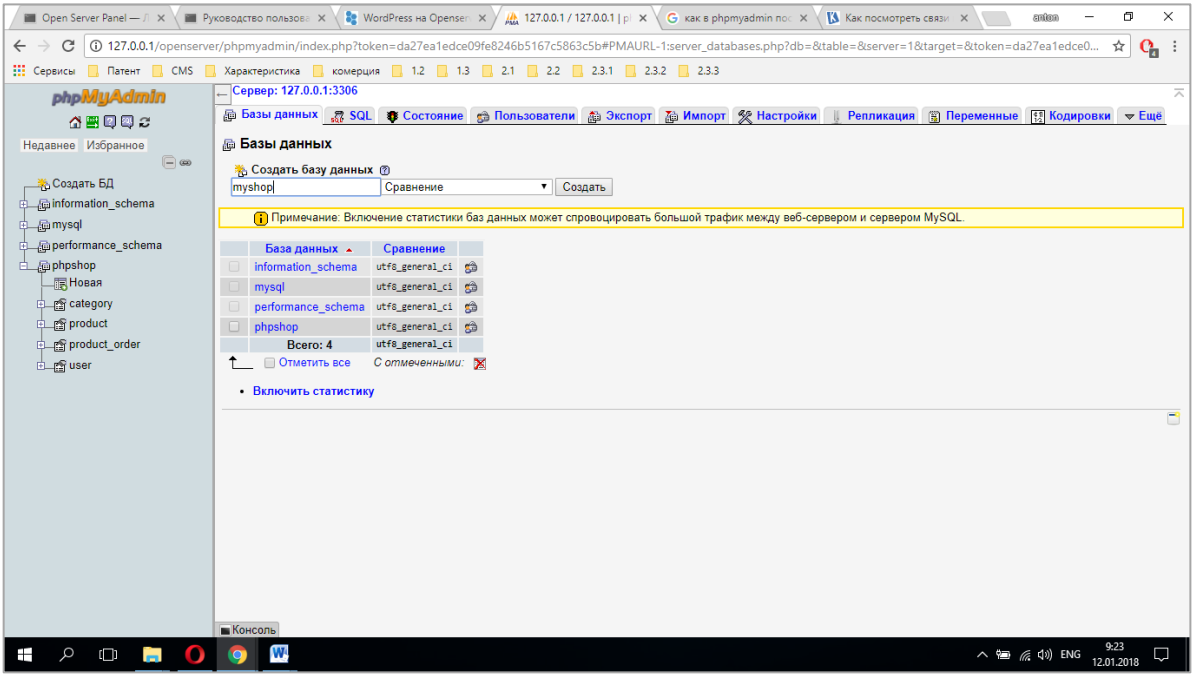


Рисунок 3.18. – Створення бази даних PhpMyAdmin

Для кращого розуміння таблиць БД опишемо їх детальніше. Її структура представлена у таблиці 3.5.

Таблиця 3.5. – Структура та опис таблиці користувачів додатку

Назва атрибуту	Тип даних	Короткий опис
_ID	Integer	Первинний ключ, унікальний ідентифікатор рядків
Name	String	Ім’я користувача
Sort order	Integer	Порядок сортування

Приведемо приклад запису категорії яка записується до бази даних.

Лістинг 3.1. – Запис категорій в БД

```
public static function createCategory($name, $sortOrder, $status)
{

        $db = Db::getConnection();
        $sql = 'INSERT INTO category (name, sort_order, status) '
                    'VALUES (:name, :sort_order, :status)';

$result = $db->prepare($sql);
$result->bindParam(':name', $name, PDO::PARAM_STR);
$result->bindParam(':sort_order', $sortOrder, PDO::PARAM_INT);
$result->bindParam(':status', $status, PDO::PARAM_INT);
return $result->execute();

}
```

Іншою важливою таблицею є таблиця продукції. Структура та короткий опис атрибутів таблиці детально наведено в таблиці 3.5.

Таблиця 3.6. – Структура та опис продукції

Назва атрибуту	Тип даних	Короткий опис
_ID	Integer	Первинний ключ, унікальний ідентифікатор рядків

name	varchar	Ім’я користувача

Продовження таблиці 3.6.

Category_id	Integer	Категорії
code	Integer	Код
price	Float	Ціна
availability	Integer	Наявність товару
brand	varchar	Бренд
description	text	Опис
Is_new	Integer	Новий товар
Is_recomended	Integer	Рекомендований
status	Integer	Статус товару

Дана таблиця використовується для збереження інформації про товари магазину. Вказується ціна, товар який присутній та готовий до продажу, перелік брендів, лаконічний опис товару, нова продукція та рекомендації.

Наступна таблиця вміщує дані про закази покупців. Структура та опис вказано у таблиці 3.7.

Таблиця 3.7. – Структура та опис про закази покупців

Назва атрибуту	Тип даних	Короткий опис
_ID	Integer	Первинний ключ, унікальний ідентифікатор рядків
user name	String	Ім’я користувача
User phone	Integer	Телефон користувача
User_commend	text	Коментар користувача
User_id	Integer	Id користувача
date	timestamp	Дата
products	text	Продукція

status	Integer	Статус

В таблиці вказані атрибути, які використовуються під час заказу товару користувачем. При цьому вказуються ім’я, телефон для зв’язку, коментар до заказу.

Таблиця User призначена для збереження інформації про зареєстрованих клієнтів Інтернет-магазину. Структура та опис вказані у таблиці 3.8.

Таблиця 3.8. – Структура та опис таблиці про зареєстрованих клієнтів

Назва атрибуту	Тип даних	Короткий опис
_ID	String	Первинний ключ, унікальний ідентифікатор рядків
name	String	Ім’я користувача
emeil	String	Пошта
password	String	Пароль
role	String	Роль

Атрибути, які вказані в таблиці 3.4. використовуються для опису користувача. При реєстрації нового клієнта магазину вводяться: ім’я, електронна пошта та пароль. Лістинг 3.1.

Розглянемо моделі та категорії БД.

Лістинг 3.1. – Отримання категорії

```
	public function getCategoryList()
{
		$db = Db::getConnection();
		 $categoryList = array();

		$result = $db->query('SELECT id, name FROM category ORDER BY sort_order ASC');

		$i = 0;
	while($row = $result->fetch()) {
		$categoryList[$i]['id'] = $row['id'];
		$categoryList[$i]['name'] = $row['name'];
		$i++;
}
```

```
 Return$categoryList;
}
```

За допомогою функції getConnection виконується з’єднання з базою даних. Далі надсилаємо запит к БД і потім вже використовуючи цикл while перебираємо кожні елементи які знаходяться в БД та отримуємо результат. Лістинг 3.2.

Лістинг 3.2. – Видалення категорії

```
public static function deleteCategoryById($id)
{

          $db=Db::getConnection();
               $sql = 'DELETE FROM category WHERE id = :id';
               $result = $db->prepare($sql);
               $result->bindParam(':id', $id, PDO::PARAM_INT);
               return $result->execute();
                                          }
```

Для того щоб видалити категорію id спочатку з’єднуємося, відправляємо текст запиту та використовуючи підготовлений запит виконується отримання та повернення запиту. Лістинг 3.3.

Лістинг 3.3 – Зміна характеристик на категоріях

```
public static function updateCategoryById($id, $name, $sortOrder, $status)
 {
        $db = Db::getConnection();
        $sql = "UPDATE category
               SET
                      name = :name,
                      sort_order = :sort_order,
                      status = :status
               WHERE id = :id";

$result = $db->prepare($sql);
$result->bindParam(':id', $id, PDO::PARAM_INT);
$result->bindParam(':name', $name, PDO::PARAM_STR);
$result->bindParam(':sort_order', $sortOrder, PDO::PARAM_INT);
$result->bindParam(':status', $status, PDO::PARAM_INT);
return $result->execute();
}
```

В даному коді мі бачимо те що можна змінити певні характеристики таких категорій як им’я, порядок сортирування та статус. Лістинг 3.4.

Лістинг 3.4. – Опис контролера

```
public function actionIndex()
{
	self::checkAdmin();
	$categoriesList = Category::getCategoriesListAdmin();
	require_once(ROOT. '/views/admin_category/index.php');
	return true;
}
```

Вказуємо метод для сторінки «Управління категоріями». Після йде перевірка доступу, отримуємо список категорій та підключаємо вид. Лістинг 3.5.

Лістинг 3.5. – Додавання категорії

```
public function actionCreate()
{
	self::checkAdmin();
	if (isset($_POST['submit'])) {
		$name = $_POST['name'];
		$sortOrder = $_POST['sort_order'];
		$status = $_POST['status'];
		$errors = false;
		if (!isset($name) || empty($name)) {
		$errors[] = 'Заполните поля';
		if ($errors == false) {
			Category::createCategory($name, $sortOrder, $status);
			header("Location: /admin/category");
	}
}
require_once(ROOT. '/views/admin_category/create.php');
return true;
}
}
```

Метод для сторінки «Додати категорію». Також спочатку йде перевірка доступу. Обробка форми, коли форма відправлена то отримуємо дані з форми. Якщо

помилок немає то додаємо нову категорію та перенаправляємо користувача на сторінку управліннями категоріями. Лістинг 3.6.

Лістинг 3.6. – Редактування категорії

```
public function actionUpdate($id)
{
	self::checkAdmin();
$category = Category::getCategoryById($id);
if (isset($_POST['submit'])) {
		$name = $_POST['name'];
		$sortOrder = $_POST['sort_order'];
		$status = $_POST['status'];
		Category::updateCategoryById($id, $name, $sortOrder, $status);
		header("Location: /admin/category");
	}
	require_once(ROOT. '/views/admin_category/update.php');
	return true;
}
```

Спочатку отримуємо дані о конкретній категорії. Форма обробляється та відправляється. Зберігаємо змінення. Після цьго за допомогою header перенаправляємо користувача на сторінку управліннями категоріями та йде підключення виду. Лістинг 3.7.

Лістинг 3.7. – Видалення категорії

```
public function actionDelete($id)
	self::checkAdmin();
	if (isset($_POST['submit'])) {
		Category::deleteCategoryById($id);
		header("Location: /admin/category");
	}
	require_once(ROOT. '/views/admin_category/delete.php');
	return true;
}
}
```

В даному коді по порядку йде перевірка доступу, обробка форми, переправлення користувача та підключення виду.

3.2. Особливості розробки клієнтської частини

Клієнтська частина полягає в написанні HTML розмітки сторінки та складання стилів CSS. В даному розділі будуть представлені лістинги з view php.

View – це не тільки HTML, але і взагалі уявлення в цілому, а так же логіка його формування. Шаблонизатор, фільтри, різні функції / об'єкти помогаютщіе нам сформувати view (наприклад форматування дат, серіалізатор і т.д.) У переважній більшості випадків "уявлення" наших даних – це HTTP запити і HTTP відповіді. HTML – це лише частина HTTP відповіді. Лістинг 3.8.

Лістинг 3.8 – Приклад файлу view php

```
<?php include ROOT.
'/views/layouts/header.php';
?>
                                    <section>
                                        <div class="container">
                                            <div class="row">
                                                <div class="col-sm-3">
                                                    <div class="left-sidebar">
                                                        <h2>Каталог</h2>
                                                        <div class="panel-group category-products">
                                                            <?php foreach ($categories as
$categoryItems): ?>
                                                                <div class="panel panel-default">
                                                                    <div class="panel-heading">
                                                                        <h4 class="panel-title"><a

href="/category/<?php echo $categoryItems['id']; ?>"
                                                                                                class="<?php if
($categoryId == $categoryItems['id']) echo 'active'; ?>"><?php echo
$categoryItems['name']; ?></a>
                                                                        </h4>
                                                                    </div>
                                                                </div>
                                                            <?php endforeach; ?>
                                                        </div>
                                                        <div class="shipping text-center">
```

Даний код виводить категорію для відображення користувачам. Код знаходиться в розділі view/catalog.(рис. 3.19).

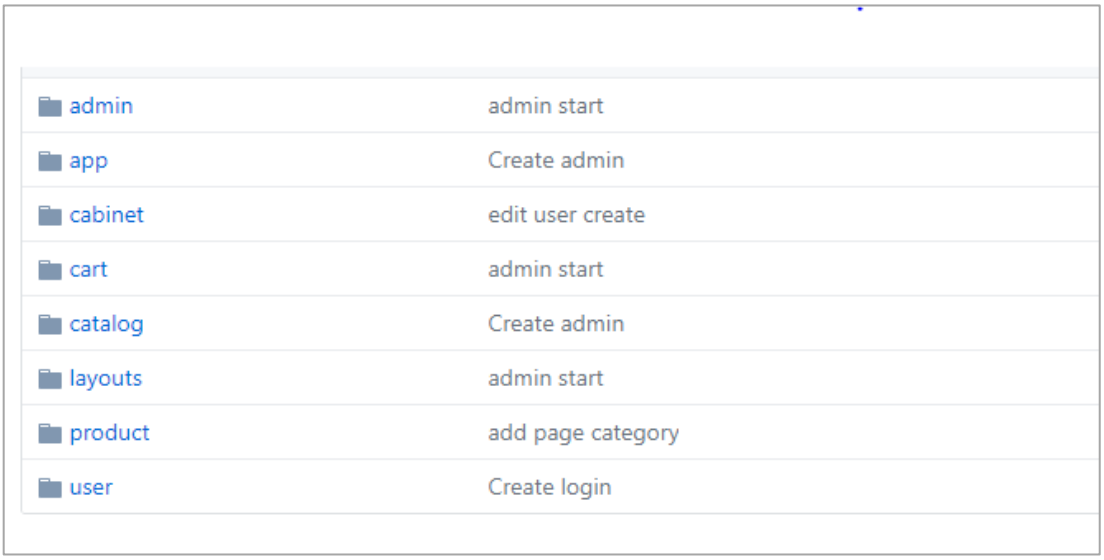


Рисунок 3.19. – Каталоги файлів для інтерфейсу користувача

Дизайн сайту визначається створеними стилями. CSS (каскадні таблиці стилів) – формальна мова опису зовнішнього вигляду документа, написаного з використанням мови розмітки. Фрагмент розробленого файлу css наведено у лістингу 3.9.

Лістинг 3.9. – Приклад коду main css

```
body
{
          font-family: 'Roboto', sans-serif;
          background:;
          position: relative;
          font-weight:400px;
        }
        ul li {
          list-style: none;
        }
        a:hover {
        outline: none;
        text-decoration:none;
        }
        a:focus {
```

```
        outline:none;
        outline-offset: 0;
      }
```

Також ми використовуємо систему Bootstrap, він підключається до виду сторінки. Bootstrap – CSS / HTML фреймворк для створення сайтів. Лістинг 3.10.

Лістинг 3.10. – Приклад підключення Bootstrap

```
<div class="modal" tabindex="-1" role="dialog">
  <div class="modal-dialog" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title">Modal title</h5>
        <button type="button" class="close" data-dismiss="modal" aria-label="Close">
          <span aria-hidden="true">&times;</span>
        </button>
      </div>
      <div class="modal-body">
        <p>Modal body text goes here.</p>
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-primary">Save changes</button>
        <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
      </div>
    </div>
  </div>
</div>
```

Для асинхронності запитів використовуєм JavaScript, а саме підключимо бібліотеку JQuery. Лістинг 3.11.

Лістинг 3.11 – Приклад асинхронного запиту JQuery

```
$(document).ready(function
() {
                                $(function () {
                                    $.scrollUp({
                                        scrollName: 'scrollUp', // Element ID
                                        scrollDistance: 300, // Distance from top/bottom before
```

```
showing element (px)
            scrollFrom: 'top', // 'top' or 'bottom'
            scrollSpeed: 300, // Speed back to top (ms)
            easingType: 'linear', // Scroll to top easing (see
http://easings.net/)
            animation: 'fade', // Fade, slide, none
            animationSpeed: 200, // Animation in speed (ms)
            scrollTrigger: false, // Set a custom triggering element. Can
be an HTML string or jQuery object
            //scrollTarget: false, // Set a custom target element for
scrolling to the top
            scrollText: '<i class="fa fa-angle-up"></i>', // Text for
element, can contain HTML
            scrollTitle: false, // Set a custom <a> title if required.
            scrollImg: false, // Set true to use image
            activeOverlay: false, // Set CSS color to display scrollUp
active point, e.g '#00FFFF'
            zIndex: 2147483647 // Z-Index for the overlay
        });
    });
```

За допомогою AJAX ($.ajax () в jQuery або XMLHttpRequest в чистому JS) на сервер відправляється запит. При цьому сторінка не перезавантажується. З точки зору сервера запит абсолютно аналогічний запитам, що відправляється при переходах по посиланнях або при введенні адреси в адресний рядок.

Також Bootstrap вимагає використання функції JavaScript. Зокрема, вимагають jQuery, Popper.js та наших власних плагінів JavaScript. Лістинг 3.12.

Лістинг 3.12 – Bootstrap jQuery

```
jQuery.fn = jQuery.prototype = {

    // The current version of jQuery being used
    jquery: version,

    constructor: jQuery,

    // The default length of a jQuery object is 0
    length: 0,
```

```
toArray: function () {
    return slice.call(this);
},

// Get the Nth element in the matched element set OR
// Get the whole matched element set as a clean array
get: function (num) {

    // Return all the elements in a clean array
    if (num == null) {
        return slice.call(this);
    }
```

3.3 Інтерфейс користувача та інструкція по використанню

Під інтерфейсом користувача розуміють сукупність елементів комп'ютерної програми, які забезпечує взаємодію користувача с даним програмним засобом. До елементів інтерфейсу користувача веб-сайту Інтернет-магазина відносять: головне меню системи, вікна входу на сайт зареєстрованих користувачів, введення інформації про товари, розділи керування змістом сайту та ін.

Контрольний приклад з використання розробленої CMS реалізовано на прикладі Інтернет-магазину електроніки.

Головна сторінка Інтернет-магазину вміщує логотип магазину, головне меню, список останніх надходжень у магазин, посилання на корзину та вхід у власний кабінет покупця, контакти (рис.3.20).

За допомогою головної сторінки покупець може переходити на інші сторінки сайту, використовуючи гіперпосилання та елементи меню.

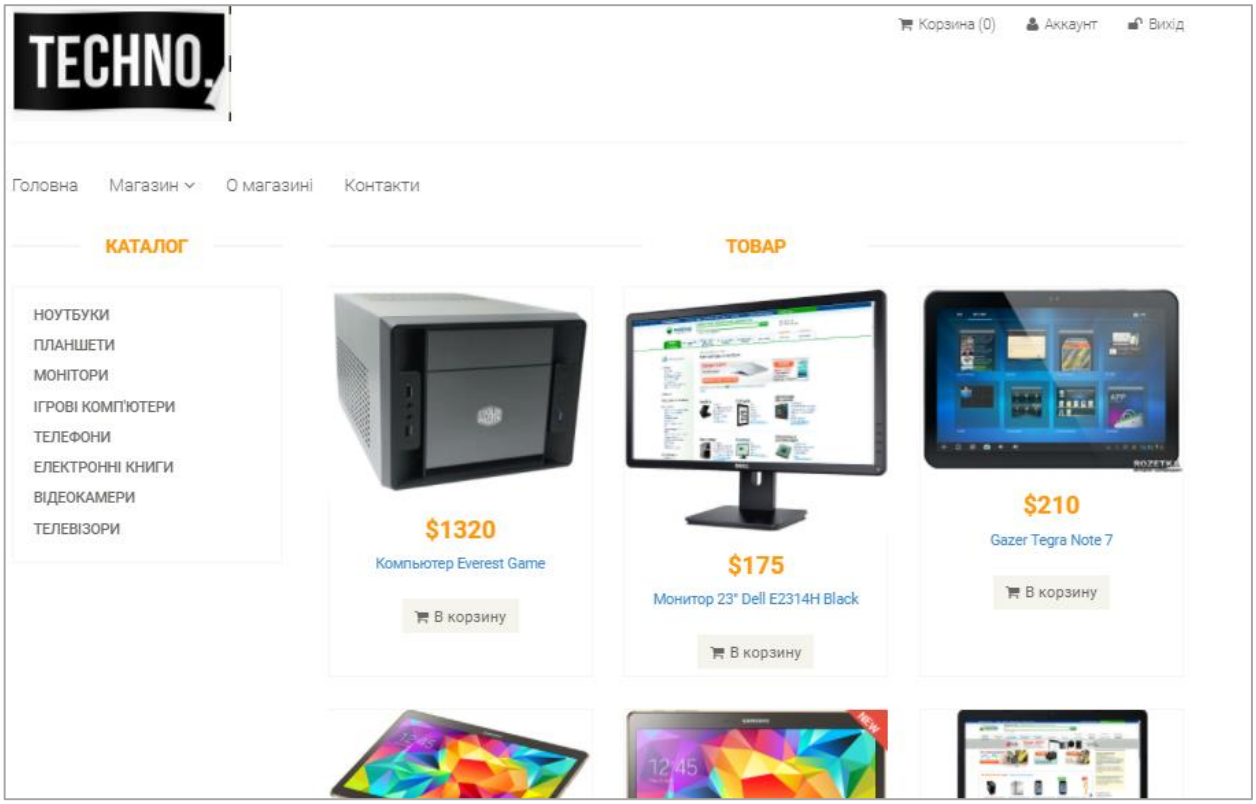


Рисунок 3.20 – Вид головної сторінки магазину

Розташування елементів керування на головной сторінці:

– каталог товарів розташований з лівої частини екрану;

– реєстрація користувачів розташована в правому верхньому куті;

– кошик покупця розташований зліва від реєстрації користувачів;

– у верхньому лівому куті знаходиться відображається контактний телефон;

– праворуч від телефону знаходиться посилання електронної пошти;

– логотип магазину знаходиться зліва зверху;

– під логотипом знаходиться головне меню системи з пунктами "Главная", "Магазин", "О Магазине", "Контакти".

Сторінка категорії товарів вміщує інформацію про наявні товари з вибраної категорії (рис.3.21.). Для додавання товару в корзину покупок потрібно під картинкою товару натиснути кнопку «В корзину».

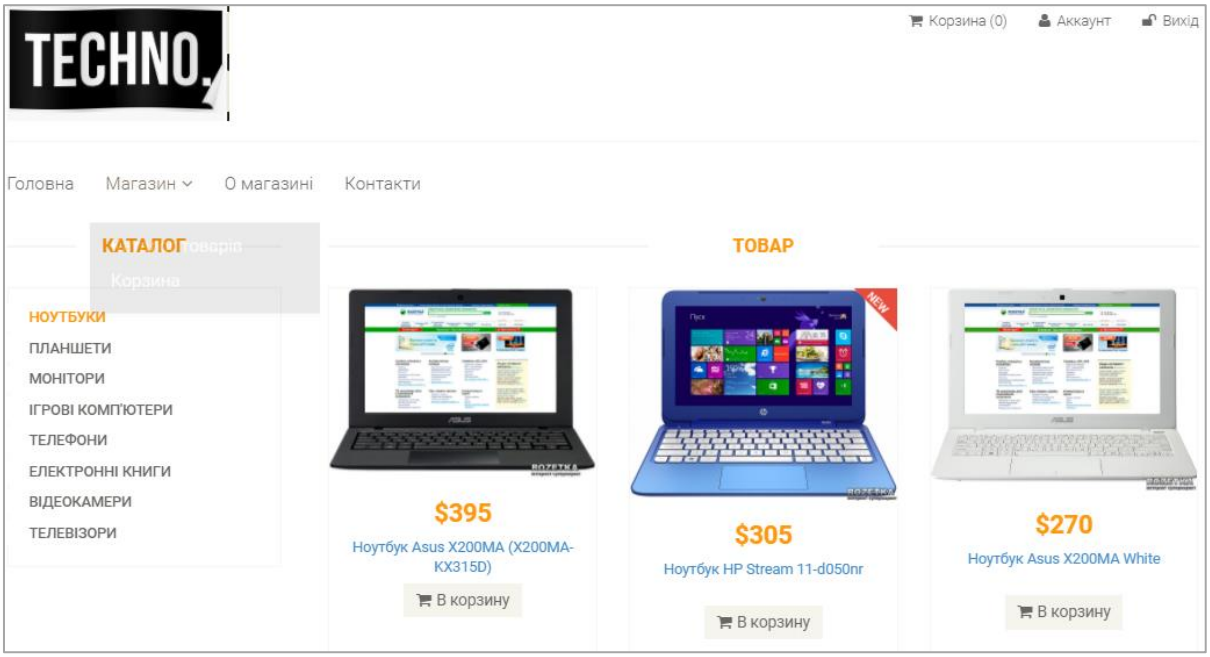


Рисунок 3.21 – Сторінка категорії товарів

Обравши товар і натиснувши на його назву здійснюється перехід до розгорнутого опису товару(рис. 3.22)

Рисунок 3.22. – Сторінка опису товару

Реєстрація клієнтів магазину здійснюється на відповідної сторінці (рис. 3.23). На даної сторінці покупець вводить е-мейл та пароль.

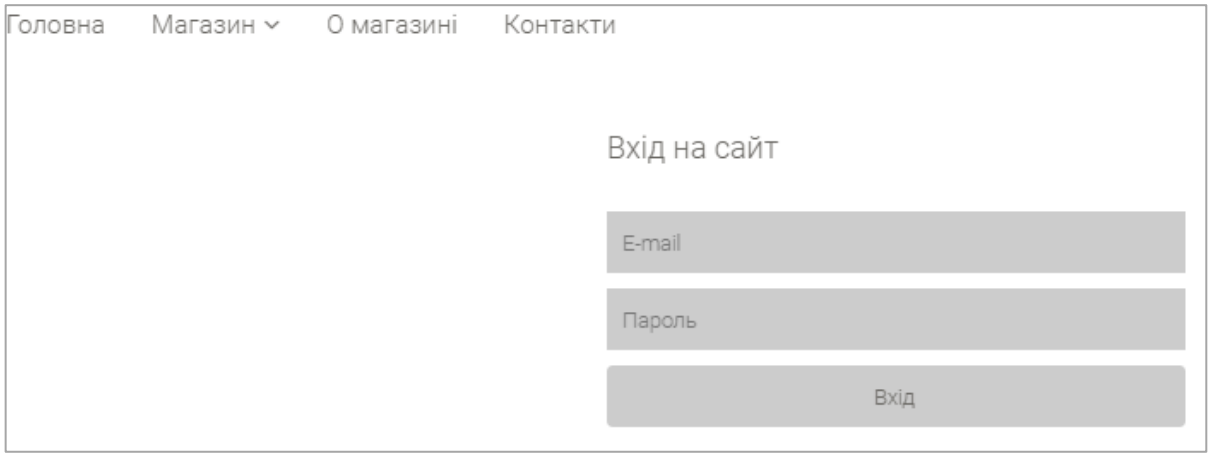


Рисунок 3.23 – Сторінка реєстрації нового клієнта магазину

При вводі даних з неприпустимими знаками, символами під час реєстрації з’явиться повідомлення про помилку (рис. 3.24).

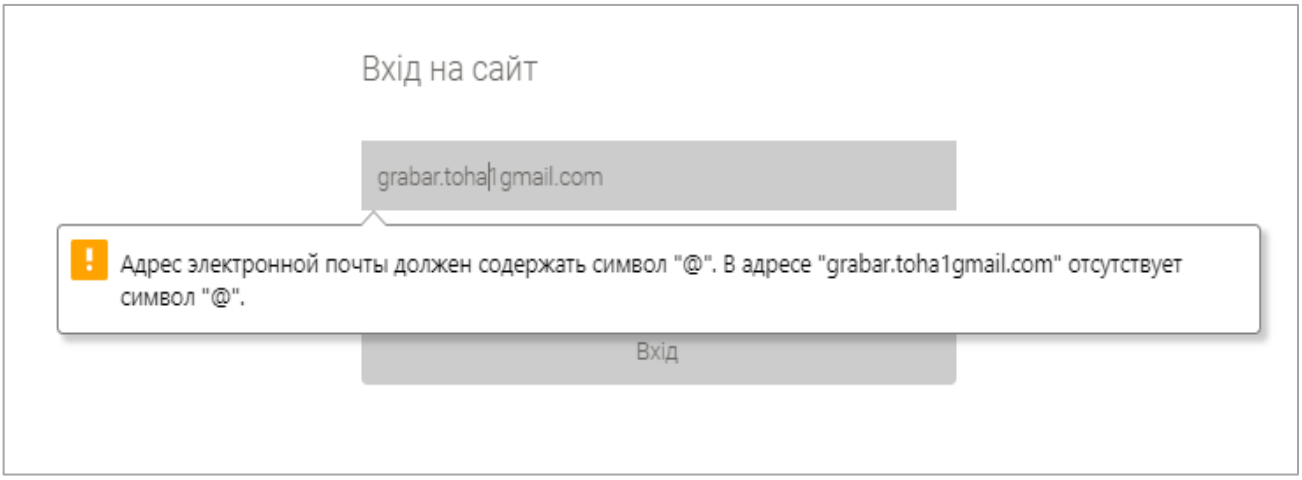


Рисунок 3.24 – Сторінка реєстрації

Для редагування списку товарів (видалення) використовують кнопку "Видалити". (рис. 3.25).

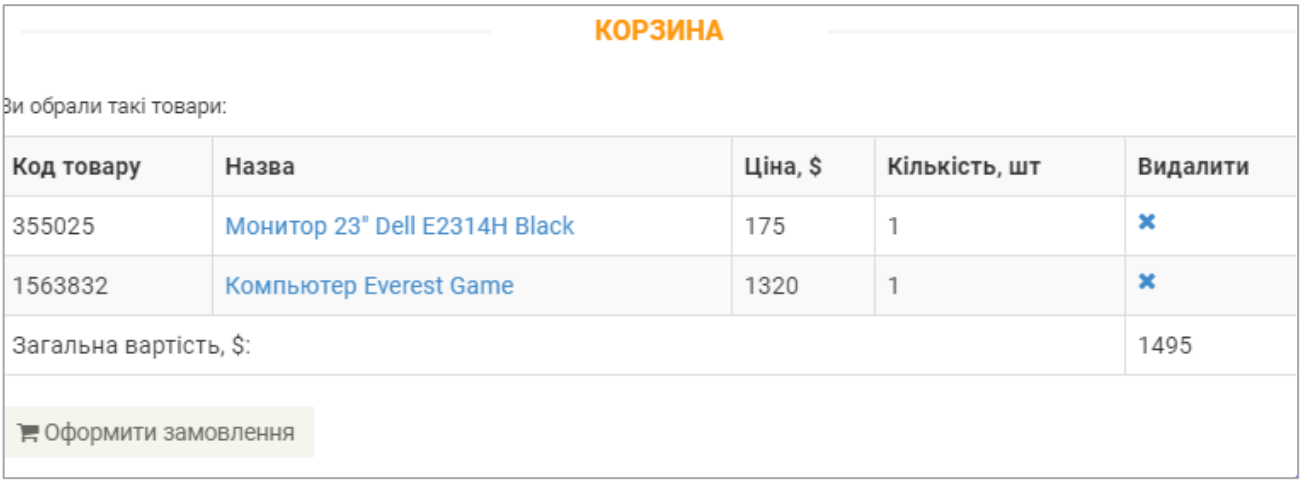


Рисунок 3.25 – Сторінка "Корзина"

На сторінці оформлення заказу заповнюється форма в якій треба вказати им’я, номер телефону та коментар до заказу (3.27) Також видно кількість обраних товарі які включені до заказу та загальну ціну. Після заповнення форми треба натиснути кнопку «Оформить» (рис. 3.26).

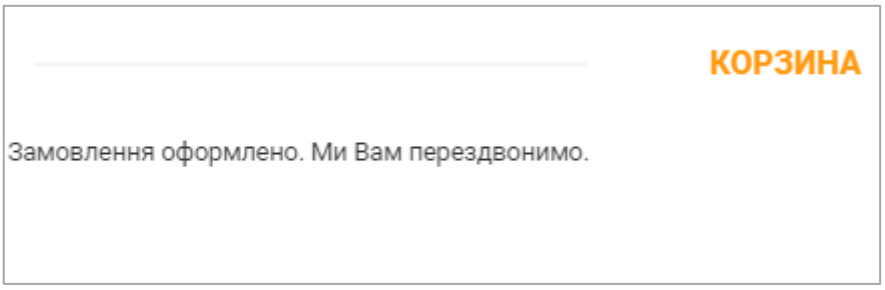


Рисунок 3.26. – Остання фаза оформлення заказу

Керування контентом магазину здійснюється в панелі адміністратора. Там відбувається модифікація параметрів магазину, додавання нових категорій товарів, додавання списку товарів, завантаження їх зображень, відстеження клієнтів, управління оплатами та інші функції. Саме настройки в адміністративній панелі впливають на те, як буде проходити процес взаємодії покупця з магазином, шляхом зміни зовнішнього вигляду, структури і наповнення вітрини.

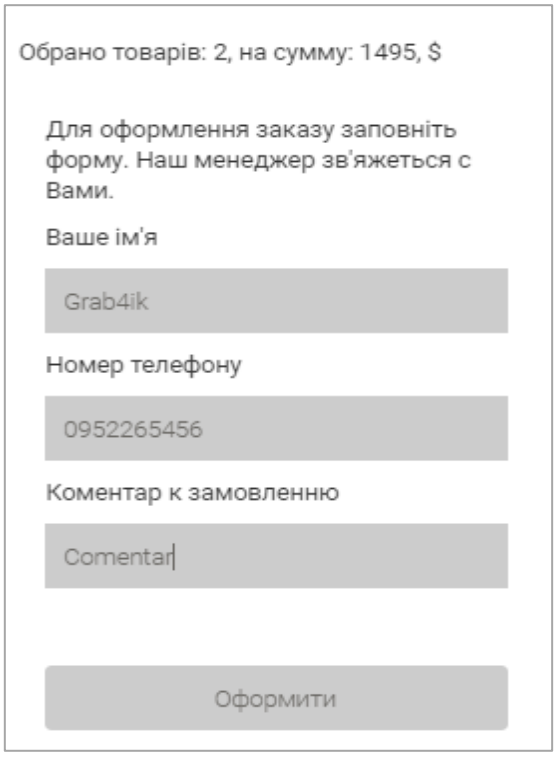


Рисунок 3.27. – Сторінка оформлення заказу

Для відкриття панелі адміністратора в рядок адреси вводиться назва сайту та символи "/admin", після чого відкривається сторінка адміністратора (рис. 3.28). Панель дозволяє працювати з такими розділами: "Управління товарами", "Управління категоріями", "Управління заказами".

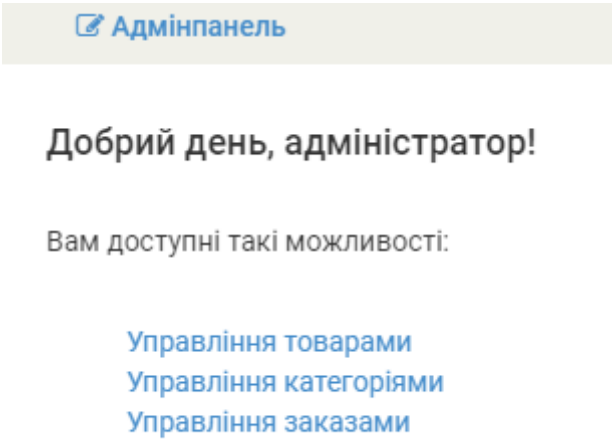


Рисунок 3.28 – Панель адміністратора

Розділ "Управління товарами" дозволяє додавати товари у список та видаляти (рис. 3.29).

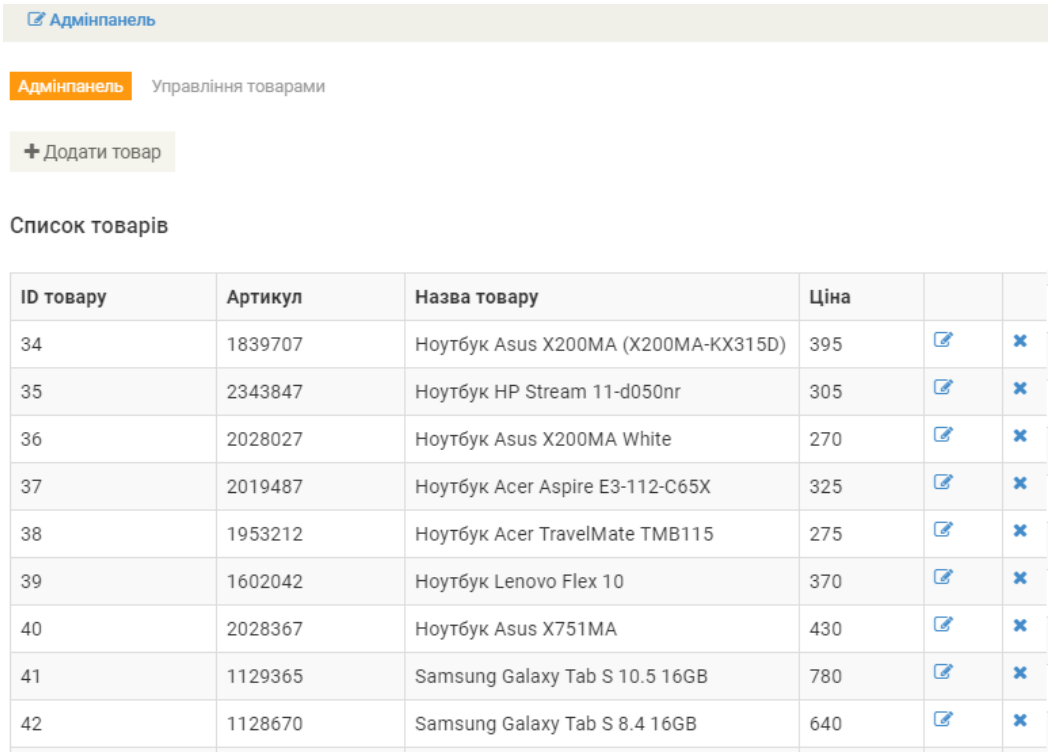


Рисунок 3.29 – Сторінка адміністратора для управління товарами

Для редагування або видалення товару зі списку потрібно натиснути відповідну кнопку у рядку з назвою товару (рис. 3.29).

Для додавання нового товару обирають кнопку "Добавить товар". У вікні введення нового товару адміністратор магазину має можливість вказувати назву товару, його артикул, вартість, категорію, виробника товару, додавати зображення та детально описувати переваги товару (рис. 3.30). Також можна вказувати статус товару, наявність на скаді, являється він новинкою та розширена інформація про товар (характеристики, переваги, поради з використання, посилання на youtube відео).

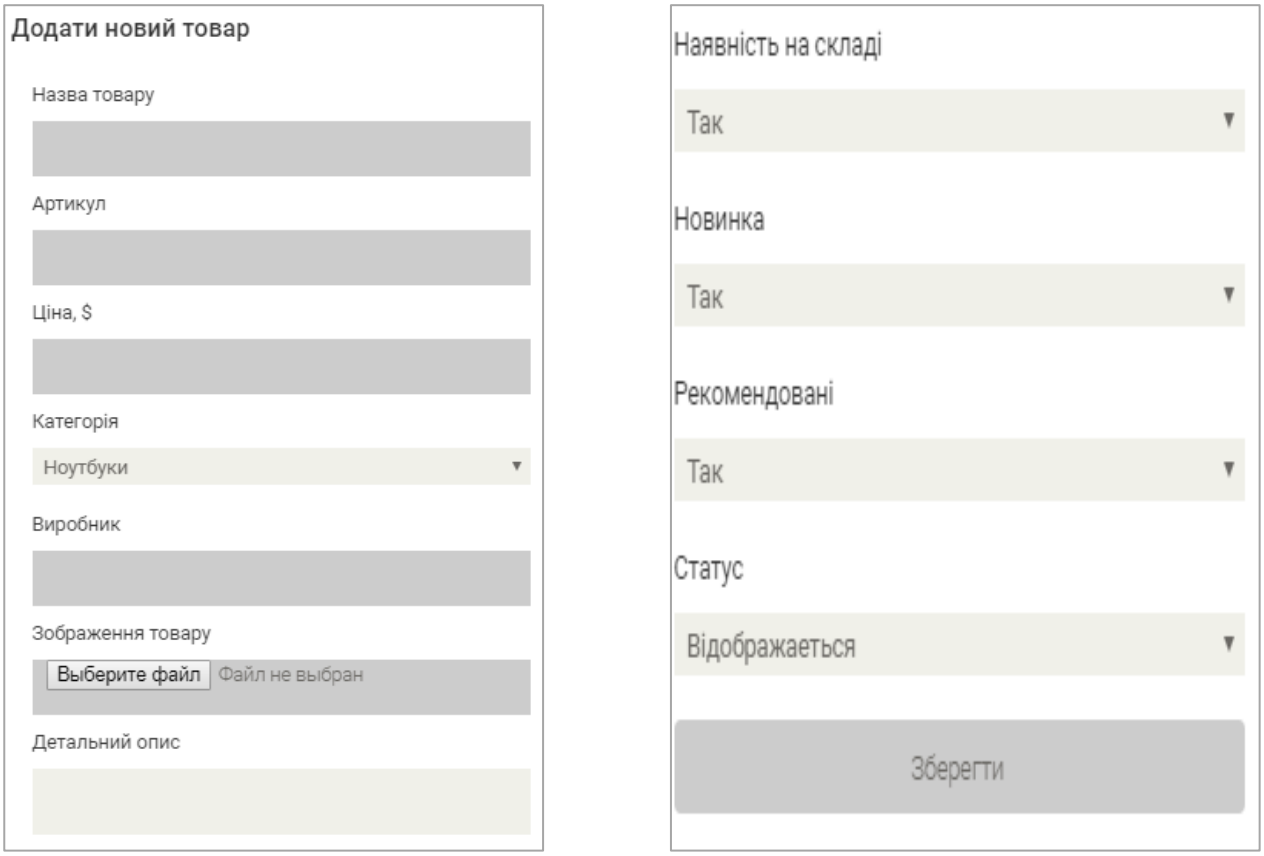


Рисунок 3.30 – Сторінка додавання товару

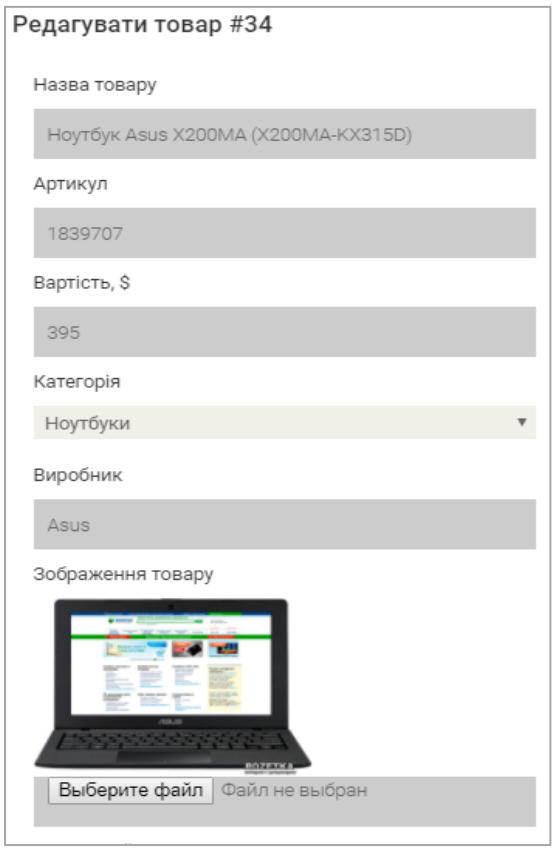


Рисунок 3.31. – Сторінка редагування даних товару

Редагування параметрів товарів магзину здійснюється адміністратором системи натисканням відповідної кнопки у списку товарів, після чого відкривається вікно редагування даних (рис.3.31).

При видаленні товару зі списку товарів з'являється вікно підтвердження операції (рис. 3.32). При позитивної відповіді треба натиснути кнопку "Удалить".

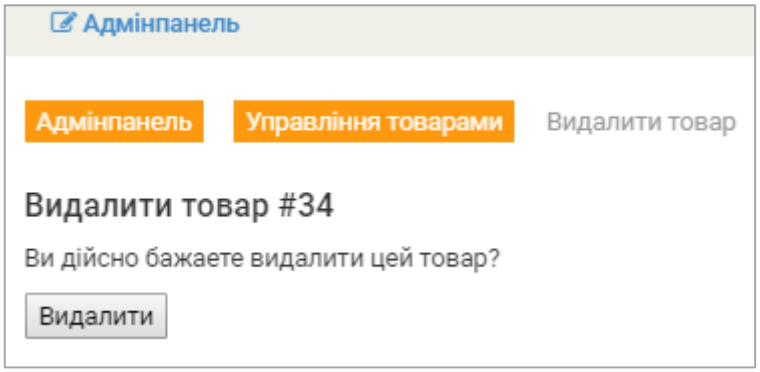
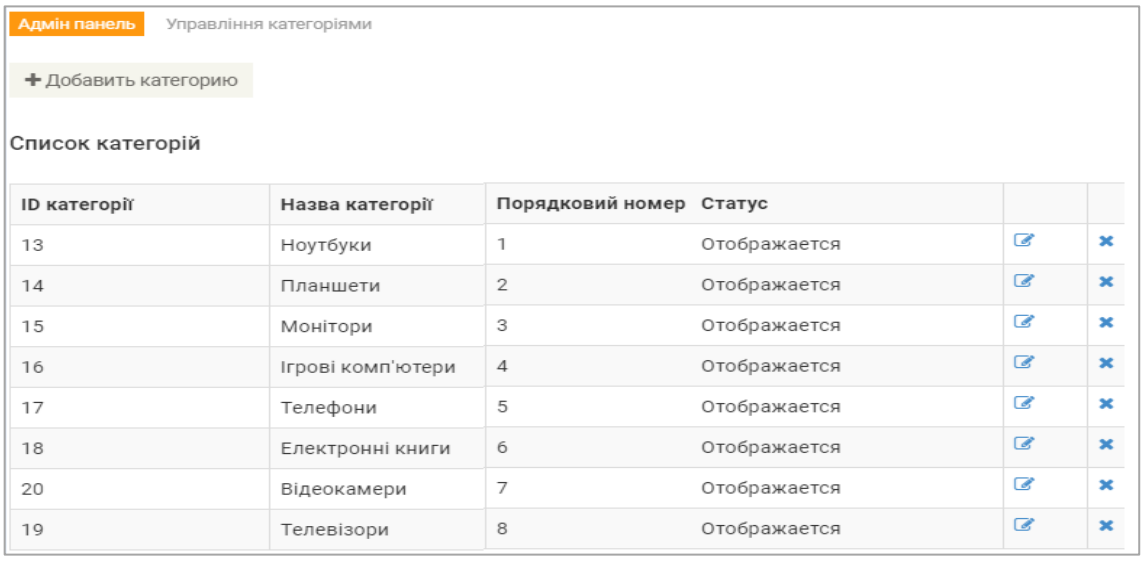


Рисунок 3.32. – Сторінка видалення товару

Управління категоріями товарів здійснюється вибором пункта "Управление категориями" вкладки "Админпанель" (рис. 3.33). Для додавання нової категорії призначено кнопка "Добавить категорию".



ID категорії	Назва категорії	Порядковий номер	Статус		
13	Ноутбуки	1	Отображается		
14	Планшети	2	Отображается		
15	Монітори	3	Отображается		
16	Ігрові комп'ютери	4	Отображается		
17	Телефони	5	Отображается		
18	Електронні книги	6	Отображается		
20	Відеокамери	7	Отображается		
19	Телевізори	8	Отображается		

Рисунок 3.33 – Сторінка управління категоріями

При додаванні нової категорії вказується її назва та порядок відображення у списку категорій, який відповідає порядку пунктів у списку "Каталог" сторінок сайту.

Для редагування товару в розділі управління товарами треба натиснути на значок . На сторінці редагування категоріями вказується им’я, порядковий номер та статус товару.(рис. 3.34.).

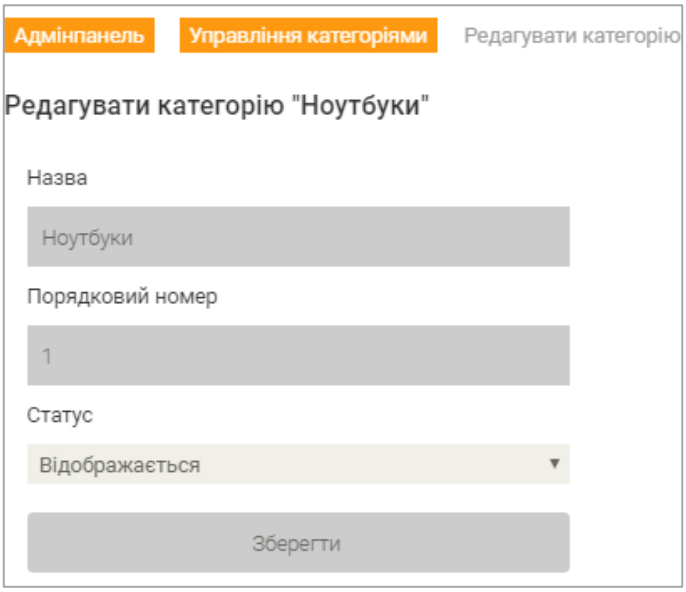


Рисунок 3.34. – Сторінка редагування категорії

Управління заказами здійснюється за допомогою відповідної панелі при виборі команди на вкладці "Админпанель" (рис. 3.34). Панель управління заказами вміщує інформацію про номер заказу, прізвище покупця, телефон, дата заказу, статус заказу (рис. 3.35). Сторінка дозволяє переглядати, редагувати та видаляти закази.

Список замовлень

ID замовлення	Ім'я покупця	Телефон покупця	Дата оформлення	Статус			
51	Grab4ik	0952265456	2018-01-15 11:06:36	Новый заказ			
50	Николай	0996584847	2018-01-15 02:26:09	Новый заказ			
49	Евгений	0955568558	2018-01-15 02:25:41	Новый заказ			
48	Елена	0665874558	2018-01-15 02:25:11	Новый заказ			
47	Anton	0954678454	2018-01-15 01:43:32	Новый заказ			
46	Екатерина	0995412687	2018-01-05 18:34:42	В обработке			
45	Виктор	0664587255	2018-01-10 12:54:45	Закрыт			

Рисунок 3.35 – Сторінка управління заказами

Щоб переглянути товар натискаємо кнопку  та переходимо на сторінку «Перегляд заказу».(рис. 3.36). На сторінці вказана інформація заказу, ім’я, телефон, номер клієнта, коментар, номер, статус та дата заказу. На нижньому рівні знаходиться заказаний клієнтом товар. Після перегляду заказу, натиснувши на кнопку «Назад» повертаємось до сторінки управління заказами.

Адмінпанель | Управління замовленнями | Огляд замовлення

Огляд замовлення #51

Інформація замовлення

Номер замовлення	51
Ім'я клієнта	Grab4ik
Телефон клієнта	0952265456
Коментар клієнта	Comentar
Статус замовлення	Новый заказ
Дата замовлення	2018-01-15 11:06:36

Товари в замовленні

ID товару	Артикул товару	Назва	Ціна	Кількість
44	355025	Монитор 23" Dell E2314H Black	$175	1
45	1563832	Компьютер Everest Game	$1320	1

Назад

Рисунок 3.36. – Сторінка перегляду заказу

3.3 Оцінка очікуваних ефектів від впровадження CMS

Продуктивність та ефективність від провадження автоматизованої системи визначають порівнянням результатів її роботи і затрат всіх видів ресурсів, необхідних для її створення і розвитку.Таблиця 3.9.

Таблиця 3.9 – Основні позначення розрахунку ефективності системи

№	Позначення	Пояснення
1.	Впроект	Витрати на проектування комплексу задач
2.	Впрогр	Витрати на програмування комплексу задач
3.	Вн	Витрати на програмування в період налагодження ПЗ комплексу задач
4.	Ввпров	Витрати на впровадження
5.	Вб	Приведені до одного року витрати на обробку інформації при базовому варіанті організації обробки
6.	Вп	Приведені до одного року витрати на обробку інформації при впроваджуваному (пропонованому) варіанті організації системи обробки
7.	E_y	Річний економічний ефект
8.	S	Річна економія, яку отримає підприємство в результаті впровадження АСУ
9.	C	Капітальні вкладення, які було витрачено в результаті впровадження АСУ
10.	r_n	Нормативний коефіцієнт окупності капітальних вкладень, узятий для конкретної галузі

Формули для розрахунку та результати наведені у таблиці 3.10

Таблиця 3.10 – Розрахунок ефективності системи

№	Формула	Опис формули	Розрахунок, грн..
1	C = Впроект + Впрограм + Вн + Ввпров	Сума капітальних витрат	6340
2	S=Вб–Вп	Сума річної економії	9950 – 4280 = 6260
3	E =S–C×r	Річний ефект від впровадження	6260 – 2252 = 4508
4	P = C/S = E	Термін окупності	6340 / 6260=1,01

ВИСНОВКИ

Дана дипломна робота присвячена розробці CMS Інтернет-магазину. В дослідженні наведено загальну характеристику системи управління контентом, проаналізовано основні переваги та проблеми CMS, сформовано основні вимоги щодо системи управління контентом Інтернет-магазину, обрано архітектуру та технологію розробки. З функціональної точки зору наведено список функцій використовуваних програмних засобів. Розроблено керівництво користувача по взаємодії з інтерфейсом програмного засобу.

В результаті проведеного дослідження розроблена система управління контентом Інтернет-магазину. Розроблена система реалізує наступні функції:

– модифікація розділів сайту;
– додавання, видалення, редагування товарів сайту;
– ведення обліку заказів сайту;
– управління вбудованими блоками сайту;
– відправка повідомлень адміністратору;
– розмежування прав доступу для користувачів;
– вбудований файловий менеджер.
– Переваги системи:
– модульність;
– гнучке налаштування;
– модель звіту про помилки, що дозволяє відстежувати будь-які помилки, навіть після введення в експлуатацію системи.

Система розроблена на мові програмування PHP. За базу даних використовувався MySQL, як бібліотеку – jQuery.

У техніко-економічному обґрунтуванні система була оцінена за різними критеріями. Проведено розрахунок ефективності від провадження системи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бурников М.Ю. CMS обзор [Електронний ресурс] / М.Ю. Бурников, Н.С. Акимов. – Режим доступу : http://cmsobzor.ru/art/cms_all.
2. Вакалюк Т.А. Переваги застосування існуючих моделей розробки програмного забезпечення у реальному проекті / Т. А. Вакалюк, О. В. Куліковська // Теорія і практика професійної підготовки фахівців у контексті загальноєвропейських інтеграційних процесів. – Житомир: Вид-во ЖДУ, 2016. – С. 56-59.
3. Вандюк Д. CMS Drupal. Руководство по разработке системы управления сайтом / Д. Вандюк. – М.: Вильямс, 2009. – 576 с.
4. Горнаков С.Г. Осваиваем популярные системы управления сайтом / С.Г. Горнаков. – М.: ДМК Пресс, 2009. – 336 с.
5. Движки для сайтов, платные и бесплатные CMS системы, каталог систем управления сайтами [Електронний ресурс]. – Режим доступу: http://www.cmsmagazine.ru/catalogue.
6. Десять критериев выбора идеальной системы управления контентом. [Електронний ресурс]: Боуг П. О. // Статьи от AKmedia – 2009. – Режим доступу: http://akmedia.ru/page/27.
7. Дорин В. Электронные системы адаптивного компьютерного обучения, на основе стандартов образо- вательных сред [Електронний ресурс] / В. Дорин, С.Р. Лейкец. – Режим доступа : http://www.- solus.ru/articl_314.htm.
8. Електронна комерція: за і проти [Електронний ресурс] / Л. Л. Філіппова // Вісник Національного технічного університету "ХПІ". Технічний прогрес та ефективність виробництва. – 2013. – № 44. – С. 58-65. – Режим доступу: http://nbuv.gov.ua/UJRN/Vcpitp_2013_44_10.
9. Електронний бізнес, електронна комерція, Інтернет-торгівля: сутність та взаємозв’язок понять [Електронний ресурс] / Д. Д. Євтушенко // Бізнес Інформ. – 2014. – № 8. – С. 184 – 188. – Режим доступу: http://nbuv.gov.ua/UJRN/binf_2014_8_33.

10. Замерина В.В. Система управления содержимым [Электронный ресурс] / В.В. Замерина, Д.Б. Головин. – Режим доступа : http://cmslist.ru/doc/cms_1.htm.
11. Інтелектуальні засоби організації сайтів шкіл на базі CMS Drupal / О. М. Марковець, А. М. Пелещишин, Ю. В. Форкун, Д. В. Делечук // Вісник Національного університету "Львівська політехніка". – 2010. – № 689 : Інформаційні системи та мережі. – С. 201–209. – Бібліографія: 4 назви.
12. Классификация систем управления содержимым web-ресурсов и их использование для разработки сайта дистанционного обучения / М.А. Курилов, С.В. Терещенко // Штучний інтелект. — 2010. — № 3. — С. 648-654. — Бібліогр.: 6 назв. — рос.
13. Колисниченко Д.Н. Выбираем лучший бесплатный движок для вашего сайта. CMS, Joomla! и Drupal / Д.Н.Колисниченко – СПб.:БЧВ-Петербург, 2010. – С. 288.
14. Куртов Л.М. Общая классификация CMS [Электронный ресурс] / Л.М. Куртов. – Режим доступа : http://www.solus.ru/articl_267.htm.
15. Мищишин В. І. Аналіз проблеми уніфікації архітектури систем управління контентом / В. І. Мищишин, П. І. Жежнич // Вісник Національного університету "Львівська політехніка". – 2010. – № 689 : Інформаційні системи та мережі. – С. 218–226. – Бібліографія: 10 назв.
16. Мэрриотт Дж., Уоринг Э., Joomla! 3.0: Официальное руководство. - СПб.: Питер, 2013. – 496 с.
17. Обґрунтування вибору систем керування базою даних та управління контентом при створенні web-сайту / В. О. Сацик, О. М. Решетило, О.В. Сацик // Комп’ютерно-інтегровані технології: освіта, наука, виробництво. - 2013. - № 13. - С. 57-65. - Режим доступу: http://nbuv.gov.ua/UJRN/Kitonv_2013_13_12
18. Огляд CMS. Сайт про системи управління сайтом. http://www.cmslist.ru/cms-list-55.html.
19. Особливості проектування та аналіз узагальненої архітектури систем електронної контент-комерції / В. А. Висоцька, Л. Б. Чирун, Л. В. Чирун // Вісник Національного університету "Львівська політехніка". Інформаційні

системи та мережі. – 2013. – № 770. – С. 83 – 100. – Режим доступу: http://nbuv.gov.ua/UJRN/VNULPICM_2013_770_13.

20. Пелещишин А.М. Веб 2.0 та Семантичний Веб: порівняльний аналіз перспективних тенденцій розвитку WWW / А.М. Пелещишин, О.Л. Березко // Східно-Європейський журнал передових технологій. – Харків, 2008. – 6/2 (24). – С.43–51.
21. Пелещишин А.М. Позиціонування сайтів у глобальному інформаційному середовищі: Монографія / А.М. Пелещішин. – Львів: Вид-во Нац. ун-ту "Львівська політехніка", 2008. – 260 с.
22. Побудова інтерактивного електронного навчального посібника в системі управління контентом E107 / О. В. Конюшенко, Д. О. Івченко // Управляющие системы и машины. – 2014. – № 5. – С. 68 – 76. – Режим доступу: http://nbuv.gov.ua/UJRN/USM_2014_5_10.
23. Полубояров В.В. Понятие и функции системы управления контентом [Електронний ресурс]:– Режим доступу: http://www.intuit.ru/department/internet/inwwwtech/7/#sect5.
24. Порівняння СУБД [Електронний ресурс] // SQL, PostgreSQL, MySQL - Режим доступу: http://devacademy.ru/posts/sqlite-vs-mysql-vs-postgresql.
25. Прохоров Н. Системы управления контентом / Н. Прохоров –Компьютер Пресс, 2009. – С. 134-137.
26. Рассел С., Норвиг П. Искусственный интеллект: современный подход; пер. с англ. М.: Издат. дом «Вильямс», 2008. 2-е изд. 1408 с.
27. Руденко Г. Р. Аналіз програмного забезпечення логістичного моделювання / Г. Р. Руденко // Бізнес Інформ. – 2012. – № 8 (415). – С. 247–251.
28. Системи керування контентом як засіб електронної web-публикації медичної інформації: підхід на основі opencms / В. П. Марценюк // Медична інформатика та інженерія. –2008.–№4.- С. 9-25. - Режим доступу: http://nbuv.gov.ua/UJRN/Mii_2008_4_4.

29. Стратегія і основні кроки при розробці web-сайту. [Електронний ресурс]: Режим доступу: http://ruszura.in.ua/neobhidno-znaty/stratehiya-i-osnovni-kroky-pryrozrobtsiweb-sajta.html
30. Технології Java SE [Електронний ресурс]. – Режим доступу: https://www.ibm.com/developerworks/ru/java/newto/.
31. Шаньгин В.С. Защита компьютерной информации. Эффективные методы и средства / В. С. Шаньгин. – М.: ДМК Пресс, 2010. – 544 с.
32. Шелестов Н.Д. Перспективы CMS [Електронний ресурс] / Н.Д. Шелестов. – Режим доступа : http://whatis.techtarget.com/definition/gci508916,00.html
33. Щеглова Е.А. Классификация CMS [Електронний ресурс] / Е.А. Щеглова, П.К. Павлов. – Режим доступа: http://stroimweb.moy.su/publ/6
34. Buchanan B. G., Shortliffe E. H. Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project. – MA: Addison-Wesley, 1984. – 769 p.
35. Byron A Using Drupal / Byron A., Berry A., Haug N., Eaton J., Walker J, Robbins J. – 2008. – 496 p.
36. Dorian Gorgan, Victor Bacu, Teodor Stefanut, Denisa Rodila, Danut Mihon Computer Science Department, Technical University of Cluj-Napoca. Grid based Satellite Image Processing Platform for Earth Observation Application Development // IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications 21-23 September 2009, Rende (Cosenza), Italy. pp. 247-252 11.
37. Giovanni Caire. Programming Developing multi-agent applications with JADE. Tutorial for beginners / Caire Giovanni (TILAB, formerly CSELT). 2009. pp. 23.
38. OMG Unified Modeling Language Superstructure Specification, version 2.1.1. Document formal/2007-02-05, Object Management Group, February 2008.
39. PHP and MySQL Web Development (4th Edition), Luke Welling, Laura Thomson 848 p.
40. UML диаграммы в Rational Rose [Електронний ресурс] // Rational Rose - Режим доступу: http://www.caseclub.ru/articles/rose2.html (дата звернення 01.05.2016).

41. Unified Modeling Language: Superstructure (англ.) (вид. 2.1.1). Object Management Group. 2008.

ДОДАТКИ

Додаток А

Файл index.php

```
<?php

// FRONT CONTROLLER

// Общие настройки
ini_set('display_errors',1);
error_reporting(E_ALL);

session_start();

// Подключение файлов системы
define('ROOT', dirname(__FILE__));
require_once(ROOT.'/components/Autoload.php');


// Вызов Router
$router = new Router();
$router->run();
```

Autoload.php

```
<?php

/**
 * Функция __autoload для автоматического подключения классов
 */
function __autoload($class_name)
{
    // Массив папок, в которых могут находиться необходимые классы
    $array_paths = array(
        '/models/',
        '/components/',
        '/controllers/',
    );

    // Проходим по массиву папок
    foreach ($array_paths as $path) {

        // Формируем имя и путь к файлу с классом
        $path = ROOT. $path. $class_name. '.php';

        // Если такой файл существует, подключаем его
        if (is_file($path)) {
            include_once $path;
        }
    }
}
```

Додаток Б
Cart.php

```
<?php

/**
 * Класс Cart
 * Компонент для работы корзиной
 */
class Cart
{

    /**
     * Добавление товара в корзину (сессию)
     * @param int $id <p>id товара</p>
     * @return integer <p>Количество товаров в корзине</p>
     */
    public static function addProduct($id)
    {
        // Приводим $id к типу integer
        $id = intval($id);

        // Пустой массив для товаров в корзине
        $productsInCart = array();

        // Если в корзине уже есть товары (они хранятся в сессии)
        if (isset($_SESSION['products'])) {
            // То заполним наш массив товарами
            $productsInCart = $_SESSION['products'];
        }

        // Проверяем есть ли уже такой товар в корзине
        if (array_key_exists($id, $productsInCart)) {
            // Если такой товар есть в корзине, но был добавлен еще раз, увеличим
количество на 1
            $productsInCart[$id] ++;
        } else {
            // Если нет, добавляем id нового товара в корзину с количеством 1
            $productsInCart[$id] = 1;
        }

        // Записываем массив с товарами в сессию
        $_SESSION['products'] = $productsInCart;

        // Возвращаем количество товаров в корзине
        return self::countItems();
    }

    /**
     * Подсчет количество товаров в корзине (в сессии)
     * @return int <p>Количество товаров в корзине</p>
     */
    public static function countItems()
    {
        // Проверка наличия товаров в корзине
        if (isset($_SESSION['products'])) {
            // Если массив с товарами есть
            // Подсчитаем и вернем их количество
            $count = 0;
            foreach ($_SESSION['products'] as $id => $quantity) {
                $count = $count + $quantity;
            }
```

```
            return $count;
        } else {
            // Если товаров нет, вернем 0
            return 0;
        }
    }

    /**
     * Возвращает массив с идентификаторами и количеством товаров в корзине<br/>
     * Если товаров нет, возвращает false;
     * @return mixed: boolean or array
     */
    public static function getProducts()
    {
        if (isset($_SESSION['products'])) {
            return $_SESSION['products'];
        }
        return false;
    }

    /**
     * Получаем общую стоимость переданных товаров
     * @param array $products <p>Массив с информацией о товарах</p>
     * @return integer <p>Общая стоимость</p>
     */
    public static function getTotalPrice($products)
    {
        // Получаем массив с идентификаторами и количеством товаров в корзине
        $productsInCart = self::getProducts();

        // Подсчитываем общую стоимость
        $total = 0;
        if ($productsInCart) {
            // Если в корзине не пусто
            // Проходим по переданному в метод массиву товаров
            foreach ($products as $item) {
                // Находим общую стоимость: цена товара * количество товара
                $total += $item['price'] * $productsInCart[$item['id']];
            }
        }

        return $total;
    }

    /**
     * Очищает корзину
     */
    public static function clear()
    {
        if (isset($_SESSION['products'])) {
            unset($_SESSION['products']);
        }
    }

    /**
     * Удаляет товар с указанным id из корзины
     * @param integer $id <p>id товара</p>
     */
    public static function deleteProduct($id)
    {
        // Получаем массив с идентификаторами и количеством товаров в корзине
        $productsInCart = self::getProducts();
```

```
        // Удаляем из массива элемент с указанным id
        unset($productsInCart[$id]);

        // Записываем массив товаров с удаленным элементом в сессию
        $_SESSION['products'] = $productsInCart;
    }

}
```

AdminBase.php

```
<?php

/**
 * Абстрактный класс AdminBase содержит общую логику для контроллеров, которые
 * используются в панели администратора
 */
abstract class AdminBase
{

    /**
     * Метод, который проверяет пользователя на то, является ли он администратором
     * @return boolean
     */
    public static function checkAdmin()
    {
        // Проверяем авторизирован ли пользователь. Если нет, он будет переадресован
        $userId = User::checkLogged();

        // Получаем информацию о текущем пользователе
        $user = User::getUserById($userId);

        // Если роль текущего пользователя "admin", пускаем его в админпанель
        if ($user['role'] == 'admin') {
            return true;
        }

        // Иначе завершаем работу с сообщением об закрытом доступе
        die('Access denied');
    }

}
```

Додаток В
Pagination.php

```
<?php

/*
 * Класс Pagination для генерации постраничной навигации
 */

class Pagination
{

    /**
     *
     * @var Ссылок навигации на страницу
     *
     */
    private $max = 10;

    /**
     *
     * @var Ключ для GET, в который пишется номер страницы
     *
     */
    private $index = 'page';

    /**
     *
     * @var Текущая страница
     *
     */
    private $current_page;

    /**
     *
     * @var Общее количество записей
     *
     */
    private $total;

    /**
     *
     * @var Записей на страницу
     *
     */
    private $limit;

    /**
     * Запуск необходимых данных для навигации
     * @param type $total <p>Общее количество записей</p>
     * @param type $currentPage <p>Номер текущей страницы</p>
     * @param type $limit <p>Количество записей на страницу</p>
     * @param type $index <p>Ключ для url</p>
     */
    public function __construct($total, $currentPage, $limit, $index)
    {
        # Устанавливаем общее количество записей
        $this->total = $total;

        # Устанавливаем количество записей на страницу
        $this->limit = $limit;
```

```
        # Устанавливаем ключ в url
        $this->index = $index;

        # Устанавливаем количество страниц
        $this->amount = $this->amount();

        # Устанавливаем номер текущей страницы
        $this->setCurrentPage($currentPage);
    }

    /**
     *  Для вывода ссылок
     * @return HTML-код со ссылками навигации
     */
    public function get()
    {
        # Для записи ссылок
        $links = null;

        # Получаем ограничения для цикла
        $limits = $this->limits();

        $html = '<ul class="pagination">';
        # Генерируем ссылки
        for ($page = $limits[0]; $page <= $limits[1]; $page++) {
            # Если текущая это текущая страница, ссылки нет и добавляется класс
active
            if ($page == $this->current_page) {
                $links.= '<li class="active"><a href="#">'. $page. '</a></li>';
            } else {
                # Иначе генерируем ссылку
                $links.= $this->generateHtml($page);
            }
        }

        # Если ссылки создались
        if (!is_null($links)) {
            # Если текущая страница не первая
            if ($this->current_page > 1)
            # Создаём ссылку "На первую"
                $links = $this->generateHtml(1, '&lt;'). $links;

            # Если текущая страница не первая
            if ($this->current_page < $this->amount)
            # Создаём ссылку "На последнюю"
                $links.= $this->generateHtml($this->amount, '&gt;');
        }

        $html.= $links. '</ul>';

        # Возвращаем html
        return $html;
    }

    /**
     * Для генерации HTML-кода ссылки
     * @param integer $page - номер страницы
     *
     * @return
     */
    private function generateHtml($page, $text = null)
    {
        # Если текст ссылки не указан
```

```
        if (!$text)
        # Указываем, что текст - цифра страницы
            $text = $page;

        $currentURI = rtrim($_SERVER['REQUEST_URI'], '/'). '/';
        $currentURI = preg_replace('~/page-[0-9]+~', '', $currentURI);
        # Формируем HTML код ссылки и возвращаем
        return
                '<li><a href="'. $currentURI. $this->index. $page. '">'. $text.
'</a></li>';
    }

    /**
     *  Для получения, откуда стартовать
     *
     * @return массив с началом и концом отсчёта
     */
    private function limits()
    {
        # Вычисляем ссылки слева (чтобы активная ссылка была посередине)
        $left = $this->current_page - round($this->max / 2);

        # Вычисляем начало отсчёта
        $start = $left > 0 ? $left : 1;

        # Если впереди есть как минимум $this->max страниц
        if ($start + $this->max <= $this->amount) {
        # Назначаем конец цикла вперёд на $this->max страниц или просто на минимум
            $end = $start > 1 ? $start + $this->max : $this->max;
        } else {
            # Конец - общее количество страниц
            $end = $this->amount;

            # Начало - минус $this->max от конца
            $start = $this->amount - $this->max > 0 ? $this->amount - $this->max : 1;
        }

        # Возвращаем
        return
                array($start, $end);
    }

    /**
     * Для установки текущей страницы
     *
     * @return
     */
    private function setCurrentPage($currentPage)
    {
        # Получаем номер страницы
        $this->current_page = $currentPage;

        # Если текущая страница больше нуля
        if ($this->current_page > 0) {
            # Если текущая страница меньше общего количества страниц
            if ($this->current_page > $this->amount)
            # Устанавливаем страницу на последнюю
                $this->current_page = $this->amount;
        } else
        # Устанавливаем страницу на первую
            $this->current_page = 1;
    }
```

```
    /**
     * Для получения общего числа страниц
     *
     * @return число страниц
     */
    private function amount()
    {
        # Делим и возвращаем
        return ceil($this->total / $this->limit);
    }

}
```

Db.php

```
<?php

/**
 * Класс Db
 * Компонент для работы с базой данных
 */
class Db
{

    /**
     * Устанавливает соединение с базой данных
     * @return \PDO <p>Объект класса PDO для работы с БД</p>
     */
    public static function getConnection()
    {
        // Получаем параметры подключения из файла
        $paramsPath = ROOT. '/config/db_params.php';
        $params = include($paramsPath);

        // Устанавливаем соединение
        $dsn = "mysql:host={$params['host']};dbname={$params['dbname']}";
        $db = new PDO($dsn, $params['user'], $params['password']);

        // Задаем кодировку
        $db->exec("set names utf8");

        return $db;
    }

}
```

Додаток Г
Router.php

```
<?php

/**
 * Класс Router
 * Компонент для работы с маршрутами
 */
class Router
{

    /**
     * Свойство для хранения массива роутов
     * @var array
     */
    private $routes;

    /**
     * Конструктор
     */
    public function __construct()
    {
        // Путь к файлу с роутами
        $routesPath = ROOT. '/config/routes.php';

        // Получаем роуты из файла
        $this->routes = include($routesPath);
    }

    /**
     * Возвращает строку запроса
     */
    private function getURI()
    {
        if (!empty($_SERVER['REQUEST_URI'])) {
            return trim($_SERVER['REQUEST_URI'], '/');
        }
    }

    /**
     * Метод для обработки запроса
     */
    public function run()
    {
        // Получаем строку запроса
        $uri = $this->getURI();

        // Проверяем наличие такого запроса в массиве маршрутов (routes.php)
        foreach ($this->routes as $uriPattern => $path) {

            // Сравниваем $uriPattern и $uri
            if (preg_match("~$uriPattern~", $uri)) {

                // Получаем внутренний путь из внешнего согласно правилу.
                $internalRoute = preg_replace("~$uriPattern~", $path, $uri);

                // Определить контроллер, action, параметры

                $segments = explode('/', $internalRoute);

                $controllerName = array_shift($segments). 'Controller';
```

```
                $controllerName = ucfirst($controllerName);

                $actionName = 'action'. ucfirst(array_shift($segments));

                $parameters = $segments;

                // Подключить файл класса-контроллера
                $controllerFile = ROOT. '/controllers/'.
                        $controllerName. '.php';

                if (file_exists($controllerFile)) {
                    include_once($controllerFile);
                }

                // Создать объект, вызвать метод (т.е. action)
                $controllerObject = new $controllerName;

                /* Вызываем необходимый метод ($actionName) у определенного
                 * класса ($controllerObject) с заданными ($parameters) параметрами
                 */
                $result = call_user_func_array(array($controllerObject, $actionName),
$parameters);

                // Если метод контроллера успешно вызван, завершаем работу роутера
                if ($result != null) {
                    break;
                }
            }
        }
    }

}
```

Db_params.php

```
<?php

// Массив с параметрами подключения к базе данных
return array(
    'host' => 'localhost',
    'dbname' => 'phpshop',
    'user' => 'root',
    'password' => '',
);
```

Додаток Д
Routes.php

```
<?php

return array(
    // Товар:
    'product/([0-9]+)' => 'product/view/$1', // actionView в ProductController
    // Каталог:
    'catalog' => 'catalog/index', // actionIndex в CatalogController
    // Категория товаров:
    'category/([0-9]+)/page-([0-9]+)' => 'catalog/category/$1/$2', // actionCategory
в CatalogController
    'category/([0-9]+)' => 'catalog/category/$1', // actionCategory в
CatalogController
    // Корзина:
    'cart/checkout' => 'cart/checkout', // actionAdd в CartController
    'cart/delete/([0-9]+)' => 'cart/delete/$1', // actionDelete в CartController
    'cart/add/([0-9]+)' => 'cart/add/$1', // actionAdd в CartController
    'cart/addAjax/([0-9]+)' => 'cart/addAjax/$1', // actionAddAjax в CartController
    'cart' => 'cart/index', // actionIndex в CartController
    // Пользователь:
    'user/register' => 'user/register',
    'user/login' => 'user/login',
    'user/logout' => 'user/logout',
    'cabinet/edit' => 'cabinet/edit',
    'cabinet' => 'cabinet/index',
    // Управление товарами:
    'admin/product/create' => 'adminProduct/create',
    'admin/product/update/([0-9]+)' => 'adminProduct/update/$1',
    'admin/product/delete/([0-9]+)' => 'adminProduct/delete/$1',
    'admin/product' => 'adminProduct/index',
    // Управление категориями:
    'admin/category/create' => 'adminCategory/create',
    'admin/category/update/([0-9]+)' => 'adminCategory/update/$1',
    'admin/category/delete/([0-9]+)' => 'adminCategory/delete/$1',
    'admin/category' => 'adminCategory/index',
    // Управление заказами:
    'admin/order/update/([0-9]+)' => 'adminOrder/update/$1',
    'admin/order/delete/([0-9]+)' => 'adminOrder/delete/$1',
    'admin/order/view/([0-9]+)' => 'adminOrder/view/$1',
    'admin/order' => 'adminOrder/index',
    // Адмінпанель:
    'admin' => 'admin/index',
    // О магазине
    'contacts' => 'site/contact',
    'about' => 'site/about',
    // Главная страница
    'index.php' => 'site/index', // actionIndex в SiteController
    '' => 'site/index', // actionIndex в SiteController
);
```

Додаток К
AdminCategoryController.php

```
<?php.

/**
 * Контроллер AdminCategoryController
 * Управление категориями товаров в админпанели
 */
class AdminCategoryController extends AdminBase
{

    /**
     * Action для страницы "Управление категориями"
     */
    public function actionIndex()
    {
        // Проверка доступа
        self::checkAdmin();

        // Получаем список категорий
        $categoriesList = Category::getCategoriesListAdmin();

        // Подключаем вид
        require_once(ROOT. '/views/admin_category/index.php');
        return true;
    }

    /**
     * Action для страницы "Добавить категорию"
     */
    public function actionCreate()
    {
        // Проверка доступа
        self::checkAdmin();

        // Обработка формы
        if (isset($_POST['submit'])) {
            // Если форма отправлена
            // Получаем данные из формы
            $name = $_POST['name'];
            $sortOrder = $_POST['sort_order'];
            $status = $_POST['status'];

            // Флаг ошибок в форме
            $errors = false;

            // При необходимости можно валидировать значения нужным образом
            if (!isset($name) || empty($name)) {
                $errors[] = 'Заполните поля';
            }


            if ($errors == false) {
                // Если ошибок нет
                // Добавляем новую категорию
                Category::createCategory($name, $sortOrder, $status);

                // Перенаправляем пользователя на страницу управлениями категориями
                header("Location: /admin/category");
            }
        }
```

```
        require_once(ROOT. '/views/admin_category/create.php');
        return true;
    }

    /**
     * Action для страницы "Редактировать категорию"
     */
    public function actionUpdate($id)
    {
        // Проверка доступа
        self::checkAdmin();

        // Получаем данные о конкретной категории
        $category = Category::getCategoryById($id);

        // Обработка формы
        if (isset($_POST['submit'])) {
            // Если форма отправлена
            // Получаем данные из формы
            $name = $_POST['name'];
            $sortOrder = $_POST['sort_order'];
            $status = $_POST['status'];

            // Сохраняем изменения
            Category::updateCategoryById($id, $name, $sortOrder, $status);

            // Перенаправляем пользователя на страницу управлениями категориями
            header("Location: /admin/category");
        }

        // Подключаем вид
        require_once(ROOT. '/views/admin_category/update.php');
        return true;
    }

    /**
     * Action для страницы "Удалить категорию"
     */
    public function actionDelete($id)
    {
        // Проверка доступа
        self::checkAdmin();

        // Обработка формы
        if (isset($_POST['submit'])) {
            // Если форма отправлена
            // Удаляем категорию
            Category::deleteCategoryById($id);

            // Перенаправляем пользователя на страницу управлениями товарами
            header("Location: /admin/category");
        }

        // Подключаем вид
        require_once(ROOT. '/views/admin_category/delete.php');
        return true;
    }

}
```

Додаток М
AdminOrderController.php

```
<?php.

/**
 * Контроллер AdminOrderController
 * Управление заказами в админпанели
 */
class AdminOrderController extends AdminBase
{

    /**
     * Action для страницы "Управление заказами"
     */
    public function actionIndex()
    {
        // Проверка доступа
        self::checkAdmin();

        // Получаем список заказов
        $ordersList = Order::getOrdersList();

        // Подключаем вид
        require_once(ROOT. '/views/admin_order/index.php');
        return true;
    }

    /**
     * Action для страницы "Редактирование заказа"
     */
    public function actionUpdate($id)
    {
        // Проверка доступа
        self::checkAdmin();

        // Получаем данные о конкретном заказе
        $order = Order::getOrderById($id);

        // Обработка формы
        if (isset($_POST['submit'])) {
            // Если форма отправлена
            // Получаем данные из формы
            $userName = $_POST['userName'];
            $userPhone = $_POST['userPhone'];
            $userComment = $_POST['userComment'];
            $date = $_POST['date'];
            $status = $_POST['status'];

            // Сохраняем изменения
            Order::updateOrderById($id, $userName, $userPhone, $userComment, $date,
$status);

            // Перенаправляем пользователя на страницу управлениями заказами
            header("Location: /admin/order/view/$id");
        }

        // Подключаем вид
        require_once(ROOT. '/views/admin_order/update.php');
        return true;
    }
```

```
    /**
     * Action для страницы "Просмотр заказа"
     */
    public function actionView($id)
    {
        // Проверка доступа
        self::checkAdmin();

        // Получаем данные о конкретном заказе
        $order = Order::getOrderById($id);

        // Получаем массив с идентификаторами и количеством товаров
        $productsQuantity = json_decode($order['products'], true);

        // Получаем массив с индентификаторами товаров
        $productsIds = array_keys($productsQuantity);

        // Получаем список товаров в заказе
        $products = Product::getProdustsByIds($productsIds);

        // Подключаем вид
        require_once(ROOT. '/views/admin_order/view.php');
        return true;
    }

    /**
     * Action для страницы "Удалить заказ"
     */
    public function actionDelete($id)
    {
        // Проверка доступа
        self::checkAdmin();

        // Обработка формы
        if (isset($_POST['submit'])) {
            // Если форма отправлена
            // Удаляем заказ
            Order::deleteOrderById($id);

            // Перенаправляем пользователя на страницу управлениями товарами
            header("Location: /admin/order");
        }

        // Подключаем вид
        require_once(ROOT. '/views/admin_order/delete.php');
        return true;
    }

}
```

Додаток Н
AdminProductController.php

```
<?php.

/**
 * Контроллер AdminProductController
 * Управление товарами в админпанели
 */
class AdminProductController extends AdminBase
{

    /**
     * Action для страницы "Управление товарами"
     */
    public function actionIndex()
    {
        // Проверка доступа
        self::checkAdmin();

        // Получаем список товаров
        $productsList = Product::getProductsList();

        // Подключаем вид
        require_once(ROOT. '/views/admin_product/index.php');
        return true;
    }

    /**
     * Action для страницы "Добавить товар"
     */
    public function actionCreate()
    {
        // Проверка доступа
        self::checkAdmin();

        // Получаем список категорий для выпадающего списка
        $categoriesList = Category::getCategoriesListAdmin();

        // Обработка формы
        if (isset($_POST['submit'])) {
            // Если форма отправлена
            // Получаем данные из формы
            $options['name'] = $_POST['name'];
            $options['code'] = $_POST['code'];
            $options['price'] = $_POST['price'];
            $options['category_id'] = $_POST['category_id'];
            $options['brand'] = $_POST['brand'];
            $options['availability'] = $_POST['availability'];
            $options['description'] = $_POST['description'];
            $options['is_new'] = $_POST['is_new'];
            $options['is_recommended'] = $_POST['is_recommended'];
            $options['status'] = $_POST['status'];

            // Флаг ошибок в форме
            $errors = false;

            // При необходимости можно валидировать значения нужным образом
            if (!isset($options['name']) || empty($options['name'])) {
                $errors[] = 'Заполните поля';
            }
```

```
            if ($errors == false) {
                // Если ошибок нет
                // Добавляем новый товар
                $id = Product::createProduct($options);

                // Если запись добавлена
                if ($id) {
                    // Проверим, загружалось ли через форму изображение
                    if (is_uploaded_file($_FILES["image"]["tmp_name"])) {
                        // Если загружалось, переместим его в нужную папке, дадим
новое имя
                        move_uploaded_file($_FILES["image"]["tmp_name"],
$_SERVER['DOCUMENT_ROOT']. "/upload/images/products/{$id}.jpg");
                    }
                };

                // Перенаправляем пользователя на страницу управлениями товарами
                header("Location: /admin/product");
            }
        }

        // Подключаем вид
        require_once(ROOT. '/views/admin_product/create.php');
        return true;
    }

    /**
     * Action для страницы "Редактировать товар"
     */
    public function actionUpdate($id)
    {
        // Проверка доступа
        self::checkAdmin();

        // Получаем список категорий для выпадающего списка
        $categoriesList = Category::getCategoriesListAdmin();

        // Получаем данные о конкретном заказе
        $product = Product::getProductById($id);

        // Обработка формы
        if (isset($_POST['submit'])) {
            // Если форма отправлена
            // Получаем данные из формы редактирования. При необходимости можно
валидировать значения
            $options['name'] = $_POST['name'];
            $options['code'] = $_POST['code'];
            $options['price'] = $_POST['price'];
            $options['category_id'] = $_POST['category_id'];
            $options['brand'] = $_POST['brand'];
            $options['availability'] = $_POST['availability'];
            $options['description'] = $_POST['description'];
            $options['is_new'] = $_POST['is_new'];
            $options['is_recommended'] = $_POST['is_recommended'];
            $options['status'] = $_POST['status'];

            // Сохраняем изменения
            if (Product::updateProductById($id, $options)) {


                // Если запись сохранена
                // Проверим, загружалось ли через форму изображение
                if (is_uploaded_file($_FILES["image"]["tmp_name"])) {
```

```
                    // Если загружалось, переместим его в нужную папке, дадим новое
имя
                    move_uploaded_file($_FILES["image"]["tmp_name"],
$_SERVER['DOCUMENT_ROOT']. "/upload/images/products/{$id}.jpg");
                }
            }

            // Перенаправляем пользователя на страницу управлениями товарами
            header("Location: /admin/product");
        }

        // Подключаем вид
        require_once(ROOT. '/views/admin_product/update.php');
        return true;
    }

    /**
     * Action для страницы "Удалить товар"
     */
    public function actionDelete($id)
    {
        // Проверка доступа
        self::checkAdmin();

        // Обработка формы
        if (isset($_POST['submit'])) {
            // Если форма отправлена
            // Удаляем товар
            Product::deleteProductById($id);

            // Перенаправляем пользователя на страницу управлениями товарами
            header("Location: /admin/product");
        }

        // Подключаем вид
        require_once(ROOT. '/views/admin_product/delete.php');
        return true;
    }

}
```

Додаток Р
CartController.php

```
<?php

/**
 * Контроллер CartController
 * Корзина
 */
class CartController
{

    /**
     * Action для добавления товара в корзину синхронным запросом<br/>
     * (для примера, не используется)
     * @param integer $id <p>id товара</p>
     */
    public function actionAdd($id)
    {
        // Добавляем товар в корзину
        Cart::addProduct($id);

        // Возвращаем пользователя на страницу с которой он пришел
        $referrer = $_SERVER['HTTP_REFERER'];
        header("Location: $referrer");
    }

    /**
     * Action для добавления товара в корзину при помощи асинхронного запроса (ajax)
     * @param integer $id <p>id товара</p>
     */
    public function actionAddAjax($id)
    {
        // Добавляем товар в корзину и печатаем результат: количество товаров в
корзине
        echo Cart::addProduct($id);
        return true;
    }

    /**
     * Action для добавления товара в корзину синхронным запросом
     * @param integer $id <p>id товара</p>
     */
    public function actionDelete($id)
    {
        // Удаляем заданный товар из корзины
        Cart::deleteProduct($id);

        // Возвращаем пользователя в корзину
        header("Location: /cart");
    }

    /**
     * Action для страницы "Корзина"
     */
    public function actionIndex()
    {
        // Список категорий для левого меню
        $categories = Category::getCategoriesList();

        // Получим идентификаторы и количество товаров в корзине
        $productsInCart = Cart::getProducts();
```

```
        if ($productsInCart) {
            // Если в корзине есть товары, получаем полную информацию о товарах для
списка
            // Получаем массив только с идентификаторами товаров
            $productsIds = array_keys($productsInCart);

            // Получаем массив с полной информацией о необходимых товарах
            $products = Product::getProdustsByIds($productsIds);

            // Получаем общую стоимость товаров
            $totalPrice = Cart::getTotalPrice($products);
        }

        // Подключаем вид
        require_once(ROOT. '/views/cart/index.php');
        return true;
    }

    /**
     * Action для страницы "Оформление покупки"
     */
    public function actionCheckout()
    {
        // Получием данные из корзины
        $productsInCart = Cart::getProducts();

        // Если товаров нет, отправляем пользователи искать товары на главную
        if ($productsInCart == false) {
            header("Location: /");
        }

        // Список категорий для левого меню
        $categories = Category::getCategoriesList();

        // Находим общую стоимость
        $productsIds = array_keys($productsInCart);
        $products = Product::getProdustsByIds($productsIds);
        $totalPrice = Cart::getTotalPrice($products);

        // Количество товаров
        $totalQuantity = Cart::countItems();

        // Поля для формы
        $userName = false;
        $userPhone = false;
        $userComment = false;

        // Статус успешного оформления заказа
        $result = false;

        // Проверяем является ли пользователь гостем
        if (!User::isGuest()) {
            // Если пользователь не гость
            // Получаем информацию о пользователе из БД
            $userId = User::checkLogged();
            $user = User::getUserById($userId);
            $userName = $user['name'];
        } else {
            // Если гость, поля формы останутся пустыми
            $userId = false;
        }
```

```
        // Обработка формы
        if (isset($_POST['submit'])) {
            // Если форма отправлена
            // Получаем данные из формы
            $userName = $_POST['userName'];
            $userPhone = $_POST['userPhone'];
            $userComment = $_POST['userComment'];

            // Флаг ошибок
            $errors = false;

            // Валидация полей
            if (!User::checkName($userName)) {
                $errors[] = 'Неправильное имя';
            }
            if (!User::checkPhone($userPhone)) {
                $errors[] = 'Неправильный телефон';
            }


            if ($errors == false) {
                // Если ошибок нет
                // Сохраняем заказ в базе данных
                $result = Order::save($userName, $userPhone, $userComment, $userId,
$productsInCart);

                if ($result) {
                    // Если заказ успешно сохранен
                    // Оповещаем администратора о новом заказе по почте
                    $adminEmail = 'php.start@mail.ru';
                    $message = '<a href="http://digital-
mafia.net/admin/orders">Список заказов</a>';
                    $subject = 'Новый заказ!';
                    mail($adminEmail, $subject, $message);

                    // Очищаем корзину
                    Cart::clear();
                }
            }
        }

        // Подключаем вид
        require_once(ROOT. '/views/cart/checkout.php');
        return true;
    }

}
```

Додаток С
CatalogController.php

```
<?php

/**
 * Контроллер CatalogController
 * Каталог товаров
 */
class CatalogController
{

    /**
     * Action для страницы "Каталог товаров"
     */
    public function actionIndex()
    {
        // Список категорий для левого меню
        $categories = Category::getCategoriesList();

        // Список последних товаров
        $latestProducts = Product::getLatestProducts(12);

        // Подключаем вид
        require_once(ROOT. '/views/catalog/index.php');
        return true;
    }

    /**
     * Action для страницы "Категория товаров"
     */
    public function actionCategory($categoryId, $page = 1)
    {
        // Список категорий для левого меню
        $categories = Category::getCategoriesList();

        // Список товаров в категории
        $categoryProducts = Product::getProductsListByCategory($categoryId, $page);

        // Общее количетсво товаров (необходимо для постраничной навигации)
        $total = Product::getTotalProductsInCategory($categoryId);

        // Создаем объект Pagination - постраничная навигация
        $pagination = new Pagination($total, $page, Product::SHOW_BY_DEFAULT, 'page-
');

        // Подключаем вид
        require_once(ROOT. '/views/catalog/category.php');
        return true;
    }

}
```

Додаток У
SiteController.php

```
<?php

/**
 * Контроллер CartController
 */
class SiteController
{

    /**
     * Action для главной страницы
     */
    public function actionIndex()
    {
        // Список категорий для левого меню
        $categories = Category::getCategoriesList();

        // Список последних товаров
        $latestProducts = Product::getLatestProducts(6);

        // Список товаров для слайдера
        $sliderProducts = Product::getRecommendedProducts();

        // Подключаем вид
        require_once(ROOT. '/views/site/index.php');
        return true;
    }

    /**
     * Action для страницы "Контакты"
     */
    public function actionContact()
    {

        // Переменные для формы
        $userEmail = false;
        $userText = false;
        $result = false;

        // Обработка формы
        if (isset($_POST['submit'])) {
            // Если форма отправлена
            // Получаем данные из формы
            $userEmail = $_POST['userEmail'];
            $userText = $_POST['userText'];

            // Флаг ошибок
            $errors = false;

            // Валидация полей
            if (!User::checkEmail($userEmail)) {
                $errors[] = 'Неправильный email';
            }

            if ($errors == false) {
                // Если ошибок нет
                // Отправляем письмо администратору
                $adminEmail = 'php.start@mail.ru';
                $message = "Текст: {$userText}. От {$userEmail}";
                $subject = 'Тема письма';
```

```
                $result = mail($adminEmail, $subject, $message);
                $result = true;
            }
        }

        // Подключаем вид
        require_once(ROOT. '/views/site/contact.php');
        return true;
    }

    /**
     * Action для страницы "О магазине"
     */
    public function actionAbout()
    {
        // Подключаем вид
        require_once(ROOT. '/views/site/about.php');
        return true;
    }

}
```

ProductController.php

```
<?php

/**
 * Контроллер ProductController
 * Товар
 */
class ProductController
{

    /**
     * Action для страницы просмотра товара
     * @param integer $productId <p>id товара</p>
     */
    public function actionView($productId)
    {
        // Список категорий для левого меню
        $categories = Category::getCategoriesList();

        // Получаем инфомрацию о товаре
        $product = Product::getProductById($productId);

        // Подключаем вид
        require_once(ROOT. '/views/product/view.php');
        return true;
    }

}
```

Додаток Ф
UserController.php

```php
<?php

/**
 * Контроллер UserController
 */
class UserController
{
    /**
     * Action для страницы "Регистрация"
     */
    public function actionRegister()
    {
        // Переменные для формы
        $name = false;
        $email = false;
        $password = false;
        $result = false;

        // Обработка формы
        if (isset($_POST['submit'])) {
            // Если форма отправлена
            // Получаем данные из формы
            $name = $_POST['name'];
            $email = $_POST['email'];
            $password = $_POST['password'];

            // Флаг ошибок
            $errors = false;

            // Валидация полей
            if (!User::checkName($name)) {
                $errors[] = 'Имя не должно быть короче 2-х символов';
            }
            if (!User::checkEmail($email)) {
                $errors[] = 'Неправильный email';
            }
            if (!User::checkPassword($password)) {
                $errors[] = 'Пароль не должен быть короче 6-ти символов';
            }
            if (User::checkEmailExists($email)) {
                $errors[] = 'Такой email уже используется';
            }

            if ($errors == false) {
                // Если ошибок нет
                // Регистрируем пользователя
                $result = User::register($name, $email, $password);
            }
        }

        // Подключаем вид
        require_once(ROOT. '/views/user/register.php');
        return true;
    }

    /**
     * Action для страницы "Вход на сайт"
     */
    public function actionLogin()
```

```
    {
        // Переменные для формы
        $email = false;
        $password = false;

        // Обработка формы
        if (isset($_POST['submit'])) {
            // Если форма отправлена
            // Получаем данные из формы
            $email = $_POST['email'];
            $password = $_POST['password'];

            // Флаг ошибок
            $errors = false;

            // Валидация полей
            if (!User::checkEmail($email)) {
                $errors[] = 'Неправильный email';
            }
            if (!User::checkPassword($password)) {
                $errors[] = 'Пароль не должен быть короче 6-ти символов';
            }

            // Проверяем существует ли пользователь
            $userId = User::checkUserData($email, $password);

            if ($userId == false) {
                // Если данные неправильные - показываем ошибку
                $errors[] = 'Неправильные данные для входа на сайт';
            } else {
                // Если данные правильные, запоминаем пользователя (сессия)
                User::auth($userId);

                // Перенаправляем пользователя в закрытую часть - кабинет
                header("Location: /cabinet");
            }
        }

        // Подключаем вид
        require_once(ROOT. '/views/user/login.php');
        return true;
    }

    /**
     * Удаляем данные о пользователе из сессии
     */
    public function actionLogout()
    {
        // Стартуем сессию
        session_start();

        // Удаляем информацию о пользователе из сессии
        unset($_SESSION["user"]);

        // Перенаправляем пользователя на главную страницу
        header("Location: /");
    }

}
```

Додаток Х
Category.php

```
<?php

/**
 * Класс Category - модель для работы с категориями товаров
 */
class Category
{

    /**
     * Возвращает массив категорий для списка на сайте
     * @return array <p>Массив с категориями</p>
     */
    public static function getCategoriesList()
    {
        // Соединение с БД
        $db = Db::getConnection();

        // Запрос к БД
        $result = $db->query('SELECT id, name FROM category WHERE status = "1" ORDER
BY sort_order, name ASC');

        // Получение и возврат результатов
        $i = 0;
        $categoryList = array();
        while ($row = $result->fetch()) {
            $categoryList[$i]['id'] = $row['id'];
            $categoryList[$i]['name'] = $row['name'];
            $i++;
        }
        return $categoryList;
    }

    /**
     * Возвращает массив категорий для списка в админпанели <br/>
     * (при этом в результат попадают и включенные и выключенные категории)
     * @return array <p>Массив категорий</p>
     */
    public static function getCategoriesListAdmin()
    {
        // Соединение с БД
        $db = Db::getConnection();

        // Запрос к БД
        $result = $db->query('SELECT id, name, sort_order, status FROM category ORDER
BY sort_order ASC');

        // Получение и возврат результатов
        $categoryList = array();
        $i = 0;
        while ($row = $result->fetch()) {
            $categoryList[$i]['id'] = $row['id'];
            $categoryList[$i]['name'] = $row['name'];
            $categoryList[$i]['sort_order'] = $row['sort_order'];
            $categoryList[$i]['status'] = $row['status'];
            $i++;
        }
        return $categoryList;
    }
```

```
/**
 * Удаляет категорию с заданным id
 * @param integer $id
 * @return boolean <p>Результат выполнения метода</p>
 */
public static function deleteCategoryById($id)
{
    // Соединение с БД
    $db = Db::getConnection();

    // Текст запроса к БД
    $sql = 'DELETE FROM category WHERE id = :id';

    // Получение и возврат результатов. Используется подготовленный запрос
    $result = $db->prepare($sql);
    $result->bindParam(':id', $id, PDO::PARAM_INT);
    return $result->execute();
}

/**
 * Редактирование категории с заданным id
 * @param integer $id <p>id категории</p>
 * @param string $name <p>Название</p>
 * @param integer $sortOrder <p>Порядковый номер</p>
 * @param integer $status <p>Статус <i>(включено "1", выключено "0")</i></p>
 * @return boolean <p>Результат выполнения метода</p>
 */
public static function updateCategoryById($id, $name, $sortOrder, $status)
{
    // Соединение с БД
    $db = Db::getConnection();

    // Текст запроса к БД
    $sql = "UPDATE category
        SET
            name = :name,
            sort_order = :sort_order,
            status = :status
        WHERE id = :id";

    // Получение и возврат результатов. Используется подготовленный запрос
    $result = $db->prepare($sql);
    $result->bindParam(':id', $id, PDO::PARAM_INT);
    $result->bindParam(':name', $name, PDO::PARAM_STR);
    $result->bindParam(':sort_order', $sortOrder, PDO::PARAM_INT);
    $result->bindParam(':status', $status, PDO::PARAM_INT);
    return $result->execute();
}

/**
 * Возвращает категорию с указанным id
 * @param integer $id <p>id категории</p>
 * @return array <p>Массив с информацией о категории</p>
 */
public static function getCategoryById($id)
{
    // Соединение с БД
    $db = Db::getConnection();

    // Текст запроса к БД
    $sql = 'SELECT * FROM category WHERE id = :id';

    // Используется подготовленный запрос
```

```
        $result = $db->prepare($sql);
        $result->bindParam(':id', $id, PDO::PARAM_INT);

        // Указываем, что хотим получить данные в виде массива
        $result->setFetchMode(PDO::FETCH_ASSOC);

        // Выполняем запрос
        $result->execute();

        // Возвращаем данные
        return $result->fetch();
    }

    /**
     * Возвращает текстое пояснение статуса для категории :<br/>
     * <i>0 - Скрыта, 1 - Отображается</i>
     * @param integer $status <p>Статус</p>
     * @return string <p>Текстовое пояснение</p>
     */
    public static function getStatusText($status)
    {
        switch ($status) {
            case '1':
                return 'Отображается';
                break;
            case '0':
                return 'Скрыта';
                break;
        }
    }

    /**
     * Добавляет новую категорию
     * @param string $name <p>Название</p>
     * @param integer $sortOrder <p>Порядковый номер</p>
     * @param integer $status <p>Статус <i>(включено "1", выключено "0")</i></p>
     * @return boolean <p>Результат добавления записи в таблицу</p>
     */
    public static function createCategory($name, $sortOrder, $status)
    {
        // Соединение с БД
        $db = Db::getConnection();

        // Текст запроса к БД
        $sql = 'INSERT INTO category (name, sort_order, status) '
               . 'VALUES (:name, :sort_order, :status)';

        // Получение и возврат результатов. Используется подготовленный запрос
        $result = $db->prepare($sql);
        $result->bindParam(':name', $name, PDO::PARAM_STR);
        $result->bindParam(':sort_order', $sortOrder, PDO::PARAM_INT);
        $result->bindParam(':status', $status, PDO::PARAM_INT);
        return $result->execute();
    }

}
```

Додаток Ц
Order.php

```
<?php

/**
 * Класс Order - модель для работы с заказами
 */
class Order
{

    /**
     * Сохранение заказа
     * @param string $userName <p>Имя</p>
     * @param string $userPhone <p>Телефон</p>
     * @param string $userComment <p>Комментарий</p>
     * @param integer $userId <p>id пользователя</p>
     * @param array $products <p>Массив с товарами</p>
     * @return boolean <p>Результат выполнения метода</p>
     */
    public static function save($userName, $userPhone, $userComment, $userId,
$products)
    {
        // Соединение с БД
        $db = Db::getConnection();

        // Текст запроса к БД
        $sql = 'INSERT INTO product_order (user_name, user_phone, user_comment,
user_id, products) '
               . 'VALUES (:user_name, :user_phone, :user_comment, :user_id,
:products)';

        $products = json_encode($products);

        $result = $db->prepare($sql);
        $result->bindParam(':user_name', $userName, PDO::PARAM_STR);
        $result->bindParam(':user_phone', $userPhone, PDO::PARAM_STR);
        $result->bindParam(':user_comment', $userComment, PDO::PARAM_STR);
        $result->bindParam(':user_id', $userId, PDO::PARAM_STR);
        $result->bindParam(':products', $products, PDO::PARAM_STR);

        return $result->execute();
    }

    /**
     * Возвращает список заказов
     * @return array <p>Список заказов</p>
     */
    public static function getOrdersList()
    {
        // Соединение с БД
        $db = Db::getConnection();

        // Получение и возврат результатов
        $result = $db->query('SELECT id, user_name, user_phone, date, status FROM
product_order ORDER BY id DESC');
        $ordersList = array();
        $i = 0;
        while ($row = $result->fetch()) {
            $ordersList[$i]['id'] = $row['id'];
            $ordersList[$i]['user_name'] = $row['user_name'];
            $ordersList[$i]['user_phone'] = $row['user_phone'];
```

```
            $ordersList[$i]['date'] = $row['date'];
            $ordersList[$i]['status'] = $row['status'];
            $i++;
        }
        return $ordersList;
    }

    /**
     * Возвращает текстое пояснение статуса для заказа :<br/>
     * <i>1 - Новый заказ, 2 - В обработке, 3 - Доставляется, 4 - Закрыт</i>
     * @param integer $status <p>Статус</p>
     * @return string <p>Текстовое пояснение</p>
     */
    public static function getStatusText($status)
    {
        switch ($status) {
            case '1':
                return 'Новый заказ';
                break;
            case '2':
                return 'В обработке';
                break;
            case '3':
                return 'Доставляется';
                break;
            case '4':
                return 'Закрыт';
                break;
        }
    }

    /**
     * Возвращает заказ с указанным id
     * @param integer $id <p>id</p>
     * @return array <p>Массив с информацией о заказе</p>
     */
    public static function getOrderById($id)
    {
        // Соединение с БД
        $db = Db::getConnection();

        // Текст запроса к БД
        $sql = 'SELECT * FROM product_order WHERE id = :id';

        $result = $db->prepare($sql);
        $result->bindParam(':id', $id, PDO::PARAM_INT);

        // Указываем, что хотим получить данные в виде массива
        $result->setFetchMode(PDO::FETCH_ASSOC);

        // Выполняем запрос
        $result->execute();

        // Возвращаем данные
        return $result->fetch();
    }

    /**
     * Удаляет заказ с заданным id
     * @param integer $id <p>id заказа</p>
     * @return boolean <p>Результат выполнения метода</p>
     */
    public static function deleteOrderById($id)
```

```
    {
        // Соединение с БД
        $db = Db::getConnection();

        // Текст запроса к БД
        $sql = 'DELETE FROM product_order WHERE id = :id';

        // Получение и возврат результатов. Используется подготовленный запрос
        $result = $db->prepare($sql);
        $result->bindParam(':id', $id, PDO::PARAM_INT);
        return $result->execute();
    }

    /**
     * Редактирует заказ с заданным id
     * @param integer $id <p>id товара</p>
     * @param string $userName <p>Имя клиента</p>
     * @param string $userPhone <p>Телефон клиента</p>
     * @param string $userComment <p>Комментарий клиента</p>
     * @param string $date <p>Дата оформления</p>
     * @param integer $status <p>Статус <i>(включено "1", выключено "0")</i></p>
     * @return boolean <p>Результат выполнения метода</p>
     */
    public static function updateOrderById($id, $userName, $userPhone, $userComment,
$date, $status)
    {
        // Соединение с БД
        $db = Db::getConnection();

        // Текст запроса к БД
        $sql = "UPDATE product_order
            SET
                user_name = :user_name,
                user_phone = :user_phone,
                user_comment = :user_comment,
                date = :date,
                status = :status
            WHERE id = :id";

        // Получение и возврат результатов. Используется подготовленный запрос
        $result = $db->prepare($sql);
        $result->bindParam(':id', $id, PDO::PARAM_INT);
        $result->bindParam(':user_name', $userName, PDO::PARAM_STR);
        $result->bindParam(':user_phone', $userPhone, PDO::PARAM_STR);
        $result->bindParam(':user_comment', $userComment, PDO::PARAM_STR);
        $result->bindParam(':date', $date, PDO::PARAM_STR);
        $result->bindParam(':status', $status, PDO::PARAM_INT);
        return $result->execute();
    }

}
```

Додаток Ш
Product.php

```php
<?php

/**
 * Класс Product - модель для работы с товарами
 */
class Product
{

    // Количество отображаемых товаров по умолчанию
    const SHOW_BY_DEFAULT = 6;

    /**
     * Возвращает массив последних товаров
     * @param type $count [optional] <p>Количество</p>
     * @param type $page [optional] <p>Номер текущей страницы</p>
     * @return array <p>Массив с товарами</p>
     */
    public static function getLatestProducts($count = self::SHOW_BY_DEFAULT)
    {
        // Соединение с БД
        $db = Db::getConnection();

        // Текст запроса к БД
        $sql = 'SELECT id, name, price, is_new FROM product '
               . 'WHERE status = "1" ORDER BY id DESC '
               . 'LIMIT :count';

        // Используется подготовленный запрос
        $result = $db->prepare($sql);
        $result->bindParam(':count', $count, PDO::PARAM_INT);

        // Указываем, что хотим получить данные в виде массива
        $result->setFetchMode(PDO::FETCH_ASSOC);

        // Выполнение коменды
        $result->execute();

        // Получение и возврат результатов
        $i = 0;
        $productsList = array();
        while ($row = $result->fetch()) {
            $productsList[$i]['id'] = $row['id'];
            $productsList[$i]['name'] = $row['name'];
            $productsList[$i]['price'] = $row['price'];
            $productsList[$i]['is_new'] = $row['is_new'];
            $i++;
        }
        return $productsList;
    }

    /**
     * Возвращает список товаров в указанной категории
     * @param type $categoryId <p>id категории</p>
     * @param type $page [optional] <p>Номер страницы</p>
     * @return type <p>Массив с товарами</p>
     */
    public static function getProductsListByCategory($categoryId, $page = 1)
    {
        $limit = Product::SHOW_BY_DEFAULT;
```

```
        // Смещение (для запроса)
        $offset = ($page - 1) * self::SHOW_BY_DEFAULT;

        // Соединение с БД
        $db = Db::getConnection();

        // Текст запроса к БД
        $sql = 'SELECT id, name, price, is_new FROM product '
               . 'WHERE status = 1 AND category_id = :category_id '
               . 'ORDER BY id ASC LIMIT :limit OFFSET :offset';

        // Используется подготовленный запрос
        $result = $db->prepare($sql);
        $result->bindParam(':category_id', $categoryId, PDO::PARAM_INT);
        $result->bindParam(':limit', $limit, PDO::PARAM_INT);
        $result->bindParam(':offset', $offset, PDO::PARAM_INT);

        // Выполнение коменды
        $result->execute();

        // Получение и возврат результатов
        $i = 0;
        $products = array();
        while ($row = $result->fetch()) {
            $products[$i]['id'] = $row['id'];
            $products[$i]['name'] = $row['name'];
            $products[$i]['price'] = $row['price'];
            $products[$i]['is_new'] = $row['is_new'];
            $i++;
        }
        return $products;
    }

    /**
     * Возвращает продукт с указанным id
     * @param integer $id <p>id товара</p>
     * @return array <p>Массив с информацией о товаре</p>
     */
    public static function getProductById($id)
    {
        // Соединение с БД
        $db = Db::getConnection();

        // Текст запроса к БД
        $sql = 'SELECT * FROM product WHERE id = :id';

        // Используется подготовленный запрос
        $result = $db->prepare($sql);
        $result->bindParam(':id', $id, PDO::PARAM_INT);

        // Указываем, что хотим получить данные в виде массива
        $result->setFetchMode(PDO::FETCH_ASSOC);

        // Выполнение коменды
        $result->execute();

        // Получение и возврат результатов
        return $result->fetch();
    }

    /**
     * Возвращаем количество товаров в указанной категории
     * @param integer $categoryId
```

```
     * @return integer
     */
    public static function getTotalProductsInCategory($categoryId)
    {
        // Соединение с БД
        $db = Db::getConnection();

        // Текст запроса к БД
        $sql = 'SELECT count(id) AS count FROM product WHERE status="1" AND
category_id = :category_id';

        // Используется подготовленный запрос
        $result = $db->prepare($sql);
        $result->bindParam(':category_id', $categoryId, PDO::PARAM_INT);

        // Выполнение коменды
        $result->execute();

        // Возвращаем значение count - количество
        $row = $result->fetch();
        return $row['count'];
    }

    /**
     * Возвращает список товаров с указанными индентификторами
     * @param array $idsArray <p>Массив с идентификаторами</p>
     * @return array <p>Массив со списком товаров</p>
     */
    public static function getProdustsByIds($idsArray)
    {
        // Соединение с БД
        $db = Db::getConnection();

        // Превращаем массив в строку для формирования условия в запросе
        $idsString = implode(',', $idsArray);

        // Текст запроса к БД
        $sql = "SELECT * FROM product WHERE status='1' AND id IN ($idsString)";

        $result = $db->query($sql);

        // Указываем, что хотим получить данные в виде массива
        $result->setFetchMode(PDO::FETCH_ASSOC);

        // Получение и возврат результатов
        $i = 0;
        $products = array();
        while ($row = $result->fetch()) {
            $products[$i]['id'] = $row['id'];
            $products[$i]['code'] = $row['code'];
            $products[$i]['name'] = $row['name'];
            $products[$i]['price'] = $row['price'];
            $i++;
        }
        return $products;
    }

    /**
     * Возвращает список рекомендуемых товаров
     * @return array <p>Массив с товарами</p>
     */
    public static function getRecommendedProducts()
    {
```

```
        // Соединение с БД
        $db = Db::getConnection();

        // Получение и возврат результатов
        $result = $db->query('SELECT id, name, price, is_new FROM product '
               . 'WHERE status = "1" AND is_recommended = "1" '
               . 'ORDER BY id DESC');
        $i = 0;
        $productsList = array();
        while ($row = $result->fetch()) {
            $productsList[$i]['id'] = $row['id'];
            $productsList[$i]['name'] = $row['name'];
            $productsList[$i]['price'] = $row['price'];
            $productsList[$i]['is_new'] = $row['is_new'];
            $i++;
        }
        return $productsList;
    }

    /**
     * Возвращает список товаров
     * @return array <p>Массив с товарами</p>
     */
    public static function getProductsList()
    {
        // Соединение с БД
        $db = Db::getConnection();

        // Получение и возврат результатов
        $result = $db->query('SELECT id, name, price, code FROM product ORDER BY id
ASC');
        $productsList = array();
        $i = 0;
        while ($row = $result->fetch()) {
            $productsList[$i]['id'] = $row['id'];
            $productsList[$i]['name'] = $row['name'];
            $productsList[$i]['code'] = $row['code'];
            $productsList[$i]['price'] = $row['price'];
            $i++;
        }
        return $productsList;
    }

    /**
     * Удаляет товар с указанным id
     * @param integer $id <p>id товара</p>
     * @return boolean <p>Результат выполнения метода</p>
     */
    public static function deleteProductById($id)
    {
        // Соединение с БД
        $db = Db::getConnection();

        // Текст запроса к БД
        $sql = 'DELETE FROM product WHERE id = :id';

        // Получение и возврат результатов. Используется подготовленный запрос
        $result = $db->prepare($sql);
        $result->bindParam(':id', $id, PDO::PARAM_INT);
        return $result->execute();
    }

    /**
```

```
     * Редактирует товар с заданным id
     * @param integer $id <p>id товара</p>
     * @param array $options <p>Массив с информацей о товаре</p>
     * @return boolean <p>Результат выполнения метода</p>
     */
    public static function updateProductById($id, $options)
    {
        // Соединение с БД
        $db = Db::getConnection();

        // Текст запроса к БД
        $sql = "UPDATE product
            SET
                name = :name,
                code = :code,
                price = :price,
                category_id = :category_id,
                brand = :brand,
                availability = :availability,
                description = :description,
                is_new = :is_new,
                is_recommended = :is_recommended,
                status = :status
            WHERE id = :id";

        // Получение и возврат результатов. Используется подготовленный запрос
        $result = $db->prepare($sql);
        $result->bindParam(':id', $id, PDO::PARAM_INT);
        $result->bindParam(':name', $options['name'], PDO::PARAM_STR);
        $result->bindParam(':code', $options['code'], PDO::PARAM_STR);
        $result->bindParam(':price', $options['price'], PDO::PARAM_STR);
        $result->bindParam(':category_id', $options['category_id'], PDO::PARAM_INT);
        $result->bindParam(':brand', $options['brand'], PDO::PARAM_STR);
        $result->bindParam(':availability', $options['availability'],
PDO::PARAM_INT);
        $result->bindParam(':description', $options['description'], PDO::PARAM_STR);
        $result->bindParam(':is_new', $options['is_new'], PDO::PARAM_INT);
        $result->bindParam(':is_recommended', $options['is_recommended'],
PDO::PARAM_INT);
        $result->bindParam(':status', $options['status'], PDO::PARAM_INT);
        return $result->execute();
    }

    /**
     * Добавляет новый товар
     * @param array $options <p>Массив с информацией о товаре</p>
     * @return integer <p>id добавленной в таблицу записи</p>
     */
    public static function createProduct($options)
    {
        // Соединение с БД
        $db = Db::getConnection();

        // Текст запроса к БД
        $sql = 'INSERT INTO product '
               . '(name, code, price, category_id, brand, availability,'
               . 'description, is_new, is_recommended, status)'
               . 'VALUES '
               . '(:name, :code, :price, :category_id, :brand, :availability,'
               . ':description, :is_new, :is_recommended, :status)';

        // Получение и возврат результатов. Используется подготовленный запрос
        $result = $db->prepare($sql);
```

```
        $result->bindParam(':name', $options['name'], PDO::PARAM_STR);
        $result->bindParam(':code', $options['code'], PDO::PARAM_STR);
        $result->bindParam(':price', $options['price'], PDO::PARAM_STR);
        $result->bindParam(':category_id', $options['category_id'], PDO::PARAM_INT);
        $result->bindParam(':brand', $options['brand'], PDO::PARAM_STR);
        $result->bindParam(':availability', $options['availability'],
PDO::PARAM_INT);
        $result->bindParam(':description', $options['description'], PDO::PARAM_STR);
        $result->bindParam(':is_new', $options['is_new'], PDO::PARAM_INT);
        $result->bindParam(':is_recommended', $options['is_recommended'],
PDO::PARAM_INT);
        $result->bindParam(':status', $options['status'], PDO::PARAM_INT);
        if ($result->execute()) {
            // Если запрос выполенен успешно, возвращаем id добавленной записи
            return $db->lastInsertId();
        }
        // Иначе возвращаем 0
        return 0;
    }

    /**
     * Возвращает текстое пояснение наличия товара:<br/>
     * <i>0 - Под заказ, 1 - В наличии</i>
     * @param integer $availability <p>Статус</p>
     * @return string <p>Текстовое пояснение</p>
     */
    public static function getAvailabilityText($availability)
    {
        switch ($availability) {
            case '1':
                return 'В наличии';
                break;
            case '0':
                return 'Под заказ';
                break;
        }
    }

    /**
     * Возвращает путь к изображению
     * @param integer $id
     * @return string <p>Путь к изображению</p>
     */
    public static function getImage($id)
    {
        // Название изображения-пустышки
        $noImage = 'no-image.jpg';

        // Путь к папке с товарами
        $path = '/upload/images/products/';

        // Путь к изображению товара
        $pathToProductImage = $path. $id. '.jpg';

        if (file_exists($_SERVER['DOCUMENT_ROOT'].$pathToProductImage)) {
            // Если изображение для товара существует
            // Возвращаем путь изображения товара
            return $pathToProductImage;
        }

        // Возвращаем путь изображения-пустышки
        return $path. $noImage;
    }
```

Додаток Щ
User.php

```
<?php

/**
 * Класс User - модель для работы с пользователями
 */
class User
{

    /**
     * Регистрация пользователя
     * @param string $name <p>Имя</p>
     * @param string $email <p>E-mail</p>
     * @param string $password <p>Пароль</p>
     * @return boolean <p>Результат выполнения метода</p>
     */
    public static function register($name, $email, $password)
    {
        // Соединение с БД
        $db = Db::getConnection();

        // Текст запроса к БД
        $sql = 'INSERT INTO user (name, email, password) '
               . 'VALUES (:name, :email, :password)';

        // Получение и возврат результатов. Используется подготовленный запрос
        $result = $db->prepare($sql);
        $result->bindParam(':name', $name, PDO::PARAM_STR);
        $result->bindParam(':email', $email, PDO::PARAM_STR);
        $result->bindParam(':password', $password, PDO::PARAM_STR);
        return $result->execute();
    }

    /**
     * Редактирование данных пользователя
     * @param integer $id <p>id пользователя</p>
     * @param string $name <p>Имя</p>
     * @param string $password <p>Пароль</p>
     * @return boolean <p>Результат выполнения метода</p>
     */
    public static function edit($id, $name, $password)
    {
        // Соединение с БД
        $db = Db::getConnection();

        // Текст запроса к БД
        $sql = "UPDATE user
            SET name = :name, password = :password
            WHERE id = :id";

        // Получение и возврат результатов. Используется подготовленный запрос
        $result = $db->prepare($sql);
        $result->bindParam(':id', $id, PDO::PARAM_INT);
        $result->bindParam(':name', $name, PDO::PARAM_STR);
        $result->bindParam(':password', $password, PDO::PARAM_STR);
        return $result->execute();
    }

    /**
     * Проверяем существует ли пользователь с заданными $email и $password
     * @param string $email <p>E-mail</p>
     * @param string $password <p>Пароль</p>
```

```
     * @return mixed : integer user id or false
     */
    public static function checkUserData($email, $password)
    {
        // Соединение с БД
        $db = Db::getConnection();

        // Текст запроса к БД
        $sql = 'SELECT * FROM user WHERE email = :email AND password = :password';

        // Получение результатов. Используется подготовленный запрос
        $result = $db->prepare($sql);
        $result->bindParam(':email', $email, PDO::PARAM_INT);
        $result->bindParam(':password', $password, PDO::PARAM_INT);
        $result->execute();

        // Обращаемся к записи
        $user = $result->fetch();

        if ($user) {
            // Если запись существует, возвращаем id пользователя
            return $user['id'];
        }
        return false;
    }

    /**
     * Запоминаем пользователя
     * @param integer $userId <p>id пользователя</p>
     */
    public static function auth($userId)
    {
        // Записываем идентификатор пользователя в сессию
        $_SESSION['user'] = $userId;
    }

    /**
     * Возвращает идентификатор пользователя, если он авторизирован.<br/>
     * Иначе перенаправляет на страницу входа
     * @return string <p>Идентификатор пользователя</p>
     */
    public static function checkLogged()
    {
        // Если сессия есть, вернем идентификатор пользователя
        if (isset($_SESSION['user'])) {
            return $_SESSION['user'];
        }

        header("Location: /user/login");
    }

    /**
     * Проверяет является ли пользователь гостем
     * @return boolean <p>Результат выполнения метода</p>
     */
    public static function isGuest()
    {
        if (isset($_SESSION['user'])) {
            return false;
        }
        return true;
    }
```

```
/**
 * Проверяет имя: не меньше, чем 2 символа
 * @param string $name <p>Имя</p>
 * @return boolean <p>Результат выполнения метода</p>
 */
public static function checkName($name)
{
    if (strlen($name) >= 2) {
        return true;
    }
    return false;
}

/**
 * Проверяет телефон: не меньше, чем 10 символов
 * @param string $phone <p>Телефон</p>
 * @return boolean <p>Результат выполнения метода</p>
 */
public static function checkPhone($phone)
{
    if (strlen($phone) >= 10) {
        return true;
    }
    return false;
}

/**
 * Проверяет имя: не меньше, чем 6 символов
 * @param string $password <p>Пароль</p>
 * @return boolean <p>Результат выполнения метода</p>
 */
public static function checkPassword($password)
{
    if (strlen($password) >= 6) {
        return true;
    }
    return false;
}

/**
 * Проверяет email
 * @param string $email <p>E-mail</p>
 * @return boolean <p>Результат выполнения метода</p>
 */
public static function checkEmail($email)
{
    if (filter_var($email, FILTER_VALIDATE_EMAIL)) {
        return true;
    }
    return false;
}

/**
 * Проверяет не занят ли email другим пользователем
 * @param type $email <p>E-mail</p>
 * @return boolean <p>Результат выполнения метода</p>
 */
public static function checkEmailExists($email)
{
    // Соединение с БД
    $db = Db::getConnection();

    // Текст запроса к БД
```

```
        $sql = 'SELECT COUNT(*) FROM user WHERE email = :email';

        // Получение результатов. Используется подготовленный запрос
        $result = $db->prepare($sql);
        $result->bindParam(':email', $email, PDO::PARAM_STR);
        $result->execute();

        if ($result->fetchColumn())
            return true;
        return false;
    }

    /**
     * Возвращает пользователя с указанным id
     * @param integer $id <p>id пользователя</p>
     * @return array <p>Массив с информацией о пользователе</p>
     */
    public static function getUserById($id)
    {
        // Соединение с БД
        $db = Db::getConnection();

        // Текст запроса к БД
        $sql = 'SELECT * FROM user WHERE id = :id';

        // Получение и возврат результатов. Используется подготовленный запрос
        $result = $db->prepare($sql);
        $result->bindParam(':id', $id, PDO::PARAM_INT);

        // Указываем, что хотим получить данные в виде массива
        $result->setFetchMode(PDO::FETCH_ASSOC);
        $result->execute();

        return $result->fetch();
    }

}
```

Додаток Ю
Index.php

```
<?php include ROOT. '/views/layouts/header_admin.php'; ?>

<section>
    <div class="container">
        <div class="row">

            <br/>

            <h4>Добрий день, адміністратор!</h4>

            <br/>

            <p>Вам доступні такі можливості:</p>

            <br/>

            <ul>
                <li><a href="/admin/product">Управління товарами</a></li>
                <li><a href="/admin/category">Управління категоріями</a></li>
                <li><a href="/admin/order">Управління заказами</a></li>
            </ul>

        </div>
    </div>
</section>

<?php include ROOT. '/views/layouts/footer_admin.php'; ?>
```

Додаток Я
Create.php

```
<?php include ROOT. '/views/layouts/header_admin.php'; ?>

<section>
    <div class="container">
        <div class="row">

            <br/>

            <div class="breadcrumbs">
                <ol class="breadcrumb">
                    <li><a href="/admin">Адмінпанель</a></li>
                    <li><a href="/admin/order">Управління категоріями</a></li>
                    <li class="active">Додати категорію</li>
                </ol>
            </div>


            <h4>Додати нову категорію</h4>

            <br/>

            <?php if (isset($errors) && is_array($errors)): ?>
                <ul>
                    <?php foreach ($errors as $error): ?>
                        <li> - <?php echo $error; ?></li>
                    <?php endforeach; ?>
                </ul>
            <?php endif; ?>

            <div class="col-lg-4">
                <div class="login-form">
                    <form action="#" method="post">

                        <p>Назва</p>
                        <input type="text" name="name" placeholder="" value="">

                        <p>Порядковый номер</p>
                        <input type="text" name="sort_order" placeholder="" value="">

                        <p>Статус</p>
                        <select name="status">
                            <option value="1"
selected="selected">Отображается</option>
                            <option value="0">Скрыта</option>
                        </select>

                        <br><br>

                        <input type="submit" name="submit" class="btn btn-default"
value="Зберегти">
                    </form>
                </div>
            </div>


        </div>
    </div>
</section>
```

Додаток АГ
Delete.php

```
<?php include ROOT. '/views/layouts/header_admin.php'; ?>

<section>
    <div class="container">
        <div class="row">

            <br/>

            <div class="breadcrumbs">
                <ol class="breadcrumb">
                    <li><a href="/admin">Адмінпанель</a></li>
                    <li><a href="/admin/category">Управління категоріями</a></li>
                    <li class="active">Видалити категорію</li>
                </ol>
            </div>


            <h4>Видалити категорію #<?php echo $id; ?></h4>


            <p>Ви дійсно хочете видалити цю категорію?</p>

            <form method="post">
                <input type="submit" name="submit" value="Видалити" />
            </form>

        </div>
    </div>
</section>

<?php include ROOT. '/views/layouts/footer_admin.php'; ?>
```

Index.php

```
<?php include ROOT. '/views/layouts/header_admin.php'; ?>

<section>
    <div class="container">
        <div class="row">

            <br/>

            <div class="breadcrumbs">
                <ol class="breadcrumb">
                    <li><a href="/admin">Адмін панель</a></li>
                    <li class="active">Управління категоріями</li>
                </ol>
            </div>

            <a href="/admin/category/create" class="btn btn-default back"><i
class="fa fa-plus"></i> Добавить категорию</a>

            <h4>Список категорій</h4>

            <br/>
```

```
            <table class="table-bordered table-striped table">
                <tr>
                    <th>ID категорії</th>
                    <th>Назва категорії</th>
                    <th>Порядковий номер</th>
                    <th>Статус</th>
                    <th></th>
                    <th></th>
                </tr>
                <?php foreach ($categoriesList as $category): ?>
                    <tr>
                        <td><?php echo $category['id']; ?></td>
                        <td><?php echo $category['name']; ?></td>
                        <td><?php echo $category['sort_order']; ?></td>
                        <td><?php echo Category::getStatusText($category['status']);
?></td>
                        <td><a href="/admin/category/update/<?php echo
$category['id']; ?>" title="Редагувати"><i class="fa fa-pencil-square-
o"></i></a></td>
                        <td><a href="/admin/category/delete/<?php echo
$category['id']; ?>" title="Видалити"><i class="fa fa-times"></i></a></td>
                    </tr>
                <?php endforeach; ?>
            </table>

        </div>
    </div>
</section>

<?php include ROOT. '/views/layouts/footer_admin.php'; ?>
```

Додаток А3

View.php

```
<?php include ROOT. '/views/layouts/header_admin.php'; ?>

<section>
    <div class="container">
        <div class="row">

            <br/>

            <div class="breadcrumbs">
                <ol class="breadcrumb">
                    <li><a href="/admin">Адмінпанель</a></li>
                    <li><a href="/admin/order">Управління замовленнями</a></li>
                    <li class="active">Огляд замовлення</li>
                </ol>
            </div>


            <h4>Огляд замовлення #<?php echo $order['id']; ?></h4>
            <br/>



            <h5>Інформація замовлення</h5>
            <table class="table-admin-small table-bordered table-striped table">
                <tr>
                    <td>Номер замовлення</td>
                    <td><?php echo $order['id']; ?></td>
                </tr>
                <tr>
                    <td>Ім'я клієнта</td>
                    <td><?php echo $order['user_name']; ?></td>
                </tr>
                <tr>
                    <td>Телефон клієнта</td>
                    <td><?php echo $order['user_phone']; ?></td>
                </tr>
                <tr>
                    <td>Коментар клієнта</td>
                    <td><?php echo $order['user_comment']; ?></td>
                </tr>
                <?php if ($order['user_id'] != 0): ?>
                    <tr>
                        <td>ID клієнта</td>
                        <td><?php echo $order['user_id']; ?></td>
                    </tr>
                <?php endif; ?>
                <tr>
                    <td><b>Статус замовлення</b></td>
                    <td><?php echo Order::getStatusText($order['status']); ?></td>
                </tr>
                <tr>
                    <td><b>Дата замовлення</b></td>
                    <td><?php echo $order['date']; ?></td>
                </tr>
            </table>

            <h5>Товари в замовленні</h5>
```

```
            <table class="table-admin-medium table-bordered table-striped table ">
                <tr>
                    <th>ID товару</th>
                    <th>Артикул товару</th>
                    <th>Назва</th>
                    <th>Ціна</th>
                    <th>Кількість</th>
                </tr>
                <?php foreach ($products as $product): ?>
                    <tr>
                        <td><?php echo $product['id']; ?></td>
                        <td><?php echo $product['code']; ?></td>
                        <td><?php echo $product['name']; ?></td>
                        <td>$<?php echo $product['price']; ?></td>
                        <td><?php echo $productsQuantity[$product['id']]; ?></td>
                    </tr>
                <?php endforeach; ?>
            </table>

            <a href="/admin/order/" class="btn btn-default back"><i class="fa fa-
arrow-left"></i> Назад</a>
        </div>


</section>

<?php include ROOT. '/views/layouts/footer_admin.php'; ?>
```

Додаток АИ

Create.php

```
<?php include ROOT. '/views/layouts/header_admin.php'; ?>

<section>
    <div class="container">
        <div class="row">

            <br/>

            <div class="breadcrumbs">
                <ol class="breadcrumb">
                    <li><a href="/admin">Адмінпанель</a></li>
                    <li><a href="/admin/product">Управління товарами</a></li>
                    <li class="active">Редагувати товар</li>
                </ol>
            </div>


            <h4>Додати новий товар</h4>

            <br/>

            <?php  if (isset($errors) && i_array($errors)): ?>
                <ul>
                    <?php  foreach ($errors as $error): ?>
                        <li> - <?php echo $error; </li>
                    <?php  endforeach;
                </ul>
            <?php endif;

            <div class="col-lg-4">
                <div class="login-form">
                    <form action="#" method="post" enctype="multipart/form-data">

                        <p>Назва товару</p>
                        <input type="text" name="name" placeholder="" value="">

                        <p>Артикул</p>
                        <input type="text" name="code" placeholder="" value="">

                        <p>Ціна, $</p>
                        <input type="text" name="price" placeholder="" value="">

                        <p>Категорія</p>
                        <select name="category_id">
                            <?php if (is_array($categoriesList)):
                                <?php foreach ($categoriesList as $category):
                                    <option value="<?php echo $category['id']; ">
                                        <?php  echo $category['name'];
                                    </option>
                                <?php endforeach;
                            <?php endif;
                        </select>

                        <br/><br/>

                        <p>Виробник</p>
                        <input type="text" name="brand" placeholder="" value="">
```

```
                        <p>Зображення товару</p>
                        <input type="file" name="image" placeholder="" value="">

                        <p>Детальний опис</p>
                        <textarea name="description"></textarea>

                        <br/><br/>

                        <p>Наявність на складі</p>
                        <select name="availability">
                            <option value="1" selected="selected">Так</option>
                            <option value="0">Ні</option>
                        </select>

                        <br/><br/>

                        <p>Новинка</p>
                        <select name="is_new">
                            <option value="1" selected="selected">Так</option>
                            <option value="0">Ні</option>
                        </select>

                        <br/><br/>

                        <p>Рекомендовані</p>
                        <select name="is_recommended">
                            <option value="1" selected="selected">Так</option>
                            <option value="0">Ні</option>
                        </select>

                        <br/><br/>

                        <p>Статус</p>
                        <select name="status">
                            <option value="1"
selected="selected">Відображається</option>
                            <option value="0">Прихований</option>
                        </select>

                        <br/><br/>

                        <input type="submit" name="submit" class="btn btn-default"
value="Зберегти">

                        <br/><br/>

                    </form>
                </div>
            </div>

        </div>
    </div>
</section>

<?php include ROOT. '/views/layouts/footer_admin.php';
```

Додаток АМ

Edit.php

```
<?php include ROOT. '/views/layouts/header.php'; ?>

<section>
    <div class="container">
        <div class="row">

            <div class="col-sm-4 col-sm-offset-4 padding-right">

                <?php if ($result): ?>
                    <p>Данні відредаговані!</p>
                <?php else: ?>
                    <?php if (isset($errors) && is_array($errors)): ?>
                        <ul>
                            <?php foreach ($errors as $error): ?>
                                <li> - <?php echo $error; ?></li>
                            <?php endforeach; ?>
                        </ul>
                    <?php endif; ?>

                    <div class="signup-form"><!--sign up form-->
                        <h2>Редагування даних</h2>
                        <form action="#" method="post">
                            <p>Ім'я:</p>
                            <input type="text" name="name" placeholder="Ім'я"
value="<?php echo $name; ?>"/>

                            <p>Пароль:</p>
                            <input type="password" name="password"
placeholder="Пароль" value="<?php echo $password; ?>"/>
                            <br/>
                            <input type="submit" name="submit" class="btn btn-
default" value="Зберегти" />
                        </form>
                    </div><!--/sign up form-->

                <?php endif; ?>
                <br/>
                <br/>
            </div>
        </div>
    </div>
</section>

<?php include ROOT. '/views/layouts/footer.php'; ?>
```