ЗМІСТ

Скорочення та умовні познаки 6
Вступ 7
1 Обгрунтування вибору інструментальних засобів для створення електронного магазину 8
1.1 Мова розмітки гіпертекстових документів HTML 8
1.2 Формальна мова опису зовнішнього вигляду документу CSS 11
1.3 Прототипно-орієнтована сценарна мова програмування JavaScript 13
1.4 Бібліотека jQuery 15
1.5 Скриптова мова програмування PHP 15
1.6 Мова структурованих запитів SQL 17
1.7 Система керування базами даних MySQL 19
1.8 Підхід до створення web-застосувань AJAX 21
1.9 Архітектура проекту 24
2 Опис етапів верстання макету 26
2.1 Шапка (Header) 27
2.2 Основна частина 29
2.3 Права колонка 30
2.4 Ліва колонка 31
2.5 Підвал (footer) 32
2.6 Каталог товарів 33
2.7 Кошик 35
2.8 Детальний перегляд товару 36
3 Проектування інформаційно-пошукової системи 37
3.1 Структура бази даних 37
3.2 Шаблон проектування MVC 38
4 Програмна реалізація інформаційно-пошукової системи 41
4.1 Реєстрація користувачів 41
4.2 Авторизація користувачів 43

4.3 Модуль кошику 44
4.3 Оформлення замовлення 46
4.4 Функція посторінкової навігації 49
4.5 Реалізація пошуку по магазину 50
Висновки 53
Перелік джерел посилання 54
Додаток А Програмний код реалізації інформаційної системи 56

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

БД	–	база даних;
ІС	–	інформаційна система;
ІТ	–	інформаційні технології;
СКБД	–	система керування базами даних;
ER	–	Entity-Relationship

ВСТУП

Сьогодні, інтернет-магазини стають одним з найбільш перспективних способів ведення та розвитку великого і малого бізнесу при зниженні витрат і збільшенні прибутку.

Інтернет магазин – це інтернет сайт, завдання і основне призначення якого є організація продажів різних товарів і послуг. Створення інтернет магазину, стає все більш актуальним завданням. За останніми даними продажі через інтернет у великих містах, досягають до 25%, при цьому фахівці підкреслюють тенденцію до зростання ціх продажів. Щороку кількість інтернет-магазинів збільшується, оскільки це дійсно прибутково для підприємств і зручно для покупця. Інтернет-магазин працює цілодобово і може продавати певні товари в автоматичному режимі без участі продавця. Так само не треба закуповувати товар наперед, а це істотна економія, на складських приміщеннях. У порівнянні зі звичайним магазином, територія продажів якого обмежується населенням міста чи району, територія охоплення інтернет-магазину збільшується на всю інтернет аудиторію.

Створення інтернет-магазину є трудомістким завданням. В оригіналі над створенням такого проекту працює кілька людей:

- дизайнер – відповідає за макет сайту, розробляє через консультації із замовником зовнішній вигляд сайту;
- верстальник – займається перенесенням, підготовленого дизайнером шаблону в гіпертекстовий документ. Сюди входить розмітка, стилізація і початкове програмування;
- програміст – займається програмуванням функціоналу сайту. Розробляє бази даних, впроваджує їх в структуру програми. Це найбільший обсяг роботи, тому цією частиною може займатися кілька програмістів.

Метою кваліфікаційної роботи є розроблення інформаційно-пошукової системи комерційного підприємства, представленої у вигляді Інтернет-магазину для покупки різних видів товарів через Інтернет.

У розробленому інтернет-магазині обов’язково повинні бути присутніми: рекламовані товари і послуги, прайс-лист, контактна інформація (телефон, факс, поштова адреса, e-mail), пропозиції різних варіантів оплати, надання рахунків, можливість здійснення покупок як фізичними так і юридичними особами. Необхідно передбачити відповідний дизайн, інтерфейс повинен бути зрозумілий навіть не досвідченим користувачеві.

Для досягненн поставленої мети в роботі потрібно вирішити наступні завдання:

- провести аналіз предметної області;
- обґрунтувати вибір засобів розробки інтернет-магазину;
- обґрунтувати вибір системи керування базами даних;
- розробити дизайн сайта і верстку макету;
- розробити структуру бази даних додатку;
- установити та налаштувати компоненти інтернет-магазину.

Структура кваліфікаційної роботи складається з вступу, чотирьох розділів, висновків, переліку посилань на 20 найменування, додатку. Повний обсяг проекту становить 69 сторінок, містить 29 малюнків.

1 ОБГРУНТУВАННЯ ВИБОРУ ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ ДЛЯ СТВОРЕННЯ ЕЛЕКТРОННОГО МАГАЗИНУ

1.1 Мова розмітки гіпертекстових документів HTML

HTML (HyperText Markup Language – мова розмітки гіпертекстових документів) – стандартна мова розмітки веб-сторінок в Інтернеті. Більшість

веб-сторінок створюються за допомогою мови HTML (або XHTML) [1][1)]. Документ HTML оброблюється браузером та відтворюється на екрані у звичному для людини вигляді.

HTML є похідною мовою від SGML, успадкувавши від неї визначення типу документу та ідеологію структурної розмітки тексту. HTML разом із каскадними таблицями стилів та вбудованими скриптами – це три основні технології побудови веб-сторінок. HTML впроваджує засоби для:

− створення структурованого документу шляхом позначення структурного складу тексту: заголовки, абзаци, списки, таблиці, цитати та інше;

− отримання інформації із Всесвітньої мережі через гіперпосилання;

− створення інтерактивних форм;

− включення зображень, звуку, відео та інших об'єктів до тексту.

Документ HTML. Для поліпшення взаємодії, SGML вимагає аби кожна похідна мова (HTML у тому числі) визначала свою кодову таблицю для кожного документа, яка складається з репертуара (перелік різноманітних символів) та позиції символа (перелік цифрових посилань на символи з репертуара). Кожен документ HTML – це послідовність символів з репертуара.

HTML використовує найповнішу кодову таблицю UCS (Universal Character Set – Універсальний Набір Символів). Проте, однієї кодової таблиці недостатньо для того, щоб браузери могли правильно відтворювати документи HTML. Для цього браузерам потрібно «знати» специфічну кодову таблицю документа, яку автор має зазначати завжди в елементі meta із параметром charset. За замовчуванням використовується кодова таблиця ISO-8859-1, відома також як Latin-1.

Розмітка в HTML складається з чотирьох основних компонентів: елементів (та їхніх атрибутів), базових типів даних, символьних мнемонік та декларації типу документа.

[1)] [1] HTML: HyperText Markup Language. URL: http://uk.wikipedia.org/wiki/HTML (дата звернення 18.05.2019)

Документ HTML 4.01 складається з трьох частин [2][1)]:

1) Декларація типу документу (Document type declaration, Doctype), на початку документа, в якій визначається тип документа (DTD).

2) Шапка документу (знаходиться в межах елементу head), в якій записані загальні технічні відомості або додаткова інформація про документ, яка не відтворюється безпосередньо в браузері.

3) Тіло документу (може знаходитися в елементах body або frameset), в якому міститься основна інформація документа.

Елементи являють собою базові компоненти розмітки HTML. Кожен елемент має дві основні властивості: атрибути та зміст (контент). Існують певні настанови щодо кожного атрибута та контента елемента, які треба виконувати задля того, щоб HTML-документ був визнаний валідним.

У елемента є початковий тег, який має вигляд <element-name>, та кінцевий тег, який має вигляд </element-name>. Атрибути елемента записуються в початковому тегу одразу після назви елемента, контент елемента записується між його двома тегами. Наприклад: <element-name element-attribute=”attribute-value”>контент елемента</element-name>.

Деякі елементи, наприклад br, не містять контенту, тож і не мають кінцевого тега. Елемент може не мати початкового та кінцевого тега (наприклад, елемент head), проте він завжди буде представлений в документі.

Більшість з атрибутів елементу являє собою пару «назва-значення», розділених між собою знаком рівняння, та записаних у початковому тегу одразу після назви елемента. Значення атрибуту може бути окреслено лапками (подвійними або одиничними), також, якщо значення атрибуту складається з певних символів, його можна не виділяти лапками зліва. Проте, не виділення значення атрибутів в лапки вважається небезпечним кодом. На відміну від атрибутів виду «назва-значення», є певні атрибути, що впливають на еле-

[1)] [2] Комолова Н. В., Яковлева Е. С. HTML, XHTML и CSS. СПб.: Питер, 2012. 300 с.

мент, назва яких лише з'явилась в початковому тегу (наприклад, атрибут ismap елементу img).

Оскільки HTML є похідною мовою від SGML, усі типи даних HTML ґрунтуються на базових типах даних SGML (наприклад, PCDATA, CDATA, NAME, ID, NUMBER).

Кожен елемент має дві властивості – атрибути і вміст, які мають певні значення. Всі можливі значення цих двох властивостей прописуються відповідно до визначених у DTD типів даних [3][1)].

1.2 Формальна мова опису зовнішнього вигляду документу CSS

CSS використовується авторами та відвідувачами веб-сторінок, щоб визначити кольори, шрифти, верстку та інші аспекти вигляду сторінки. Одна з головних переваг – можливість розділити зміст сторінки (або контент, наповнення, зазвичай HTML, XML або подібна мова розмітки) від вигляду документу (що описується в CSS) [4][2)].

Таке розділення може покращити сприйняття та доступність контенту, забезпечити більшу гнучкість та контроль за відображенням контенту в різних умовах, зробити контент більш структурованим та простим, прибрати повтори тощо. CSS також дозволяє адаптувати контент до різних умов відображення (на екрані монітора, мобільного пристрою (КПК), у роздрукованому вигляді, на екрані телевізора, пристроях з підтримкою шрифту Брайля або голосових браузерах та ін.)

Один і той самий HTML або XML документ може бути відображений по-різному залежно від використаного CSS.

[1)] [3] Фрэйн Б. HTML5 и CSS3. Разработка сайтов для любых браузеров и устройств. СПБ.: Питер, 2014. 298 с.

[2)] [4] CSS: Cascading Style Sheets. URL:http://uk.wikipedia.org/wiki/CSS (дата звернення 18.05.2019)

Стандарт CSS визначає порядок та діапазон застосування стилів, те, в якій послідовності і для яких елементів застосовуються стилі. Таким чином, використовується принцип каскадності, коли для елементів вказується лише та інформація про стилі, що змінилася або не визначена загальнішими стилями. Переваги [5][1)]:

− інформація про стиль для усього сайту або його частин може міститися в одному .css-файлі, що дозволяє швидко робити зміни в дизайні та презентації сторінок;

− різна інформація про стилі для різних типів користувачів: наприклад, великий розмір шрифту для користувачів з послабленим зором, стилі для виводу сторінки на принтер, стиль для мобільних пристроїв;

− сторінки зменшуються в об'ємі та стають більш структурованими, оскільки інформація про стилі відділена від тексту та має певні правила застосування і сторінка побудована з урахуванням їх;

− прискорення завантаження сторінок і зменшення обсягів інформації, що передається, навантаження на сервер та канал передачі. Досягається за рахунок того, що сучасні браузери здатні кешувати (запам'ятовувати) інформацію про стилі і використовувати для всіх сторінок, а не завантажувати для кожної.

CSS має порівняно простий синтаксис і використовує небагато англійських слів для найменування різних складових стилю. Стилі складаються зі списку правил. Кожне правило має один або більше селектор (Selector) та блок визначення (Declaration block). Блок визначення складається із оточеного фігурними дужками списку властивостей [6][2)].

---

1) [5] Дакетт, Д. HTML и CSS. Разработка и дизайн веб-сайтов. М.: Эксмо, 2013. 480 с.

2) [6] Дронов, В. HTML 5, CSS 3 и Web 2.0. Разработка современных Web- сайтов. СПб.: БХВ-Петербург, 2011. 416 с

1.3 Прототипно-орієнтована сценарна мова програмування JavaScript

JavaScript (JS) – динамічна, об’єктно-орієнтована мова програмування [7][1)]. Реалізація стандарту ECMAScript. Найчастіше використовується як частина браузера, що надає можливість коду на стороні клієнта (такому, що виконується на пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд веб-сторінки. Мова JavaScript також використовується для програмування на стороні серверу (подібно до таких мов програмування, як Java і C#), розробки ігор, стаціонарних та мобільних додатків, сценаріїв в прикладному ПЗ (наприклад, в програмах зі складу Adobe Creative Suite), всередині PDF-документів тощо.

JavaScript класифікують як прототипну (підмножина об’єктно-орієнтованої), мову програмування з динамічною типізацією. Окрім прототипної, JavaScript також частково підтримує інші парадигми програмування (імперативну та частково функціональну) і деякі відповідні архітектурні властивості, зокрема: динамічна та слабка типізація, автоматичне керування пам’яттю, наслідування, функції як об’єкти першого класу.

JavaScript є однією з найпопулярніших мов програмування в інтернеті. Але спочатку багато професіональних програмістів скептично ставилися до мови, цільова аудиторія якої складалася з програмістів-любителів. Поява AJAX змінила ситуацію та повернула увагу професійної спільноти до мови. В результаті, були розроблені та покращені багато практик використання JavaScript (зокрема, тестування та налагодження), створені бібліотеки та фреймворки, поширилося використання JavaScript поза браузером.

[1)] [7] JavaScript. URL: http://uk.wikipedia.org/wiki/JavaScript (дата звернення 18.05.2019)

JavaScript має низку властивостей об’єктно-орієнтованої мови, але завдяки концепції прототипів підтримка об’єктів в ній відрізняється від традиційних мов ООП. Крім того, JavaScript має ряд властивостей, притаманних функціональним мовам,– функції як об’єкти першого класу, об’єкти як списки, каррінг (currying), анонімні функції, замикання (closures) – що додає мові додаткову гнучкість.

JavaScript має С-подібний синтаксис, але в порівнянні з мовою Сі має такі корінні відмінності [8][1)]:

- об'єкти, з можливістю інтроспекції і динамічної зміни типу через механізм прототипів;
- функції як об'єкти першого классу;
- обробка винятків;
- автоматичне приведення типів;
- автоматичнее прибирання сміття;
- анонімні функції.

JavaScript містить декілька вбудованих об’єктів: Global, Object, Error, Function, Array, String, Boolean, Number, Math, Date, RegExp. Крім того, JavaScript містить набір вбудованих операцій, які, строго кажучи, не обов’язково є функціями або методами, а також набір вбудованих операторів, що управляють логікою виконання програм. Синтаксис JavaScript в основному відповідає синтаксису мови Java (тобто, зрештою, успадкований від C), але спрощений порівняно з ним, щоб зробити мову сценаріїв легкою для вивчення. Так, приміром, декларація змінної не містить її типу, властивості також не мають типів, а декларація функції може стояти в тексті програми після неї [9][2)].

При використанні в рамках технології DHTML JavaScript код включається в HTML-код сторінки і виконується інтерпретатором, вбудованим в

[1)] [8] Гаевский, А.Ю. Создание Web-страниц и Web-сайтов: HTML и JavaScript. М.: Триумф, 2008. 454 с.

[2)] [9] JavaScript: Онлайн підручник. URL: http://www.wisdomweb.ru/JS/javascript-first.php (дата звернення 18.05.2019)

браузер. Код JavaScript вставляється в теги <script></script> з обов'язковим по специфікації HTML 4.01 атрибутом type=»text/javascript», хоча в більшості браузерів мова сценаріїв за умовчанням саме JavaScript.

1.4 Бібліотека jQuery

jQuery – популярна JavaScript-бібліотека з відкритим сирцевим кодом [10][1)]. Вона була представлена у січні 2006 року у BarCamp NYC Джоном Ресігом (John Resig). Згідно з дослідженнями організації W3Techs, Jquery використовується понад половиною від мільйона найвідвідуваніших сайтів. jQuery є найпопулярнішою бібліотекою JavaScript, яка посилено використовується на сьогоднішній день.

jQuery є вільним програмним забезпеченням під ліцензією MIT (до вересня 2012 було подвійне ліцензування під MIT та GNU General Public License другої версії) [11][2)].

Основне завдання jQuery –це надавати розробнику легкий та гнучкий інструментарій кросбраузерної адресації DOM об'єктів за допомогою CSS та Xpath селекторів. Також даний фреймворк надає інтерфейси для Ajax-застосунків, обробників подій і простої анімації.

Принцип роботи jQuery полягає в використанні класу (функції), який при звертанні до нього повертає сам себе. Таким чином, це дозволяє будувати послідовний ланцюг методів.

1.5 Скриптова мова програмування PHP

PHP (Hypertext Preprocessor, гіпертекстовий препроцесор), попередня назва: Personal Home Page Tools –мова програмування, була створена для ге-

---

[1)] [10] jQuery. URL:http://uk.wikipedia.org/wiki/JQuery (дата звернення 18.05.2019)

[2)] [11] jQuery: Онлайн підручник. URL: http://www.wisdomweb.ru/JQ/jquery-first.php (дата звернення 18.05.2019)

нерації HTML-сторінок на стороні веб-сервера. PHP є однією з найпоширеніших мов, що використовуються у сфері веб-розробок (разом із Java, .NET, Perl, Python, Ruby) [12][1]. PHP підтримується переважною більшістю хостинг-провайдерів. PHP– проект відкритого програмного забезпечення.

PHP інтерпретується веб-сервером у HTML-код, який передається на сторону клієнта. На відміну від мови JavaScript, користувач не бачить PHP-коду, бо браузер отримує готовий html-код. Це є перевага з точки зору безпеки, але погіршує інтерактивність сторінок. Але ніщо не забороняє використовувати PHP для генерування і JavaScript-кодів які виконуються вже на стороні клієнта.

PHP – мова, код якої можна вбудовувати безпосередньо в html-код сторінок, які, у свою чергу, будуть коректно оброблені PHP-інтерпретатором.

В PHP вбудовані бібліотеки для роботи з MySQL, PostgreSQL, mSQL, Oracle, dbm, Hyperware, Informix, InterBase, Sybase. Через стандарт відкритого інтерфейсу зв'язку з базами даних (Open Database Connectivity Standard – ODBC) можна підключатися до всіх баз даних, до яких існує драйвер [13][2].

PHP 5 володіє прекрасним потенціалом реалізації об'єктного програмування. Окрім цього, PHP збагатився рядом цінних розширень для роботи з XML, різними джерелами даних, генерації графіки і інше. Серед інших украй корисних доповнень в PHP 5 слід зазначити нову схему обробки виключень. Конструкція try/catch/throw дозволяє весь код обробки помилок локалізувати в одному місці сценарію. У PHP 5 також включені два нових модулі для роботи з протоколами –SimpleXML і SOAP. SimpleXML дозволяє значно спростити роботу з XML-даними, представляючи вміст XML-документа у вигляді PHP-об'єкта. Розширення SOAP дозволяє будувати на PHP сценарії, що обмінюються інформацією з іншими застосуваннями за допомогою XML-повідомлень поверх існуючих веб-протоколів, наприклад, HTTP. Новий мо-

[1] [12] PHP: Hypertext Preprocessor. URL: http://uk.wikipedia.org/wiki/PHP (дата звернення 18.05.2019)

[2] [13] Справочник по PHP. URL: http://www.php.su (дата звернення 18.05.2019)

дуль PHP 5 MySQLi (MySQL Improved) призначений для роботи з MySQL-сервером версій 4.1.2 і вище, реалізовуючи не тільки процедурний, але і об'єктно-орієнтований інтерфейс до MySQL. Додаткові можливості цього модуля включають – SSL,контроль транзакцій, підтримка реплікації та ін. Очевидно, що, на цьому історія PHP не закінчується. Слід очікувати наступних версій мови із розширеними можливостями [14,15][1)].

1.6 Мова структурованих запитів SQL

SQL (Structured query language – мова структурованих запитів) – декларативна мова програмування для взаємодії користувача з базами даних, що застосовується для формування запитів, оновлення і керування реляційними БД, створення схеми бази даних і її модифікації, системи контролю за доступом до бази даних. Сам по собі SQL не є ні системою керування базами даних, ні окремим програмним продуктом.

SQL – це діалогова мова програмування для здійснення запиту і внесення змін до бази даних, а також управління базами даних. Багато баз даних підтримує SQL з розширеннями до стандартної мови. Ядро SQL формує командна мова, яка дозволяє здійснювати пошук, вставку, оновлення, і вилучення даних, використовуючи систему управління і адміністративні функції. SQL також включає CLI (Call Level Interface) для доступу і управління базами даних дистанційно.

SQL складається з наступних елементів мови [16][2)]:

- Положення, які є складовими компонентами заяв і запитів.

[1)] [14] Зандстра М. PHP: объекты, шаблоны и методики программирования. Издательский дом «Вильямс», 2011. 560 с.

[15] Никсон, Р. Создаем динамические веб-сайты с помощью PHP, MySQL и JavaScript. СПб.: Питер, 2013. 496 с.

[2)] [16] SQL: Structured Query Language. URL:http://uk.wikipedia.org/wiki/SQL (дата звернення 18.05.2019)

− Предикати для тризначної логіки (3VL) (так / ні / невідомо) або логічних значень, які використовуються для обмеження наслідків заяв та *запитів або змінити хід виконання програми.

− Запити для видачі скалярних величин або таблиць, що складаються із стовпців і рядків даних.

− Запити вилучення даних на основі певних критеріїв.

− Заяви впливу на схеми і даних, зв'язків.

− Заяви управлінням транзакціями, ходом виконання програми, завдань та діагностики.

Розглянемо переваги SQL:

1) Незалежність від конкретної СКБД. Незважаючи на наявність діалектів і відмінностей в синтаксисі, в більшості своїй тексти SQL-запитів, що містять, DDL і DML, можуть бути досить легко перенесені з однієї СКБД в іншу. Існують системи, розробники яких спочатку закладалися на застосування щонайменше кількох СКБД (наприклад: система електронного документообігу Documentum може працювати як з Oracle, так і з Microsoft SQL Server та IBM DB2). Природно, що при застосуванні деяких специфічних для реалізації можливостей такої переносимості добитися вже дуже важко.

2) Наявність стандартів. Наявність стандартів і набору тестів для виявлення сумісності і відповідності конкретній реалізації SQL загальноприйнятому стандарту тільки сприяє «стабілізації» мови. Правда, варто звернути увагу, що сам по собі стандарт місцями занадто формалізований і роздутий в розмірах, наприклад, Core-частину стандарту SQL:2003 включає понад 1300 сторінок тексту.

3) Декларативність. За допомогою SQL програміст описує тільки те, які дані потрібно витягнути або модифікувати. Те, яким чином це зробити, вирішує СКБД безпосередньо при обробці SQL-запиту. Проте не варто думати, що це повністю універсальний принцип – програміст описує набір даних для вибірки або модифікації, проте йому при цьому корисно уявляти, як СКБД розбиратиме текст його запиту. Особливо критичні такі моменти ста-

ють при роботі з великими базами даних і зі складними запитами – чим складніше сконструйований запит, тим більше він допускає варіантів написання, різних за швидкістю виконання, але тих самих за набором даних.

До недоліків SQL слід віднести наступне:

1) Невідповідність реляційної моделі даних. Творці реляційної моделі даних Едгар Кодд, Крістофер Дейт та їхні прихильники указують на те, що SQL не є істинно реляційною мовою. Зокрема вони указують на такі проблеми SQL:

- рядки, що повторюються;
- невизначені значення (null);
- явна вказівка порядку стовпчиків зліва направо;
- стовпці без імені та імена стовпчиків, що дублюються;
- відсутність підтримки властивості «=»;
- використання покажчиків;
- висока надлишковість.

1.7 Система керування базами даних MySQL

MySQL – вільна система керування реляційними базами даних. MySQL була розроблена компанією «ТсХ» для підвищення швидкодії обробки великих баз даних [17][1)]. Ця система керування базами даних (СКБД) з відкритим кодом була створена як альтернатива комерційним системам. MySQL з самого початку була дуже схожою на mSQL, проте з часом вона все розширювалася і зараз MySQL – одна з найпоширеніших систем керування базами даних.

MySQL має подвійне ліцензування. MySQL може розповсюджуватися відповідно до умов ліцензії GPL. Але за умовами GPL, якщо якась програма використовує бібліотеки MySQL, то вона теж повинна розповсюджуватися за

[1)] [17] Справочное руководство по MySQL. URL:http://www.mysql.ru/docs/man/ (дата звернення 18.05.2019)

ліцензією GPL. Проте це може розходитися з планами розробників, які не бажають відкривати сирцеві тексти своїх програм. Для таких випадків передбачена комерційна ліцензія компанії Oracle, яка також забезпечує якісну сервісну підтримку. В разі використання та розповсюдження програмного забезпечення з іншими вільними ліцензіями, такими як BSD, Apache, MIT та інші, MySQL дозволяє використання бібліотек MySQL за ліцензією GPL [18][1)].

Гілка MySQL 5.5 містить ряд значних поліпшень, пов'язаних з підвищенням масштабованості та швидкодії, серед яких:

- використання за замовчуванням рушія InnoDB.
- підтримка напівсинхронного (semi-synchronous) механізму реплікації, заснованого на патчах до InnoDB від компанії Google.
- поліпшення функцій з партіціювання даних. Розширений синтаксис для розбиття великих таблиць на кілька частин, розміщених в різних файлових системах (partitioning). Додані операції RANGE, LIST і метод оптимізації «partition pruning».
- новий механізм оптимізації вкладених запитів та операцій JOIN.
- перероблена система внутрішніх блокувань.
- інтегровані патчі Google з оптимізацією роботи InnoDB на CPU з великою кількістю ядер.

MySQL – компактний багатонитевий сервер баз даних. Характеризується високою швидкістю, стійкістю і простотою використання. MySQL вважається гарним рішенням для малих і середніх застосувань. Сирцеві коди сервера компілюються на багатьох платформах. Найповніше можливості сервера виявляються в UNIX-системах, де є підтримка багатонитковості, що підвищує продуктивність системи в цілому.

Можливості сервера MySQL:

- простота у встановленні та використанні;

[1)] [18] MySQL. Оптимизация производительности. СПб.: Символ-Плюс, 2010. 832 с.

- підтримується необмежена кількість користувачів, що одночасно працюють із БД;
- кількість рядків у таблицях може досягати 50 млн;
- висока швидкість виконання команд;
- наявність простої і ефективної системи безпеки.

1.8 Підхід до створення web-застосувань AJAX

AJAX (Asynchronous JavaScript And XML) – підхід до побудови користувацьких інтерфейсів веб-застосунків, за яких веб-сторінка, не перезавантажуючись, у фоновому режимі надсилає запити на сервер і сама звідти довантажує потрібні користувачу дані [19][1)]. AJAX – один з компонентів концепції DHTML.

Про AJAX заговорили після появи в лютому 2005-го року статті Джесі Джеймса Гарретта (Jesse James Garrett) «Новий підхід до веб-застосунків». AJAX – не самостійна технологія. Це ідея. На рис.2.1 наведена модель класичного додатку для мережі (зліва) в прямому порівнюванні із застосуванням Ajax (зправа).

AJAX – це не самостійна технологія, а швидше концепція використання декількох суміжних технологій. AJAX підхід до розробки, який призначений для користувачів інтерфейсів, комбінує кілька основних методів і прийомів:

- використання DHTML для динамічної зміни змісту сторінки;
- використання XMLHttpRequest для звернення до сервера «на льоту», не перезавантажуючи всю сторінку повністю;

[1)] [19] AJAX: Asynchronous Javascript and XML. URL: http://uk.wikipedia.org/wiki/AJAX (дата звернення 18.05.2019)

- альтернативний метод – динамічне підвантаження коду JavaScript в тег <SCRIPT> з використанням DOM, що здійснюється із використанням формату JSON);
- динамічне створення дочірніх фреймів.

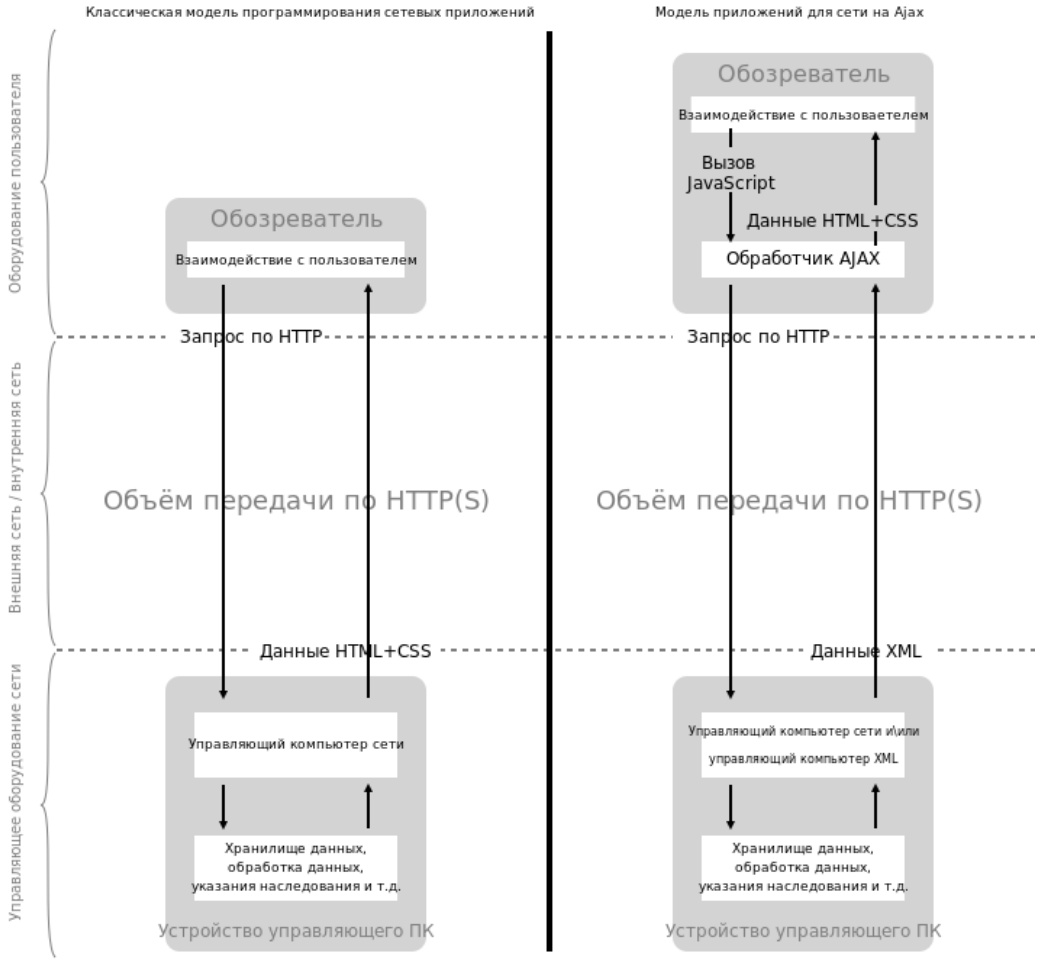

Рисунок 2.1 – Схема роботи AJAX

AJAX – асинхронний, тому користувач може переглядати далі контент сайту, поки сервер все ще обробляє запит. Браузер не перезавантажує web-сторінку і дані посилаються на сервер без візуального підтвердження (крім випадків, коли ми самі захочемо показати процес з’єднання з сервером).

Порівняння класичного підходу та AJAX. Класична модель веб-застосування:

- користувач заходить на веб-сторінку і натискає на який-небудь її елемент;
- браузер надсилає запит серверу;
- у відповідь сервер генерує повністю нову веб-сторінку і відправляє її браузеру і т.д. З боку сервера можлива генерація не всієї сторінки наново, а тільки деяких її частин, з подальшою передачею користувачу.

Модель AJAX:

- користувач заходить на веб-сторінку і натискає на який-небудь її елемент;
- браузер відправляє відповідний запит на сервер;
- сервер віддає тільки ту частину документа, яка змінилася.

Переваги:

- економія трафіку;
- зменшення навантаження на сервер;
- прискорення реакції інтерфейсу;
- майже безмежні можливості для інтерактивної обробки. Наприклад, при введенні пошукового запиту в Google виводиться підказка з можливими варіантами запиту.

Недоліки:

- відсутність інтеграції зі стандартними інструментами браузера;
- зміни вмісту сторінки при постійному URL полягає в неможливості збереження закладки на бажаний матеріал;
- відсутня можливість динамічно завантажувати вміст недоступний пошуковикам (якщо не перевіряти запит, звичайний він або XMLHttpRequest). Пошукові машини не можуть виконувати JavaScript, тому розробники мають подбати про альтернативні способи доступу до вмісту сайту;

- старі методи обліку статистики сайтів стають неактуальними. Багато сервісів статистики ведуть облік переглядів нових сторінок сайту. Для сайтів, сторінки яких широко використовують AJAX, така статистика втрачає актуальність;
- ускладнення проекту. Перерозподіляється логіка обробки даних – відбувається виділення і часткове перенесення на сторону клієнта процесів первинного форматування даних. Це ускладнює контроль цілісності форматів і типів;
- потрібно включений JavaScript в браузері;
- низька швидкість при грубому програмуванні;
- ризик фабрикації запитів іншими сайтами. Для захисту використовують POST-запит. Але GET вважається ідемпотентність і тому кешируєтся, POST – ні, тому Google вставляє в початок відповіді нескінченний цикл: AJAX може робити з відповіддю що завгодно, у тому числі прибрати цикл, а тег <script> підключить скрипт як є і зациклиться.

1.9 Архітектура проекту

Model-view-controller (Model-view-controller, MVC) – архітектурний шаблон (рис.1.2), який використовується під час проектування та розробки програмного забезпечення.

Цей шаблон поділяє систему на три частини: модель даних, вигляд даних та керування. Застосовується для відокремлення даних (модель) від інтерфейсу користувача (вигляду) так, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними, а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача.

Мета шаблону – гнучкий дизайн програмного забезпечення, який повинен полегшувати подальші зміни чи розширення програм, а також надавати можливість повторного використання окремих компонентів про-

грами. Крім того використання цього шаблону у великих системах призводить до певної впорядкованості їх структури і робить їх зрозумілішими завдяки зменшенню складності.

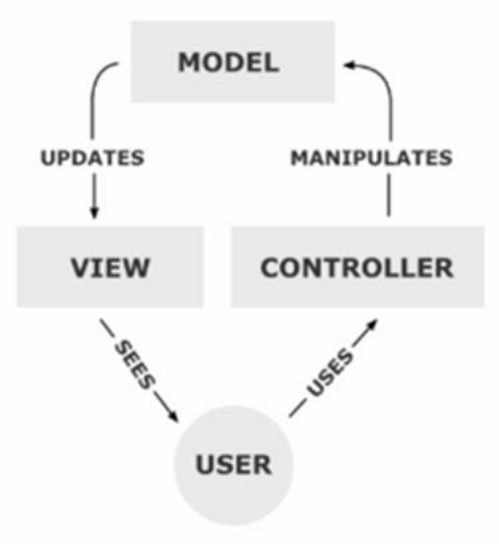

Рисунок 1.2 – Шаблон проектування MVC

Архітектурний шаблон Модель-Вид-Контролер (MVC) поділяє програму на три частини. У тріаді до обов'язків компоненту Модель (Model) входить зберігання даних і забезпечення інтерфейсу до них. Вигляд (View) відповідальний за представлення цих даних користувачеві. Контролер (Controller) керує компонентами, отримує сигнали у вигляді реакції на дії користувача, і повідомляє про зміни компоненту Модель. Така внутрішня структура в цілому поділяє систему на самостійні частини і розподіляє відповідальність між різними компонентами.

MVC поділяє цю частину системи на три самостійні частини: введення даних, компонент обробки даних і виведення інформації. Модель, як вже було відмічено, інкапсулює ядро даних і основний функціонал з їх обробки. Також компонент Модель не залежить від процесу введення або виведення даних. Компонент виводу Вигляд може мати декілька взаємопов'язаних областей, наприклад, різні таблиці і поля форм, в яких

відображається інформація. У функції Контролера входить моніторинг за подіями, що виникають в результаті дій користувача (зміна положення курсора миші, натиснення кнопки або введення даних в текстове поле).

Зареєстровані події транслюються в різні запити, що спрямовуються компонентам Моделі або об'єктам, відповідальним за відображення даних. Відокремлення моделі від вигляду даних дозволяє незалежно використовувати різні компоненти для відображення інформації. Таким чином, якщо користувач через Контролер внесе зміни до Моделі даних, то інформація, подана одним або декількома візуальними компонентами, буде автоматично відкоригована відповідно до змін, що відбулися.

Концепція MVC вперше застосувалася при проектуванні мови програмування Smalltalk як модель для інтерфейсу користувача. Також в область застосування концепції входить реалізація каркаса Документ-Вид (Document-View) в рамках бібліотеки MFC для мови Visual C++. У сучасних технологіях концепція MVC представлена схемою JSP Model 1/2 для динамічної обробки Web-змісту на основі Java Server Pages (JSP) [20][1)].

2 ОПИС ЕТАПІВ ВЕРСТАННЯ МАКЕТУ

Після того, як дизайнер приготував макети сайту (рис. 2.1) – завдання верстальника перенести ці макети в гіпертекстові документи.

Спочатку розіб’ємо макет на кілька сегментів: шапка сайту (Header), ліва колонка (Left bar), права колонка (Right bar), контент (Content) і підвал (Footter).

В умовах дизайну змінюватися динамічно на сайті буде тільки один компонент: контент – з чого випливає, що кожну сторінці контенту необхідно верстати окремо, не включаючи статичні елементи дизайну сайту.

[1)] [20] MVC: Model View Controller. URL: https://uk.wikipedia.org/wiki/Model-View-Controller (дата звернення 18.05.2019)

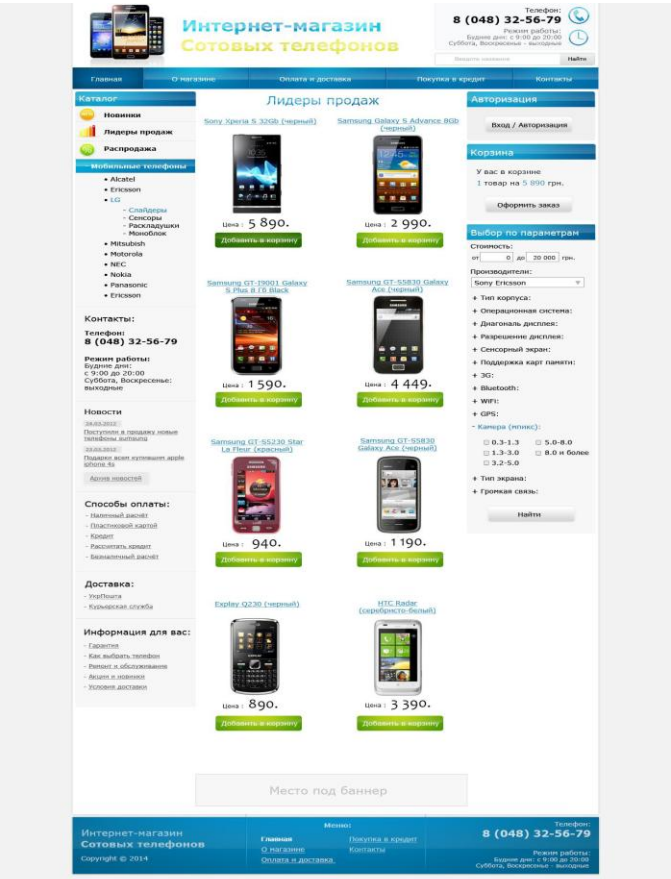

Рисунок 2.1 – Макет сторінки додатку

2.1 Шапка (Header)

Починати верстку будемо з шапки (рис. 2.2). Розіб'ємо шапку умовно на два блоки:

1) Блок з інформацією;
2) Панель керування.

Рисунок 2.2 – Шапка додатку

Починаючи розмітку блоку з інформацією, ми розбиваємо його на внутрішні блоки (рис.2.3).

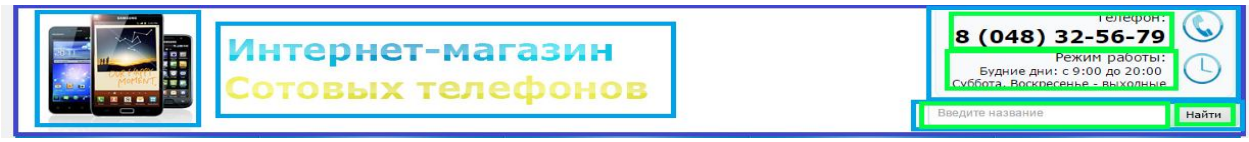

Рисунок 2.3 – Розмітка фрагментів шапки

Всього таких блоків буде 4. Блок з картинкою, блок з титулом магазину, контактна інформація і блок з формою пошуку.

Блок з контактною інформацією бует містити два параграфа, а блок з формою пошуку містить саме поле і кнопку для відправки запиту.

Після завершення не складної розмітки в HTML ми прописуємо стилі для даних блоків у файлі стилів CSS. Картинки позиціонуємо згідно їх розміщення на макеті. Текстовим даними задаємо шрифти, розміри. Також за допомогою CSS виводимо іконки, що відображають типи наданої інформації. Самі іконки виводимо як задній план параграфів зі зміщенням, щоб текст не налазив на них.

Формі пошуку задаємо розмір, зовнішній вигляд і позиціонуємо праворуч від неї картинку, що буде виконувати функцію кнопки.

Тепер переходимо до панелі управління. Вона являє собою звичайний список з посиланнями (рис. 2.4).

Рисунок 2.4 – Основне меню

Спочатку кожному елементу списку задаються розміри і фоновий колір. Після прописуємо обтікання, щоб всі елементи списку стали в рядок, як зазначено на макеті. Усередині елементів списку є посилання і параграф.

Для параграфів ми задаємо необхідні шрифти, розміри і колір тексту. Прописуємо відступи всередині елементів, для позиціонування тексту рівно по центру кожного блоку.

2.2 Основна частина

Далі нашої задачею є первинна розмітка і верстка 3-х блоків: лівої колонки, контенту і правої колонки. (рис. 2.5)

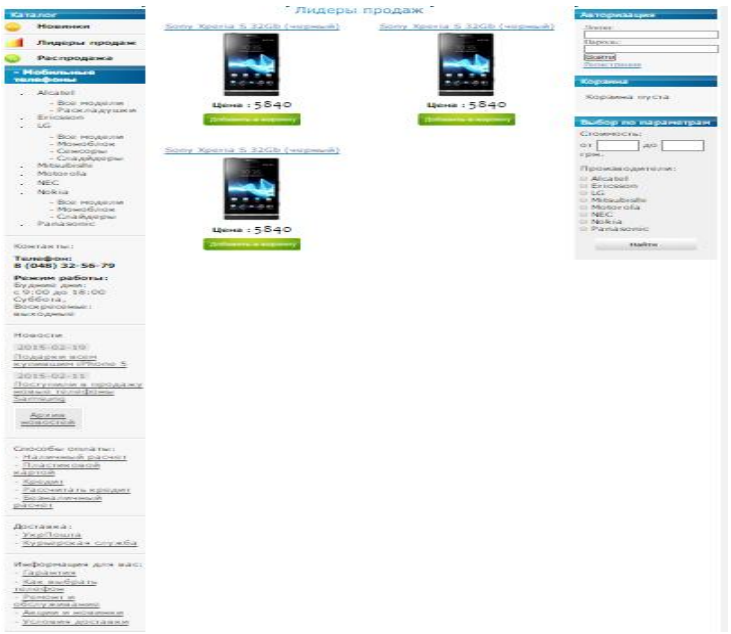

Рисунок 2.5 – Основна частина

У HTML-розмітці розміщуємо блоки в такій послідовності:

– обгортка;
– ліва колонка;
– права колонка;
– контент.

Після розміщення правого і лівого блоків, блок контенту буде якби провалюватися по центру між ними. У стилях правого і лівого блоків встановлюємо фіксовану ширину. Висоту не задаємо, його висота буде встановлюватися залежно з кількістю вмісту. Ті ж операції ми робимо з блоком контенту, який надалі буде виконувати роль обгортки, для динамічно виведеного вмісту.

2.3 Права колонка

Спочатку зробимо саму невелику колонку – праву (рис. 2.6). У нього входить всього 3 блоку – це блок авторизації, корзина і сортування за параметрами.

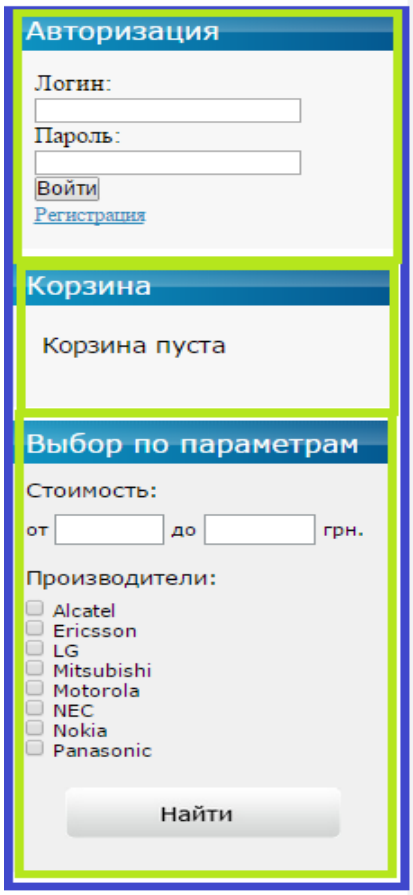

Рисунок 2.6 – Права колонка

Кожен з блоків володіє заголовком, для якого ми підбираємо фон, шрифт і позиціонування.

Блок авторизації містить форму, що відправляє дані. А в разі успішної авторизації буде просто виводити ім'я зареєстрованих користувачів. Стилями ми просто вирівнюємо поля форми і кнопку, а для параграфа виводить ім'я – задаємо розмір і стиль шрифтів.

2.4 Ліва колонка

Перейдемо тепер до лівого блоку. Він містить досить велику кількість блоків.

Отже, лівий блок містить каталог товарів (рис.2.7). Як за загальними категоріями, так і по конкретним. Так само в нього входять контакти і інформери для відвідувачів.

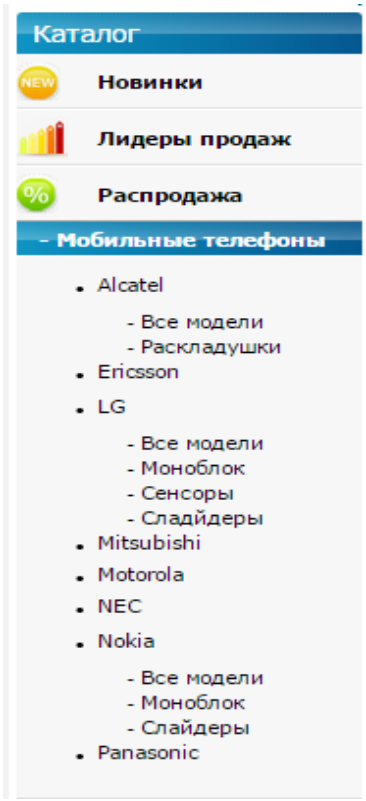

Рисунок 2.7– Ліва колонка. Каталог товарів

Заголовки блоків ми стилізуємо також, як і в правій колонці. Блок з вибором конкретної марки телефону створюється списком категорій.

Далі йде блок контактів (рис. 2.8). Тут в розмітці декілька параграфів, в яких не складними маніпуляціями ми виділяємо ключові відомості.

Потім слідують блоки-інформери (рис. 2.9). Їх три, але стилі та розмітка у них однакові: заголовок і список.

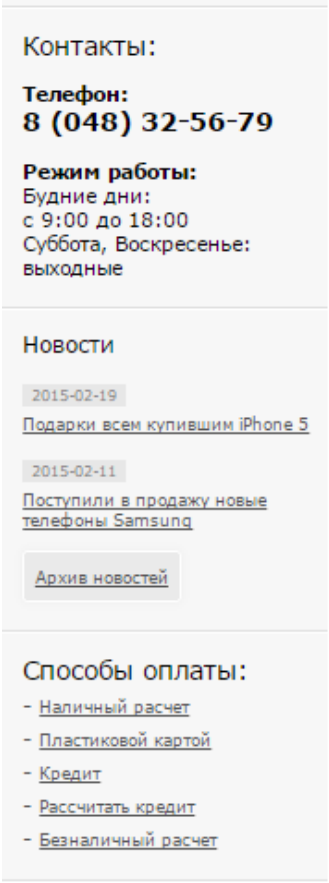

Рисунок 2.8 – Ліва колонка. Блок контактів і новин

2.5 Підвал (footer)

Розмітка підвалу (рис. 2.10) приблизно ідентична шапці сайту. Весь футер, зверстаний за принципом основної частини сайту. Тобто, спершу йдуть блоки логотипу магазину, потім блок з контактною інформацією. Вони розміщуються з боків основного блоку-обгортки, а блок з виведенням меню

як би провалюється між ними, займаючи всю ширину порожнього простору. Нам залишається спозиціонувати текст точно по центру і задати стилі відображення тексту, що робиться вже не одноразово.

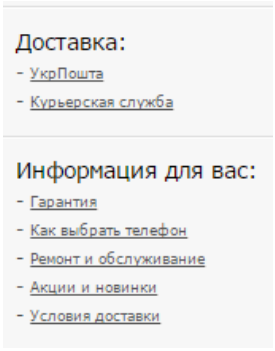

Рисунок 2.9 – Ліва колонка. Блоки-інформери

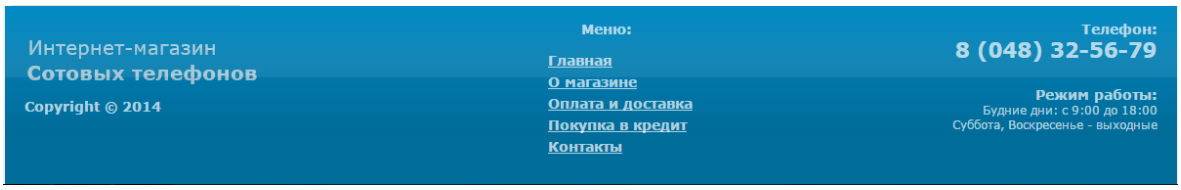

Рисунок 2.10 – Підвал

На цьому завершується верстка всіх статичних частин сайту. Від сторінки до сторінки їх зовнішній вигляд і наявність змінюватися не будуть.

2.6 Каталог товарів

Три основні пункти каталогу (рис. 2.11) нічим відрізнятися один від одного не будуть, за винятком типу вмісту. Тому верстка у них ідентична.

Усередині блоку товару мається найменування продукту посиланням, картинка самого товару, параграф з ціною і кнопка у вигляді картинки, для додавання товару в корзину.

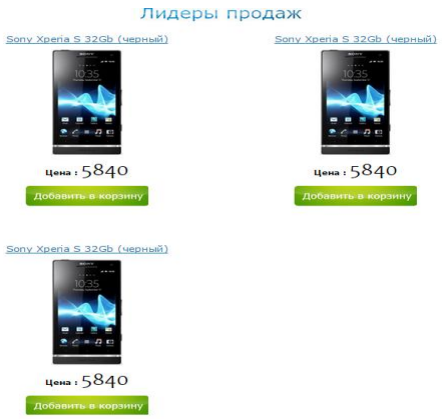

Рисунок 2.11 – Каталог товарів

Каталог товарів з конкретних категорій буде відрізнятися можливістю виведення товарів сіткою і списком. А також наявністю опцій сортування і, можливо, посторінкового навігації.

«Лінійний» вивід товарів (рис. 2.12) представлений дещо складніше. Крім основних пунктів виведення: назви, картинки, ціни, кнопки і можливих іконок, він містить короткий опис товару. Виводитися це вже застосовуваним раніше методом розмітки. З початку картинка, потім блок з ціною, кнопкою, іконками та потім опис. Після позиціонування текст, провалиться між розміщеними блоками. Це робиться для забезпечення гнучкості сайту. Адже завдання – верстати його «гумовим».

Рисунок 2.12 – Каталог товарів (лінійний вид)

2.7 Кошик

Приступимо до верстки кошика (рис. 2.13). У даному випадку поєднується техніка табличної і блокової верстки.

В цілому вся корзина являє собою форму. Для початку креслимо таблицю, комірки якої будуть містити такі дані:

– картинка товару;
– найменування;
– поле з кількістю товару, з можливістю зміни користувачем;
– окрема вартість товару;
– кнопка видалення товару з кошика, у вигляді зображення.

Останній рядок таблиці буде виводити підсумкові кількість і суму товарів.

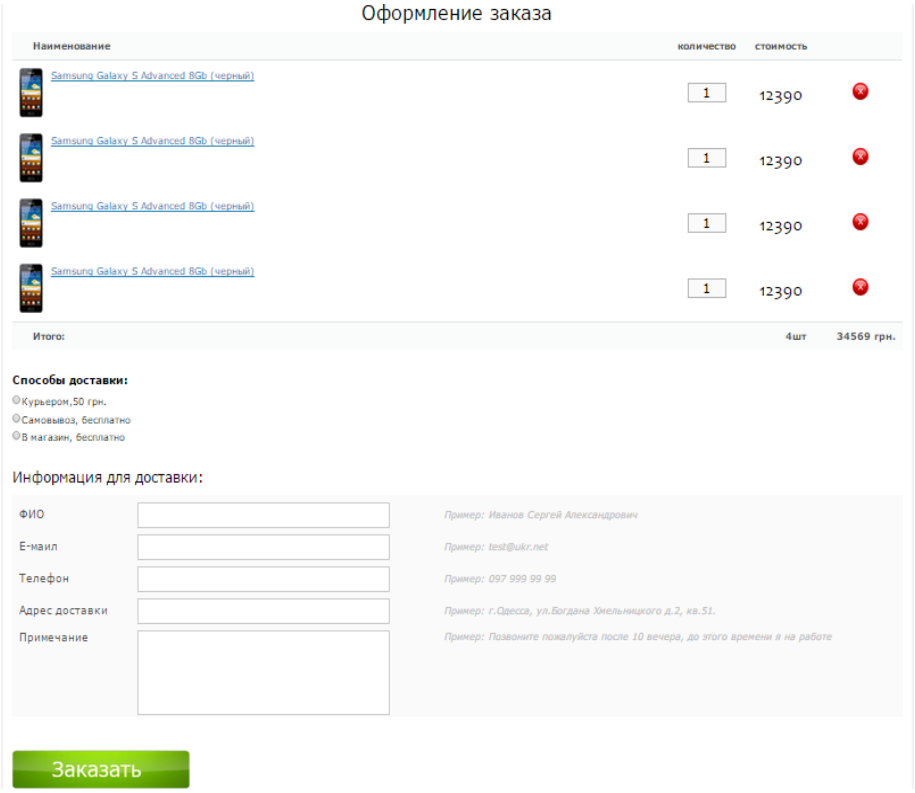

Рисунок 2.13 – Кошик (оформлення замовлення)

Після цього встановлюються радіо кнопки з можливістю вибору доставки. Після поля для введення контактних даних. Це розраховано для незареєстрованих користувачів, зареєстрованим виводитиметься лише поле для приміток.

2.8 Детальний перегляд товару

Останнім пунктом верстки буде детальний перегляд товару (рис. 2.14). Макетом передбачена наявність блоку зі слайдером і обтікаючого його по лівому краю блоку з коротким описом товару, його ціною, іконками і кнопкою замовлення.

Під цими блоками будуть розташовані параграфи і списки. В яких буде зберігатися вже детальний опис товару і його технічні характеристики.

Рисунок 2.14 – Детальний перегляд товару

3 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНО-ПОШУКОВОЇ СИСТЕМИ

3.1 Структура бази даних

База даних додатку «Інтернет-магазину» складається з десяти таблиць. Структура бази даних наведена на рис. 3.1. База даних створюється з кодуванням за замовчуванням UTF-8 і способом порівняння utf8_general_ci.

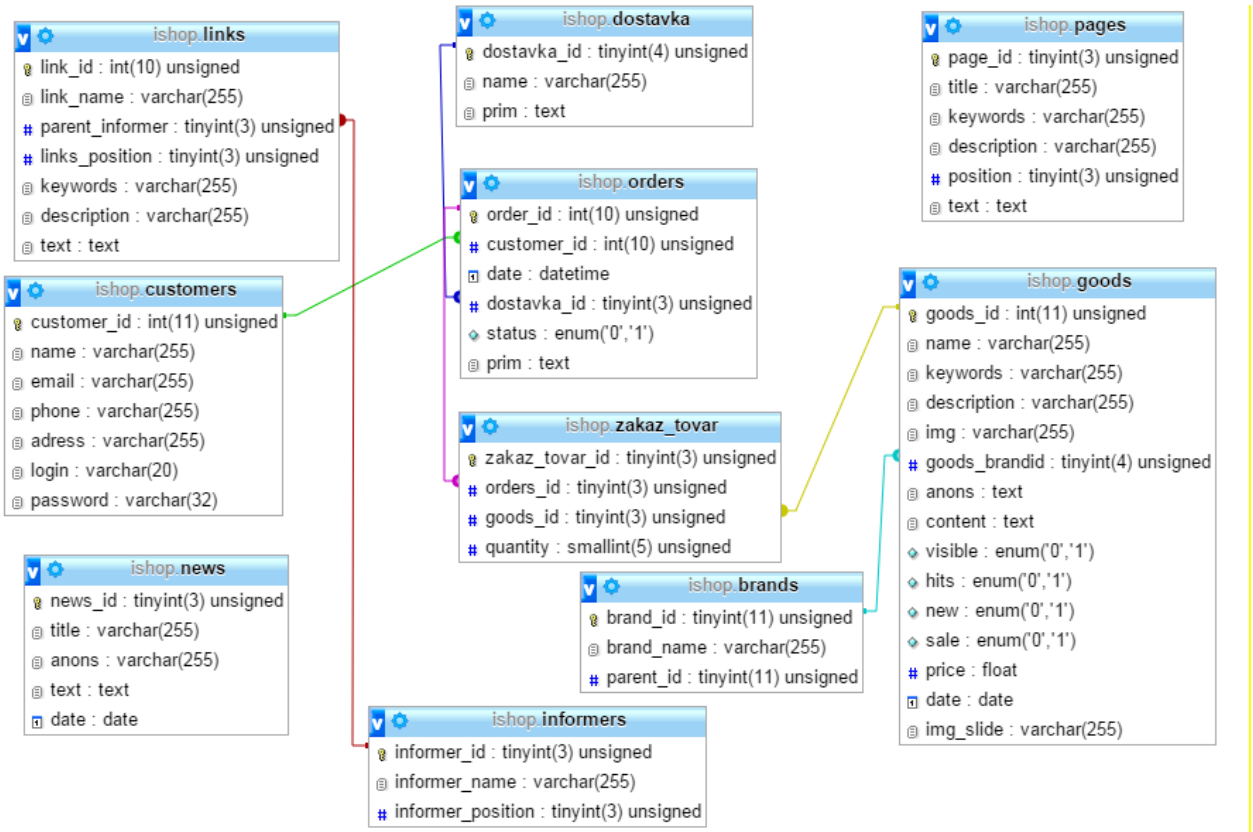

Рисунок 3.1 – Структура бази даних

Кожна таблиця БД відповідає за наступні сутності:

- Brands – в даній таблиці буде зберігатися код бренду, його назву і посилання на батьківський ідентифікатор. Сама таблиця буде містити найменування як категорій так і їх підкатегорій, таким чи-

ном моделі телефонів бренду сумісного можуть мати підкатегорії у вигляді: слайдер, моноблок і т.д.

- Customers – таблиця клієнтів містить, відповідно: ідентифікатор клієнта, ім'я, емайл, адреса, логін і пароль.
- Dostavka – невелика таблиця, яка зберігає ідентифікатор і назву пропонованих способів доставки.
- Goods –товари. Ідентифікатор, ім'я, ключові слова, опис, картинка, анонс, основний текст опису товару, бачимо товар чи ні, чи відноситься він до пропозицій типу: «новинки», «розпродаж», «хіти», ціна, дата та зображення для слайдера.
- Informers – інформери. Ідентифікатор інформера, його ім'я і порядковий номер.
- Links – посилання. Ідентифікатор, ім'я, ідентифікатор батьківського інформера, позиція, ключові слова, опис і основний текст.
- News – новини. Ідентифікатор, заголовок, анонс, текст і дата.
- Orders – замовлення. Ідентифікатор, ідентифікатор замовника, дата, ід. поставки, статус, текст примітки.
- Pages – сторінки. Дана таблиця містить інформацію з основного меню, розташованого в шапці додатка. Ідентифікатор, заголовок, ключові, опис, позиція і основний текст.
- Zakaz_tovar – замовлення товарів. Таблиця призначена для зберігання інформації відразу про безліч замовлень. Складається з ідентифікатора, ід. самого замовлення, ід. товару і кількості.

3.2 Шаблон проектування MVC

Для побудови програми була обрана архітектура MVC – модель, вид, контролер (рис. 3.2). В умовах даної архітектури проект складається з:

- Модель (Model) – файл який працює безпосередньо з базою даних проекту. Містить всі пов'язані з нею функції.

– Вид (View) – каталог містить вид проекту, або іншими словами – тему. Тут розташовані всі зверстані раніше сторінки (рис. 3.3).

– Контролер (Controller) – файл обробник. Контролер буде отримувати, обробляти і повертати користувачеві запитану інформацію. Його функціональність відповідає за виведення інформації в динамічних частинах проекту.

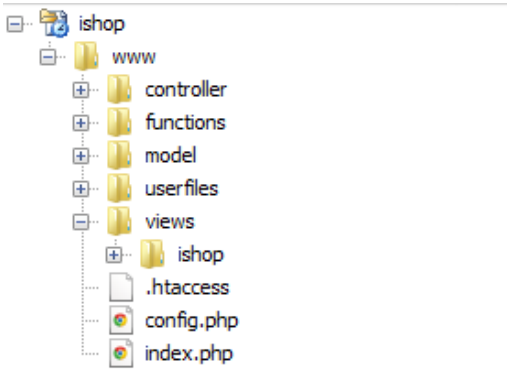

Рисунок 3.2 – Перенесення архітектури MVC у структуру програми

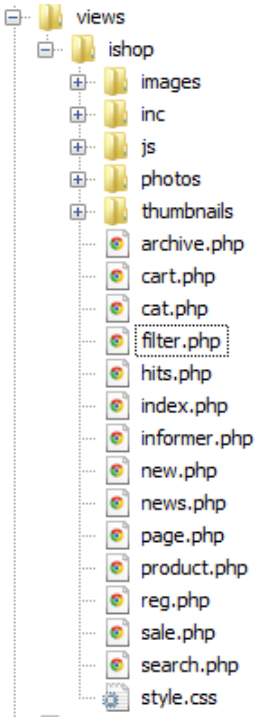

Рисунок 3.3 – Каталог «VIEW» (Вид)

Каталог functions – містить файл або файли, які виконують функції не пов'язані з базою даних. Такі як на приклад: вихід користувача, робота з кошиком і т.п.

Каталог userfiles – містить користувача файли, які не відносяться безпосередньо до виду проекту і можуть мінятися динамічно.

Файл config – являє собою файл конфігурації. Містить в собі набори констант, а також підключення до бази даних.

Index.php – файл, який за замовчуванням виводить сервер, при запиті головної сторінки сайту. У нього підключається контролер і конфігурація.

.htaccess – файл для налаштувань сервера Apache. У ньому задаються такі параметри як кодування за замовчуванням. Встановлюються файли, виведені в разі помилки, таких як помилка 404 і т.п.

Через index.php в кореневій папці підключається файл конфігурації і контролер. Контролер підключає використовуваний вид і модель. Так само в подальшому контролер містить встановлювані змінні і виклики функцій.

У наявному шаблоні сторінка містить як динамічні так і статичні елементи. Динамічним, по суті, в ній є блок «контенту». Всі інші, а саме: шапка, ліва колонка, права колонка і підвал – статичні. Це означає, що вони повинні завжди залишатися перед очима у користувача. Шаблон розбивається на зазначені елементи, які, для зручності, виводяться в окремий каталог. Створюється індексовий файл, який буде підключати статично виведені в окремі файли статичні частини. Його динамічна складова буде змінюватися в залежності від переданого контролеру параметра.

У випадку зі статичними елементами сторінки використовується функція require_once, для того, щоб в разі не завантаження якого–небудь файлу припинялося всі виконання коду. Динамічний елемент підключається за допомогою функції include, вона в свою чергу виведе нас або підключається файл або помилку, не перешкоджаючи виконанню всього іншого коли. Таким чином статична частина сайту, в якій зосереджено управління, буде доступне відвідувачеві.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНО-ПОШУКОВОЇ СИСТЕМИ

4.1 Реєстрація користувачів

Для прив'язування покупця до свого ресурсу доцільно пропонувати йому зареєструватися (рис. 4.1). Це позбавить його необхідності постійно вводити свої контактні дані при оформлені замовлення. Якщо користувач виявив бажання зареєструватися ми відправляємо його до форми реєстрації, де йому належить ввести обов'язкові поля із даними про себе.

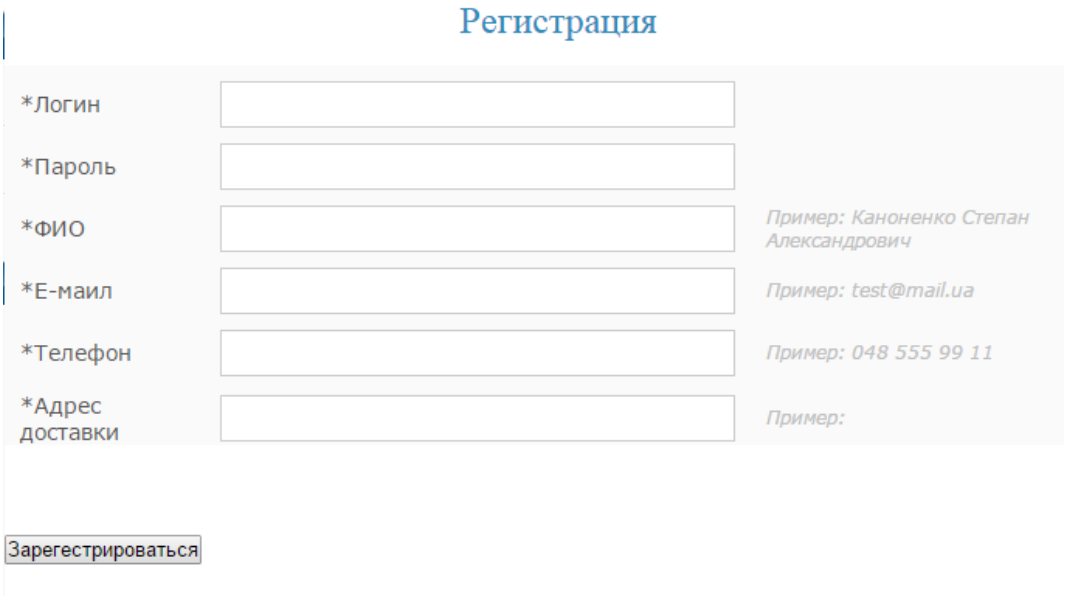

Рисунок 4.1 – Форма реєстрації

Функція реєстрації на пряму працює із базою даних. Вносячи в неї інформацію передану користувачем. З чого слідує, що описується вона в моделі додатку.

Наступним кроком є перевірка наповненості полів, за умовами шаблону усі поля форми реєстрації обов'язкові. У разі незаповненості якогось поля під формою реєстрації виводимо повідомлення з переліком помилок (рис. 4.2).

Рисунок 4.2 – Помилка при реєстрації

В разі позитивного результату виконуємо запит до бази даних, який перевірятиме чи вже є такий користувач. Запит вибирає ідентифікатор користувача з таблиці користувачів, по логіну який надав користувач при реєстрації обмежено одним записом. За допомогою функції mysql_num_rows у змінну $row отримуємо кількість зачеплених запитом строчок, в нашому випадку це буде або 1, або 0.

Виконуємо перевірку і якщо робота функції повернула 1, тобто такий користувач вже є, то виводимо повідомлення із помилкою, при цьому залишаючи введені користувачем данні у полях за допомогою сесійного масиву. Це доцільно робити, щоб користувачу не доводилось постійно вводити данні при невдалих реєстраціях.

В іншому випадку спочатку кодуємо пароль завдяки функції md5. Всі передані данні записуються в таблицю із користувачами. За допомогою функції mysql_affected_rows перевіряємо чи відбулися зміни у бузі даних під час виконаня останнього запиту, якщо так то виводимо повідомлення про успішну реєстрацію та сесійний масив заносимо данні передані при реєстрації, чим автоматично авторизуємо користувача. В разі коли функція не поверне бажаного результату, виводимо помилку. Також, якщо не всі поля заповнені виводимо повідомлення із списком помилок

4.2 Авторизація користувачів

Логічно, що наступним кроком є реалізація авторизації користувачів. Форма авторизації (рис. 4.3) за шаблоном розміщується у правій колонці додатку. Має поля для вводу логіну та паролю.

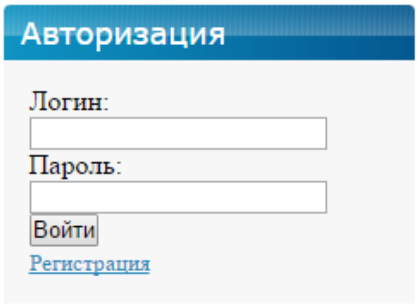

Рисунок 4.3 – Форма авторизації

Функція авторизації на пряму працює із базою данних. По-перше обробляються данні отримані скриптом від користувача. Функція mysql_real_escape_string дозволяє уникнути використання спецсимволів у коді, автоматично замінюючи їх на HTML сутності. Також отримані в полі логіну та паролю данні оброблюються функцією trim, що видаляє пробіли по обидві сторони переданих даних. Перевіряємо чи заповнені змінні логіну та паролю. Якщо ні, то сесійному масиву [auth][error] привласнюється текст помилки. Якщож все добро і дані з полів отримані, то закодуємо змінну, що містить пароль за допомогою функції md5. Складається запит до бази даних із використанням змінних, що містять введені користувачем логін та пароль.

З таблиці клієнтів вибирається ідентифікатор, ім’я та E-маіл, де поля логіну та паролю співпадають із введеними користувачем, кількість вибраних записів лімітується одним.

За допомогою функції mysql_num_rows перевіряємо чи дорівнює кількість відданих на запит строчок 1. Якщо так, то оброблюємо результат функ-

цією mysql_fetch_row, що надасть індексовий масив із вибраними даними. Далі привласнюємо отримані данні сессіонному масиву. У разі невдачі ж – елементу [error] сесійного масиву привласнюється відповідна строка із текстом помилки.

Так, якщо не існує елементу [user] в сесійному масиві, то буде виводитись форма авторизації, також якщо в масиві встановлені дані в елементі масиву відповідаючому за зміст помилок – блок, що виводить текст помилки створюється та відображається користувачу (рис.4.4). Після цього елемент знищується, щоб помилка не трималась на місці після перезавантаження сторінки.

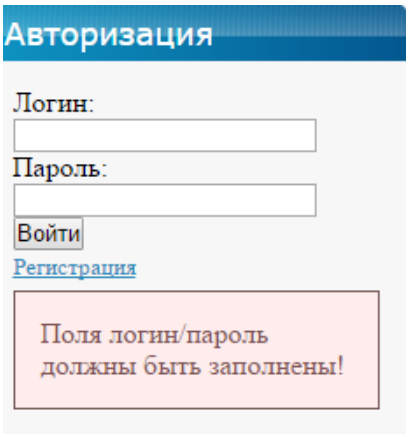

Рисунок 4.4 – Помилка при авторизації

Якщож авторизація пройшла успішно і в масиві сесії є елемент [auth][user], то виводиться повідомлення із привітанням, та посилання на вихід з облікового запису.

4.3 Модуль кошику

Кошик – одна з найважливіших частин інтернет-магазину. Він містить в собі інформацію щодо вибраних користувачем товарів, власне його функції такі самі як і у кошика у звичайному фізичному магазині.

Суть роботи кошику полягає у наступному: користувач, переглянувши товари, натискає на кнопку відносну до цього товару і відправляє на контроллер данні, які містять в собі виклик необхідної функції та параметрів самого товару. Функція створює сессійний массив, який тримає в собі всі основні данні товару.

Модуль кошику міститься у правій колонці додатку. У середині даного блоку розміщується умовний оператор, який в залежності від наявності замовлених товарів виводить: або загальну кількість та сумму товарів, або повідомлення про порожність кошику.

Функція додавання у кошик прийматиме два параметри – це ідентифікатор товару, та кількість. Оскільки дизайном не передбачено додавання декількох товарів за раз другий параметр за замовчанням буде визначений.

Одже, по перше функція перевіряє чи наявний у массиві сессіі елемент із ідентифікатором товару і якщо він там є – то його кількість збільшується на кількість, що передається (в даному випадку кількість вже є константою). Повертає функція масив сессіі з елементом кошику. В іншому випадку в масиві сесії з елементом кошику створюється вкладений масив із передаваємим ідентифікатором товару і привласнюється зазначена кількість товарів. Знову функція повертає масив сесії з елементом кошику.

Щоб функція виконалась її треба викликати в контролері, в тому випадку якщо буде відбуватися додавання товару.

Отримуючи будь-які данні від користувача необхідно їх оброблювати, задля достовірності у типі данних, що приймає скрипт і також за для безпеки від можливих недобрих намірів.

Отже, спершу переданий параметр приводиться до явного числового типу. Функцією abs ми виключаємо можливість отримання не цілого числа. Далі викликається функція додавання товару у кошик, якій передається ідентифікатор товару. В спеціальний сесійний масив привласнюється результат роботи функції, яка підраховує загальну сумму товарів у кошику. В якості параметра цій функції передається саме елемент кошику масиву сесії.

В запиті до бази данних вибираються ідентифікатор, ім’я, ціна да зображення з таблиці товарів. В якості умови і використовується отриманий діапазон ідентифікаторів. Наступним кроком є вивід загальної кількості товарів у кошику.

Буває що користувач додає товар випадково або передумав його купляти, тоді необхідно надати йому можливість видаляти конкретний товар з кошику.

Функція видалення приймає ідентифікатор товару, що користувач бажає вилучити. І поперше вона перевіряє чи взагалі є щось у кошику користувача. Якщо це так, то знов таки функція перевіряє чи наявний переданий нам ідентифікатор у кошику. Для цього використовується функція array_key_exists, яка перевіряє наявність заданих ключів у заданому масиві. І знову, якщо ці ключі наявні з масиву загальної кількості товарів видаляються всі товари із вказаним ідентифікатором, а від загальної сумми віднимається кількість таких товарів помножена на вартість одного з них. Сам товар – видаляється просто завдяки команді unset.

4.3 Оформлення замовлення

Після того як товар було додано у кошик користувач може оформити замовлення, для цього йому пропонується перейти до форми замовлення через натискання відповідної кнопки (рис. 4.4).

У формі замовлення (рис. 4.5) користувачу пропонується перегляд товарів доданих ним у кошик, можливість їх видалення з кошику та загальне оформлення замовлення. Дана форма містить в собі основні властивості товарів, засоби доставки та дані користувача, якщо він авторизований. Користувачам не авторизованим пропонується ввести данні в обов’язкові поля.

Першочергово видно, що за умовами дизайну кількість товарів у кошику може підлягати редагуванню. Тобто якщо користувач захотів купити не

одну одиницю товару, а декілька він має ввести бажану цифру в результаті чого кількість товарів та загальна ціна буде перерахована.

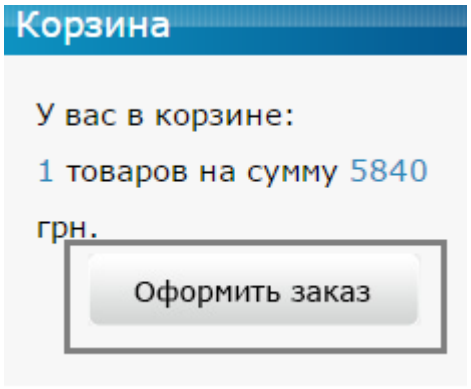

Рисунок 4.4 – Зовнішній модуль кошику

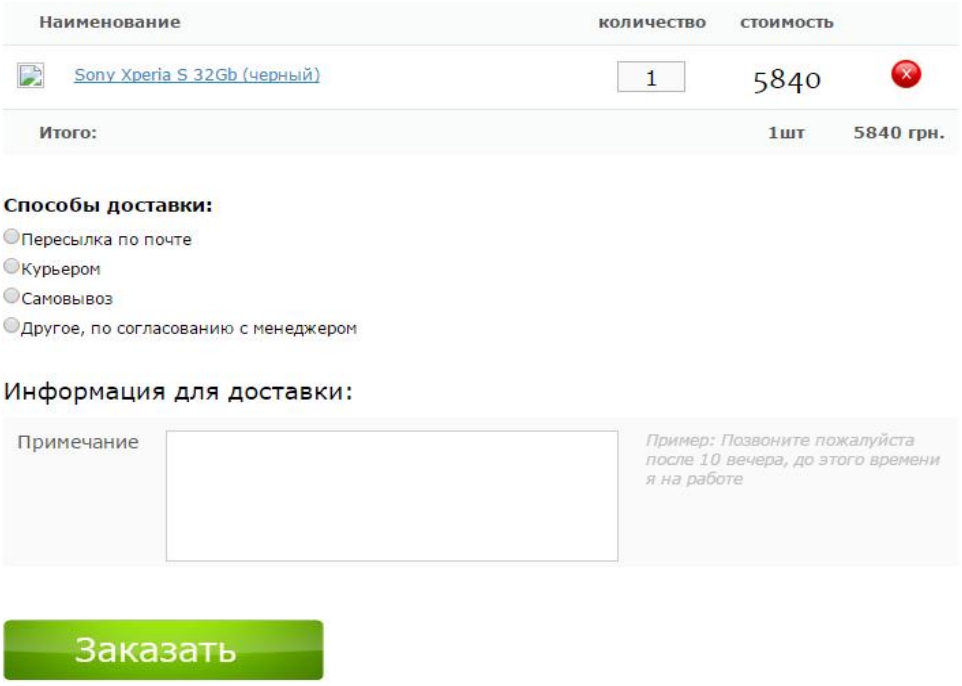

Рисунок 4.5 – Оформлення замовлення

Також для форми всі варіанти доставки описані в базі даних, тож невелика функція, що викликається в контролері, покликана надати список цих

методів. Результатом її роботи буде список засобів доставки. Які представляються у формі замовлення як параметри (рис. 4.6).

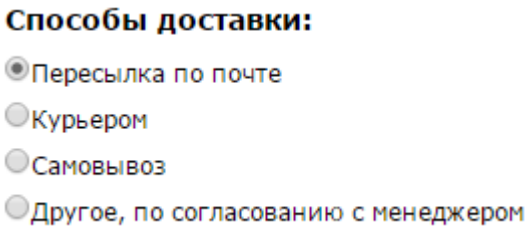

Рисунок 4.6 – Варіанти доставки

Форма також передбачає можливість видалення товарів з кошику. Так при натисканні кнопки видалення контролеру передається ідентифікатор конкретного товару, який потрібно видалити. Запускається функція видалення.

Якщож все заповнено і користувач бажає оформити замовлення він натискає кнопку «замовити» (рис. 4.7). Разом з цим запускається функція, яка додає замовлення до бази даних.

Рисунок 4.7 – Кнопка «Замовити»

Оскільки замовляти товари може як авторизований користувач так і гість, загальні дані ми отримуємо одразу для обох випадків. Отримується ідентифікатор доставки, якщо ідентифікатор не передано, то йому присвоюється цифра «1» за замовчанням. Поле з примітками замовника фільтрується, за для запобігання потраплянню на сервер різного роду «ін’єкцій». Перевіряємо чи зареєстрований користувач, якщо так то ідентифікатору замовника

присвоюється значення ідентифікатора користувача. Далі, якщо користувач не авторизований або гість, проводиться перевірка на наповненість всіх обов'язкових полів. В разі незаповненості – виводиться помилка із вказуванням конкретного поля. У разі відсутності помилок додаємо гостя за допомогою функції add_customer, в яку передаються всі отримані від нього парамтри.

Складається запит, який додає в таблицю замовлень ідентифікатор замовника, дату замовлення, ідентифікатор типу доставки та примічання.

Наступним кроком перевіряється наявність елементу Е-маіл в сесійому масиві авторизованого користувача, в позитивному випадку одноіменній змінній присвоюється його значення. В противному випадку присвоюється значення із елементу Е-маіл замовлення. Відправляється поштове повідомлення.

Функція mail відправляє два листи із вказаними параметрами : один клієнту, другий адміністратору. Після виконання цих операцій виводиться повідомленя про успішну обробку замовлення.

4.4 Функція посторінкової навігації

Посторінкова навігація допомагає відображати велику кількість контенту так, щоб на одній сторінці розміщувалось лише встановлена його кількість. Для реалізації посторінкової навігації запит до бази даних вже побудований, а в контролері для відповідного кейсу необхідно задати параметри.

Визначається кількість контенту на сторінку. Відповідна ункція складається із двох об'єднаних запитів. Перший підраховує кількість ідентифікаторів товарів де ідентифікатор бренду є передаваємим параметром та товар є видимим. Другий підраховує всі ідентифікатори товарів, де батьківський ідентифікатор сбігається із переданим параметром, також товар має бути видимим.

Кількість сторінок отримується звичайним діленням загальної кількості товарів на кількість товарів розміщених на одній сторінці. Якщо параметр із кількістю сторінок не визначений, то він створюється та йому присвоюється одиниця. Має бути хочаб одна сторінка. Якщо раптом значення у змінній сторінці більше від значення загальної кількості сторінок, то присвоюємо цій змінній значення загальної кількості. Оскільки, якщо сторінок 10, номером активної сторінки просто не може бути 11.

Імя бренду та список товарів дістається вже описаними функціями.

У каркасі сторінки викликається функція посторінкової навігації. Вона приймає конкретні дані про активну сторінку та загальну кількість сторінок.

За замовчанням кнопки посторінкової навігації містять посилання: назад, вперед, початкова сторінка, кінцева сторінка, на стрінку вперед, сторінкою назад, на дві сторінки вперед та дві сторінки назад (рис. 4.8).

4.5 Реалізація пошуку по магазину

Функція пошуку невід’ємна частина майже кожного інтернет-додатку. Як і слідує з назви вона покликана знаходити контент, використовуючи введений користувачем запит.

В даному додатку реалізован повнотекстовий пошук по полю name. Для реалізації даного виду пошуку поля по яким буде проводитися пошук індексуються повнотекстовим типом (такі поля як text та varchar). Поле пошуку (рис. 4.9) розміщено у шапці додатку.

Запит введений у поле пошуку потрапляє у глобальний масив GET, а отже він потрапить у функцію без прямої передачі. Дані оброблюються та записуються у змінну. Також створюється масив, що міститиме результат пошуку.

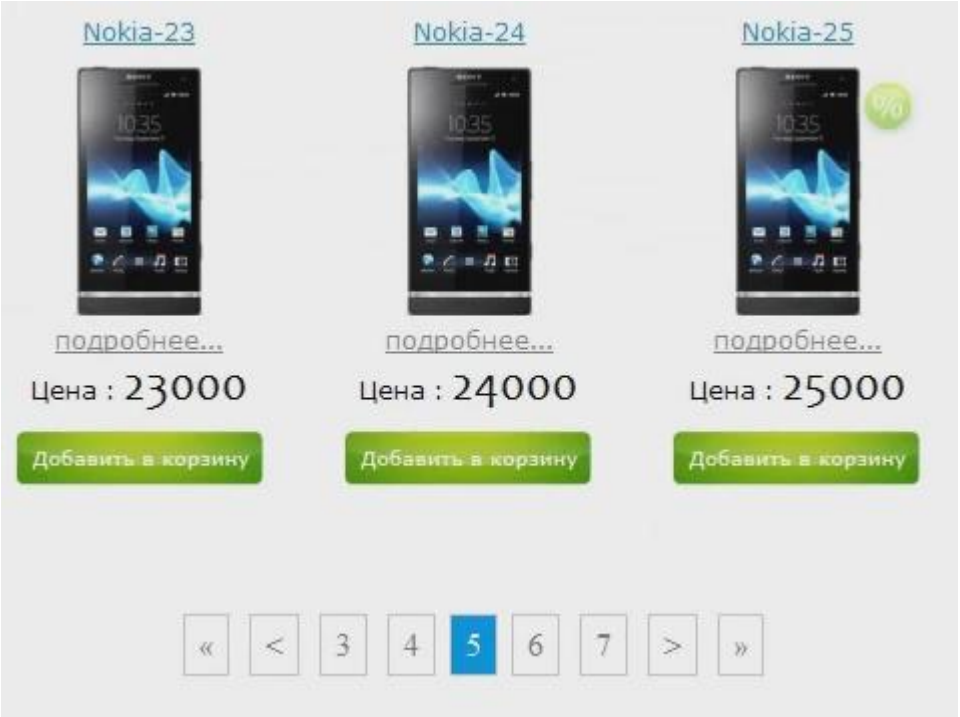

Рисунок 4.8 – Каталог із посторінковою навігацією

Рисунок 4.9 – Форма пошуку по додатку

Записується умова, в якій перевіряється кількість введених символів. У разі якщо кількість символів менша за чотири, виводимо помилку із пояснювальним текстом (рис. 4.10).

В іншому випадку формуємо запит до бази даних. Результат виконаного запиту кладемо у змінну. Перевіряємо чи повертає щось запит, у разі успіху за допомогою циклу while кладемо дані у змінну-масив, що була створена раніше. У випадки, коли запит не повернув жодного рядка виводимо повідомлення. Функція повертає масив, який видається результатом (рис. 4.11).

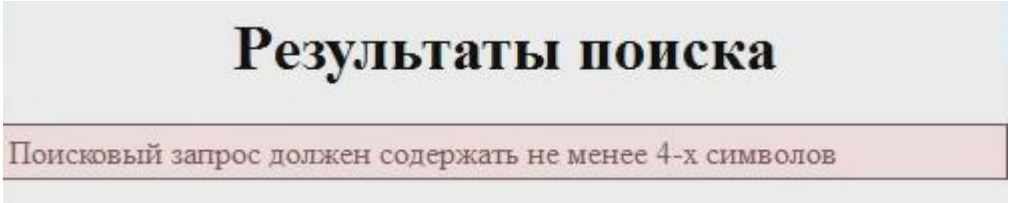

Рисунок 4.10 – Помилка пошуку

Рисунок 4.12 – Вивід результатів пошуку

Оскільки кількість виведених матеріалів по запиту може також сягати великої цифри, доцільно використовувати функцію посторінкової навігації і тут. Сама функція навігації викликається у конструкції основного документа. А в контролері передаються параметри для налаштування функції. Визначення кількості сторінок, формування параметра сторінки. Визначення кількості товарів та кількості сторінок. Встановлення мінімум однієї сторінки, стартова позиція запиту та кінцева позиція. А тукож встановлення кінцевої позиції.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи була розроблена інформаційно-пошукова система комерційного підприємства, яка представлена у вигляді Інтернет-магазину для покупки різних видів товарів через Інтернет.

В роботі було надано обґрунтування вибору засобів розробки інтернет-магазину і системи керування базами даних MySQL. Докладно описано етапи розробки дизайну сайта і верстку макету. Наведена структура бази даних інформаційної системи. Крім того, в роботі надано опис основних функціональних можливостей інтернет- магазину: регістрація та авторизація користувачів, додавання та видалення товарів у/з кошику, виконання заказу, можливості посторинкової навігації і пошуку товарів.

Якщо зовсім просто описувати інтернет додаток, то це набір запитів до бази данних, що оброблюється та виконується завдяки серверній мові програмування PHP. Однак у ньому було б мало сенсу, якби робилось це статично. Налаштування та стилізація оболонки, а також написання сценаріїв із можливісти приймати данні від користувача, що впершу чергу спрощує його стосунки із БД – це все комплекс заходів для створення гарного додатку, який може бути корисним. А безперечна користь додатку свідчить про доцільність його створення.

Отже, якщо розглядати доцільність написання додатку «інтернет-магазин» з точки зору електронної комерції сміло можна робити висновок, що такі проекти є цілком виправданими. Такий додаток має широке розповсюдження в мережі, оскільки створення власного інтернет-магазину це спосіб для фізичної кампанії збільшити обсяг продажів, за рахунок масового розповсюдження мережі інтернет.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. HTML: HyperText Markup Language. URL: http://uk.wikipedia.org /wiki/HTML (дата звернення 18.05.2019)
2. Комолова Н.В., Яковлева Е.С. HTML, XHTML и CSS. СПб.: Питер, 2012. 300 с.
3. Фрэйн Б. HTML5 и CSS3. Разработка сайтов для любых браузеров и устройств. СПб.: Питер, 2014. 298 с.
4. CSS: Cascading Style Sheets. URL:http://uk.wikipedia.org/wiki/CSS (дата звернення 18.05.2019)
5. Дакетт, Д. HTML и CSS. Разработка и дизайн веб-сайтов. М.: Эксмо, 2013. 480 с.
6. Дронов В. HTML 5, CSS 3 и Web 2.0. Разработка современных Web-сайтов. Спб.: БХВ-Петербург, 2011. 416 с
7. JavaScript. URL: http://uk.wikipedia.org/wiki/JavaScript (дата звернення 18.05.2019)
8. Гаевский, А.Ю. Создание Web-страниц и Web-сайтов: HTML и JavaScript. М.: Триумф, 2008. 454 с.
9. JavaScript: Онлайн підручник. URL: http://www.wisdomweb.ru/ JS/javascript-first.php (дата звернення 18.05.2019)
10. jQuery. URL:http://uk.wikipedia.org/wiki/JQuery (дата звернення 18.05.2019)
11. jQuery: Онлайн підручник URL: http://www.wisdomweb.ru/ JQ/jquery-first.php (дата звернення 18.05.2019)
12. PHP: Hypertext Preprocessor. URL: http://uk.wikipedia.org/wiki/PHP (дата звернення 18.05.2019)
13. Справочник по PHP. URL: http://www.php.su (дата звернення 18.05.2019)
14. Зандстра М. PHP: объекты, шаблоны и методики программирования. Издательский дом «Вильямс», 2011. 560 с.

15. Никсон, Р. Создаем динамические веб-сайты с помощью PHP, MySQL и JavaScript. СПб.: Питер, 2013. 496 с.
16. SQL: Structured Query Language. URL:http://uk.wikipedia.org/wiki/SQL (дата звернення 18.05.2019)
17. Справочное руководство по MySQL. URL:http://www.mysql.ru/docs/man/ (дата звернення 18.05.2019)
18. MySQL. Оптимизация производительности. СПб.: Символ-Плюс, 2010. 832 с.
19. AJAX: Asynchronous Javascript and XML. URL: http://uk.wikipedia.org/wiki/AJAX (дата звернення 18.05.2019)
20. MVC: Model View Controller. URL: https://uk.wikipedia.org/wiki/Model-View-Controller (дата звернення 18.05.2019)

ДОДАТОК А

Програмний код реалізації інформаційної системи

Controller.php

```
<?php
//открываем сессию
session_start();

//подключение модели
require_once MODEL;

//подключение библиотеки функций
require_once 'functions/functions.php';

//получение массива каталога
$cat = catalog();

//получение массива информера
$informers = informer();

//получение массива страниц
$pages = pages();

//получение названия новостей
$news = get_title_news();

//авторизация
if($_POST['auth']){
    authorization();
    if($_SESSION['auth']['user']){
        //если пользователь авторизовался ( в масиве появился элемент USER)
        echo "<p> Добро пожаловать, {$_SESSION['auth']['user']}</p>";
        exit();
    }else{
        //если авторизация неудачна
        echo $_SESSION['auth']['error'];
        unset($_SESSION['auth']);
        exit();
    }
}
//регистрация
if($_POST['reg']){
   registration();
   redirect();
}
//выход пользователя
if($_GET['do'] == 'logout'){
    logout();
    redirect();
}
//получение динамично части шаблона #content
$view = empty($_GET['view']) ? 'hits' : $_GET['view'];

switch($view){
    case('hits'):
        //лидеры продаж
      $eyestoppers = eyestoppers('hits');
    break;
    case('new'):
        //новинки
        $eyestoppers = eyestoppers('new');
    break;
    case('sale'):
        //распродажа
        $eyestoppers = eyestoppers('sale');
    break;
```

```
    case('cart'):
        /*корзина*/
        /*Получение способов доставки*/
        $dostavka = get_dostavka();
        //пересчет товаров в корзине
    if(isset($_GET['id'],$_GET['qty'])){
        $goods_id = abs((int)$_GET['id']);
        $qty = abs((int)$_GET['qty']);
        $qty = $qty - $_SESSION['cart'][$goods_id]['qty'];
        addtocart($goods_id, $qty);
        $_SESSION['total_sum'] = total_sum($_SESSION['cart']);//сумма заказа
        total_quantity(); //количество товаров в корзине + зашита от ввода
несуществуюшего товара
        }
        //удаление товара из корзины
        if(isset($_GET['delete'])){
            $id = (int)$_GET['delete'];
            if($id){
                delete_from_cart($id);
            }
            redirect();
        }
            if($_POST['order_x']){
            add_order();
            redirect();
        }
    break;
    case('informer'):
        $informer_id = abs((int)$_GET['informer_id']);
        $text_informer = get_text_informer($informer_id);
    break;
    case('addtocart'):
        //добавление в корзину
        $goods_id = abs((int)$_GET['goods_id']);
        addtocart($goods_id);
        $_SESSION['total_sum'] = total_sum($_SESSION['cart']);
        //количество товара в корзине + защита от ввода несуществующего ID
товара
       total_quantity();
        redirect();
        break;
    case('page'):
        //отдельная страница
        $page_id = abs((int)$_GET['page_id']);
        $get_page = get_page($page_id);
    break;
    case('cat'):
        //категории
        $category = abs((int) $_GET['category']);
        /*  Сортировка  */
        //массив параметров сортировки
        //ключи - то, что передаем GET параметрам
        //значения - то, что показываем пользователю и часть SQL запроса ,
который передаем в модель
        $order_p = array(
          'pricea' => array('от дешевых к дорогим', 'price ASC'),
          'priced' => array('от дорогих к дешевым', 'price DESC'),
          'datea' => array('по дате добавления к последним', 'date ASC'),
          'dated' => array('по дате добавления с последних', 'date DESC'),
          'namea' => array('от А до Я', 'name ASC'),
          'named' => array('от Я до А', 'name DESC') );
  $order_get = clear($_GET['order']);//получаем возможный параметр сортировки
        if(array_key_exists($order_get, $order_p)){
            $order = $order_p[$order_get][0];
            $order_db = $order_p[$order_get][1];
        }else{
            //по умолчанию сортировка по первому элементу массива $order_p
            $order = $order_p['namea'][0];
            $order_db = $order_p['namea'][1];
        }
        /*  Сортировка  */
```

```
        //параметры для навигации
        $perpage = PERPAGE; //кол-во товаров на страницу
        if(isset($_GET['page'])){
            $page = (int)$_GET['page'];
                if($page < 1) $page = 1;
        }else{
            $page = 1;
        }
        $count_rows = count_rows($category);//общее количество товаров
        $pages_count = ceil($count_rows / $perpage);//количество страниц
        if(!$pages_count) $pages_count = 1; // минимум 1 страница
        if($page > $pages_count) $page = $pages_count;
        $start_pos = ($page - 1) * $perpage;//начальная позиция запроса
        $brand_name = brand_name($category);
        $products = products($category, $order_db, $start_pos, $perpage);  //
получаем массив из модели
        break;
    case('reg'):
        //форма регистрации
        break;
    case('search'):
        // поиск
        $result_search = search();
        //параметры для навигации
        $perpage = 30; //кол-во товаров на страницу
        if(isset($_GET['page'])){
            $page = (int)$_GET['page'];
                if($page < 1) $page = 1;
        }else{
            $page = 1;
        }
        $count_rows = count($result_search);//общее количество товаров
        $pages_count = ceil($count_rows / $perpage);//количество страниц
        if(!$pages_count) $pages_count = 1; // минимум 1 страница
        if($page > $pages_count) $page = $pages_count;
        $start_pos = ($page - 1) * $perpage;//начальная позиция запроса
        $endpos = $start_pos + $perpage; // до какого товара будет вывод на
странице
        if($endpos > $count_rows) $endpos = $count_rows;
        break;
    case('filter'):
        //выбор по параметрам
        $startprice = (int)$_GET['startprice'] ;
        $endprice = (int)$_GET['endprice'] ;
        $brand = array();
        if($_GET['brand']){
        foreach($_GET['brand'] as $value){
            $value = (int)$value;
            $brand[$value] = $value;
        }
        }
            if($brand){
                $category =implode(',', $brand);
            }
        $products = filter($category, $startprice, $endprice);
    break;
    case('product'):
        //отдельный товар
        $goods_id = abs((int)$_GET['goods_id']);
        if($goods_id){
            $goods = get_goods($goods_id);
            $brand_name    =    brand_name($goods['goods_brandid']);//хлебные
крошки
        }
    break;
    case('archive'):
    //архив новостей
    //параметры для навигации
        $perpage = 10; //кол-во товаров на страницу
        if(isset($_GET['page'])){
            $page = (int)$_GET['page'];
```

```
                if($page < 1) $page = 1;
        }else{
            $page = 1;
        }
        $count_rows = count_news();//общее количество товаров
        $pages_count = ceil($count_rows / $perpage);//количество страниц
        if(!$pages_count) $pages_count = 1; // минимум 1 страница
        if($page > $pages_count) $page = $pages_count;
        $start_pos = ($page - 1) * $perpage;//начальная позиция запроса
        $all_news = get_all_news($start_pos, $perpage);
    break;
    case('news'):
        //отдельная новость
        $news_id = abs((int)$_GET['news_id']);
        $news_text = get_news_text($news_id);
    break;
    default:
    //если из адресной строки получено имя, несуществующего шаблона
        $view = 'hits';
        $eyestoppers = eyestoppers('hits');
        break;
}
//подключение вида
require_once TEMPLATE.'index.php';
?>
```

Model.php

```
<?php
/* =========Каталог - получение массива============ */
function catalog(){
    $query = 'SELECT * FROM brands ORDER BY parent_id, brand_name';
    $res = mysql_query($query) or die (mysql_query());

    //массив категорий
    $cat = array();

    while($row = mysql_fetch_assoc($res)){
        if(!$row['parent_id']){
            $cat[$row['brand_id']][] = $row['brand_name'];
        }else{
            $cat[$row['parent_id']]        ['sub'][$row['brand_id']]        =
$row['brand_name'];
        }
    }
    return $cat;
}
/* =========Каталог - получение массива============ */

/* =========Информеры - получение массива============ */
function informer(){
    $query = "SELECT * FROM links
              INNER JOIN informers ON
                    links.parent_informer  =  informers.informer_id  ORDER  BY
informer_position, links_position";
    $res = mysql_query($query) or die (mysql_query());

    $informers = array();
    $name = ' '; // флаг имени информера

    while($row = mysql_fetch_assoc($res)){
        if($row['informer_name'] != $name){ //если такого информера еще нет
            $informers[$row['informer_id']][]   =   $row['informer_name'];   //
добавляем информер в массив
            $name = $row['informer_name'];
        }
        $informers[$row['parent_informer']]['sub'][$row['link_id']]         =
$row['link_name'] ; //заносим страницы в информер
    }
    return $informers;
}
```

```
/* =========Информеры - получение массива============ */

/* ========= Айстопперы ============ */
function eyestoppers($eyestopper){
    $query = "SELECT goods_id, name, img, price FROM goods
                WHERE visible = '1' AND $eyestopper = '1'";
    $res = mysql_query($query);

    $eyestoppers = array();

    while($row = mysql_fetch_assoc($res)){
        $eyestoppers[] = $row;
    }

    return $eyestoppers;
}
/* ========= Айстопперы ============ */
/*Получение количества товаров для навигации*/
function count_rows($category){
    $query = "(SELECT COUNT(goods_id) as count_rows FROM goods
                            WHERE goods_brandid = $category AND visible =
'1')
                        UNION
                (SELECT COUNT(goods_id) as count_rows FROM goods
                            WHERE goods_brandid IN
                            (SELECT brand_id FROM brands WHERE parent_id =
$category)
                                AND visible = '1')";
    $res = mysql_query($query) or die(mysql_error);
    while($row = mysql_fetch_assoc($res)){
        if($row['count_rows'])$count_rows = $row['count_rows'];
    }
    return $count_rows;
}
/*Получение количества товаров для навигации*/

/*================     Получение     массива     товаров     по     категории
==================*/
function products($category, $order_db, $start_pos, $perpage){
    $query = "(SELECT goods_id, name, img, anons, price, hits, new, sale,
date FROM goods
                  WHERE goods_brandid = $category  AND visible = '1')
                        UNION
                (SELECT goods_id, name, img, anons, price, hits, new, sale,
date FROM goods
                WHERE goods_brandid IN
               (SELECT brand_id FROM brands WHERE parent_id = $category)
                AND visible = '1') ORDER BY $order_db LIMIT $start_pos,
$perpage";
    $res = mysql_query($query) or die(mysql_error());
    $products = array();
        while($row = mysql_fetch_assoc($res)){
            $products[] = $row;
        }
    return $products;
}
/*================     Получение     массива     товаров     по     категории
==================*/

/*Получение названий для хлебных крох*/
function brand_name($category){
    $query = "(SELECT brand_id, brand_name FROM brands
                      WHERE brand_id = (SELECT parent_id, FROM brands WHERE
brand_id = $category))
                      UNION
                      (SELECT brand_id, brand_name FROM brands WHERE brand_id =
$category )";
    $res = mysql_query($query);
    $brand_name = array();
    while($row = mysql_fetch_assoc($res)){
        $brand_name[] = $row;
```

```
    }
    return $brand_name;
}
/*Полчение названий для хлебных крох*/
/*================Выбор по параметрам=================*/
function filter($products, $startprice, $endprice ){
    $products = array();
    if($category || $endprice){
        $predicat1 = "visible = '1'";
        if($category){
            $predicat1 .= "AND goods_brandid IN($category)";
            $predicat2 .= "UNION
                       (SELECT goods_id, name, img, price, hits, new, sale
                               FROM goods
                               WHERE goods_brandid IN
                               (
                  SELECT brand_id FROM brands WHERE parent_id IN($category)
                       ) AND visible ='1'";
                if($endprice)  $predicat2  .=  " AND price BETWEEN $startprice
AND $endprice";
                $predicat2 .= ")";
        }
        if($endprice){
            $predicat1 .="AND price BETWEEN $startprice AND $endprice";
        }
        $query = "(SELECT goods_id, name, img, price, hits, new, sale
                        FROM goods WHERE $predicat1)
                                              $predicat2 ORDER BY name";
        $res = mysql_query($query) or die(mysql_error());
     if(mysql_num_rows($res) > 0) {
        while($row = mysql_fetch_assoc($res)){
            $products[] = $row ;
        }
     }else{
        $products['notfound'] = "<div class='error'> По указаным параметрам
ничего не найдено. </div>" ;
     }
    }else{
        $products['notfound'] = "<div class='error'> Вы не указали параметры
подбора. </div>" ;
    }

    return $products;
}
/*================Выбор по параметрам=================*/
/*============================   Сумма заказов в корзине   +   атрибуты товара
==============================*/
    function total_sum($goods){
        $total_sum = 0;
        $str_goods = implode(',', array_keys($goods));
        $query = "SELECT goods_id, name, price, img
                        FROM goods
                                WHERE goods_id IN($str_goods)";
        $res = mysql_query($query) or die(mysql_error());

        while($row = mysql_fetch_assoc($res)){
            $_SESSION['cart'][$row['goods_id']]['name'] = $row['name'];
            $_SESSION['cart'][$row['goods_id']]['img'] = $row['img'];
            $_SESSION['cart'][$row['goods_id']]['price'] = $row['price'];
            $total_sum   +=     $_SESSION['cart'][$row['goods_id']]['qty']   *
$row['price'];
        }
        return $total_sum;
    }
/*============================   Сумма заказов в корзине   +   атрибуты товара
==============================*/
/*====================== Регистрация ======================*/
   function registration(){
    $error = '';//флаг проверки пустых полей

    $login = clear($_POST['login']);
```

```
    $pass = trim($_POST['pass']);
    $name = clear($_POST['name']);
    $email = clear($_POST['email']);
    $phone = clear($_POST['phone']);
    $adress = clear($_POST['adress']);

    if(empty($login)) $error .= '<li>Не указан логин!</li>';
    if(empty($pass)) $error .= '<li>Не указан пароль!</li>';
    if(empty($name)) $error .= '<li>Не указано ФИО!</li>';
    if(empty($email)) $error .= '<li>Не указан E-маил!</li>';
    if(empty($phone)) $error .= '<li>Не указан телефон!</li>';
    if(empty($adress)) $error .= '<li>Не указан адрес!</li>';
    if(empty($error)){
        //если все поля заполнены (переменная ерор пуста)
        // проверяем нет ли пользователя в базе данных
        $query = "SELECT customer_id FROM customers
                        WHERE customers.login = '$login' LIMIT 1";
        $res = mysql_query($query) or die(mysql_error());
        $row = mysql_num_rows($res); //1- такой юзер есть, 0 - нет.
        if($row){
            //если такой  логин уже есть
        $_SESSION['reg']['res'] = "<div class='error'>Пользователь с таким
логином, уже зарегестрирован.</div>";
        $_SESSION['reg']['name'] = $name;
        $_SESSION['reg']['email'] = $email;
        $_SESSION['reg']['phone'] = $phone;
        $_SESSION['reg']['adress'] = $adress;
        }else{
            //если такого логина нет -  регистрация
            $pass = md5($pass) ;
            $query = "INSERT INTO customers (name, email, phone, adress, log-
in, password)
                            VALUES ('$name', '$email', '$phone', '$adress',
'$login', '$pass')";
            $res = mysql_query($query) or die(mysql_error());
            if(mysql_affected_rows()>0){
                //если запись добавлена
                $_SESSION['reg']['res'] = "<div class='success'>Регистрация
прошла успешно!</div>";
                $_SESSION['auth']['user'] = $name ;
                $_SESSION['auth']['customer_id'] = mysql_insert_id();
                $_SESSION['auth']['email'] = $email;
            }else{
                 $_SESSION['reg']['res'] = "<div class='error'>Ошибка<br />
<ul>$error</ul></div>";
                $_SESSION['reg']['login'] = $login;
                $_SESSION['reg']['name'] = $name;
                $_SESSION['reg']['email'] = $email;
                $_SESSION['reg']['phone'] = $phone;
                $_SESSION['reg']['adress'] = $adress;
            }
        }
    }else{
        //если не все поля заполнены

        $_SESSION['reg']['res'] = "<div class='error'>Не заполнены обязатель-
ные поля:<br /> <ul>$error</ul></div>";
        $_SESSION['reg']['login'] = $login;
        $_SESSION['reg']['name'] = $name;
        $_SESSION['reg']['email'] = $email;
        $_SESSION['reg']['phone'] = $phone;
        $_SESSION['reg']['adress'] = $adress;
    }
}
/*==================== Регистрация ======================*/
/*==================== Авторизация ======================*/
function authorization(){
    $login = mysql_real_escape_string(trim($_POST['login']));
    $pass = trim($_POST['pass']);

    if(empty($login) || empty($pass)){
```

```
        //если пусты поля логин/пароль
        $_SESSION['auth']['error'] = "Поля логин/пароль должны быть заполне-
ны!";
    }else{
        //если данные из полей получены
        $pass = md5($pass);

        $query = "SELECT customer_id, name, email FROM customers WHERE login
= '$login' AND password = '$pass' LIMIT 1";
        $res = mysql_query($query) or die(mysql_error());
        if(mysql_num_rows($res) == 1){
            //если авторизация успешна
            $row = mysql_fetch_row($res);
            $_SESSION['auth']['customer_id'] = $row[0];
            $_SESSION['auth']['user'] = $row[1];
            $_SESSION['auth']['email'] = $row[2];
        }else{
            //если не успешна
            $_SESSION['auth']['error'] = "Поля логин/пароль введены не вер-
но!";
        }
    }
}
function get_dostavka(){
    $query = "SELECT * FROM dostavka";
    $res = mysql_query($query);

    $dostavka = array();

    while($row = mysql_fetch_assoc($res)){
        $dostavka[] = $row;
    }
    return $dostavka;

}
    function add_order(){
        //получаем общие данные для всех (авторизованые пользователи и не ав-
торизованые)
            $dostavka_id = (int)$_POST['dostavka'];
            if(!$dostavka_id) $dostavka_id = 1;
            $prim = clear($_POST['prim']);
            if($_SESSION['auth']['user'])           $customer_id           =
$_SESSION['auth']['customer_id'];

            if(!$_SESSION['auth']['user']){
                $error = '';//флаг проверки пустых полей


                $name = clear($_POST['name']);
                $email = clear($_POST['email']);
                $phone = clear($_POST['phone']);
                $adress = clear($_POST['adress']);


                if(empty($name)) $error .= '<li>Не указано ФИО!</li>';
                if(empty($email)) $error .= '<li>Не указан E-маил!</li>';
                if(empty($phone)) $error .= '<li>Не указан телефон!</li>';
                if(empty($adress)) $error .= '<li>Не указан адрес!</li>';

                if(empty($error)){
                    //добавляем гостя в заказчики (но без данных авторизации)

                   $customer_id   =   add_customer($name,   $email,   $phone,
$adress);
                   if(!$customer_id)  return  false;//прекращаем  выполнение
скрипта в случае возникновения ошибки добавления гостя заказчика
                }else{
                    //не заполнены обязательные поля
                    $_SESSION['order']['res'] = "<div class='error'>Не запол-
нены обязательные поля:<br /> <ul>$error</ul></div>";
                    $_SESSION['order']['name'] = $name;
```

```
                    $_SESSION['order']['email'] = $email;
                    $_SESSION['order']['phone'] = $phone;
                    $_SESSION['order']['prim'] = $adress;
                    return false;
                }
            }
        $_SESSION['order']['email'] = $email;
        save_order($customer_id, $dostavka_id, $prim);

    }
/*==================Добавление заказчика -
гостя==========================*/
    function add_customer($name, $email, $phone, $adress){
        $query = "INSERT INTO customers (name, email, phone, customers)
                        VALUES ('$name', '$email', '$phone', '$adress')";
        $res = mysql_query($query);
        if(mysql_affected_rows() > 0){
            //если гость был добавлен в заказчики - получаем его ID
           return mysql_insert_id();

        }else{
                //если произошла ошибка при добавлении - не добавлен новый
заказчик
                    $_SESSION['order']['res'] = "<div class='error'>Произошла
ошибка при регистрации заказа:<br /> <ul>$error</ul></div>";
                    $_SESSION['order']['name'] = $name;
                    $_SESSION['order']['email'] = $email;
                    $_SESSION['order']['phone'] = $phone;
                    $_SESSION['order']['prim'] = $adress;
                    return false;
        }
    }
/*================================================*/
/*==================Сохранение заказа============================*/
    function save_order($customer_id, $dostavka_id, $prim){
        $query = "INSERT INTO orders (`customer_id`, `date`, `dostavka_id`,
`prim`)
                        VALUES ($customer_id, NOW(), $dostavka_id, '$prim')";
        mysql_query($query) or die(mysql_error());
        if(mysql_affected_rows() == -1){
            //если не получилось сохранить заказ - удаляем заказчика
            mysql_query("DELET FROM customers
                            WHERE  customer_id  =  $customer_id  AND  login  =
''");
                            return false ;
        }
            $order_id = mysql_insert_id(); //ID сохраненного заказа
            foreach($_SESSION['cart'] as $goods_id => $value){
            $val .= "($order_id, $goods_id, {$value['qty']}),";
                }
            $val = substr($val, 0, -1); //удаляем последнюю запятую

            $query = "INSERT INTO zakaz_tovar (orders_id, goods_id, quantity)
                            VALUES $val";
            mysql_query($query) or die(mysql_error());
            if(mysql_affected_rows() == -1){
                //если не  выгрузился заказ - удаляем заказчика (customers) и
сам заказ (orders)
                mysql_query("DELETE FROM orders WHERE order_id = $order_id");
                mysql_query("DELET FROM customers
                            WHERE  customer_id  =  $customer_id  AND  login  =
''");
                return false;
            }
            if($_SESSION['auth']['email'])              $email              =
$_SESSION['auth']['email'];
                else $email = $_SESSION['order']['email'];
            mail_order($order_id, $email);

            //если заказ выгрузился
            unset($_SESSION['cart']);
```

```
            unset($_SESSION['total_sum']);
            unset($_SESSION['total_quantity']);
            $_SESSION['order']['res'] = "<div class='success'>Спасибо за Ваш
заказ! В ближайшее время с Вами свяжется менеджер для согласования
заказа.</div>";
            return true;
    }
/*===============Отправка уведомлений о заказе=================*/
function mail_order($order_id, $email){

    //Тема письма
    $subject = "Заказ в Интернет-магазине" ;
    //заголовки
    $headers .= "Content-type: text/plain; charset=utf8\r\n" ;
    $headers .= "From: ISHOP";
    //тело письма
    $mail_body = "Благодарим Вас за заказ!\r\n Номер Вашего заказа - {$or-
der_id}\r\n\r\n
                    Заказанные товары: \r\n" ;
                    foreach($_SESSION['cart'] as $goods_id => $value){
                        $mail_body .= "Наименование: {$value['name']}, Цена:
{$value['price']}, Количество: {$value['qty']} шт.\r\n" ;
                    }
    $mail_body   .=   "\r\nИтого:   {$_SESSION['total_quantity']}   на   сумму:
{$_SESSION['total_sum']} ";
                    // отправка писем
 @mail($email, $subject, $mail_body, $headers); // письмо пользователю
 @mail(ADMIN_EMAIL, $subject, $mail_body, $headers); // письмо администратору
}
/*===============Отправка уведомлений о заказе=================*/
/*===============Поиск=================*/
function search(){
    $search = clear($_GET['search']);
    $result_search = array(); //результат поиска
    if(mb_strlen($search, 'UTF-8') < 4){
        $result_search['notfound']  =  "<div  class='error'>Поисковый  запрос
должен содержать не менее 4-х символов</div>" ;
    }else{
        $query = "SELECT goods_id, name, img, price, hits, new, sale
                    FROM goods
                        WHERE  MATCH(name)  AGAINST('{$search}*'  IN  BOOLEAN
MODE) AND visible = '1'";
        $res = mysql_query($query) or die(mysql_error());
        if(mysql_num_rows($res) > 0){
           while($row_search = mysql_fetch_assoc($res)){
            $result_search[] = $row_search;
           }
        }else{
            $result_search['notfound'] = "<div class='error'>По Вашему запро-
су ничего не найдено.</div>" ;
        }
    }
    return $result_search;
}
/*===============Поиск=================*/
/*=============== Отдельный товар =================*/
    function get_goods($goods_id){
        $query = "SELECT * FROM goods
                    WHERE goods_id = $goods_id AND visible = '1'";
        $res = mysql_query($query);

        $goods = array();
        $goods = mysql_fetch_assoc($res);
        if($goods['img_slide']){
            $goods['img_slide'] = explode("|", $goods['img_slide']);
        }
        return $goods;
    }
/*=============== Отдельный товар =================*/
/*Страницы*/
function pages(){
```

```
    $query = "SELECT page_id, title FROM pages ORDER BY position";
    $res = mysql_query($query);
    $pages = array();
    while($row = mysql_fetch_assoc($res)){
        $pages[] = $row;
    }
    return $pages;
}
/*Страницы*/
/*Названия новостей*/
    function get_title_news(){
        $query = "SELECT news_id, title, date FROM news ORDER BY news_id DESC
LIMIT 2";
        $res = mysql_query($query);
        $news = array();
        while($row = mysql_fetch_assoc($res)){
            $news[] = $row;
        }
        return $news;
    }
/*Названия новостей*/
/*Отдельная новость*/
function get_news_text($news_id){
    $query = "SELECT title, text, date FROM news WHERE news_id = $news_id";
    $res = mysql_query($query);

    $news_text = array();
    $news_text = mysql_fetch_assoc($res);
    return $news_text;
}
/*Отдельная новость*/
/*Все новости*/
function get_all_news(){
    $query = "SELECT news_id, title, anons, date FROM news ORDER BY news_id
DESC LIMIT $start_pos, $perpage";
    $res = mysql_query($query);
    $all_news = array();
    while($row = mysql_fetch_assoc($res)){
        $all_news[] = $row;
    }
    return $all_news;
}
/*Все новости*/
/*Навигация новости*/
function count_news(){
    $query = "SELECT COUNT(news_id) FROM news";
    $res = mysql_query($query);
    $count_news = mysql_fetch_row($res);
    return $count_news[0];
}
/*Навигация новости*//*Отдельная страница*/
function get_page($page_id){
    $query = "SELECT title, keywords, description, text FROM pages WHERE
page_id = $page_id";

    $res = mysql_query($query);
    $get_page = array();
    $get_page = mysql_fetch_assoc($res);
    return $get_page;
}
function get_text_informer(){
    $query = "SELECT link_id, link_name, text, informers.informer_id, inform-
ers.informer_name
                        FROM links
                                LEFT JOIN informers ON informers.informer_id
= links.parent_informer
                                    WHERE link_id = $informer_id";
    $res = mysql_query($query);
    $text_informer = array();

    $text_informer = mysql_fetch_assoc($res);
```

```
}
?>
```

Config.php

```
<?php
   // defined(I) or die('Access denide config');
    //Domen
    define('PATH', 'http://ishop.ua/');
    //Model
    define('MODEL', 'model/model.php');
    //kontroller
    define('CONTROLLER', 'controller/controller.php');
    //BUD
    define('VIEW', 'views/');
    //AKTIVE SHABLON
    define('TEMPLATE', VIEW.'ishop/');
    //папка с картинками контента
    define('PRODUCTIMG', PATH.'userfiles/');
//Параметры подключения к Базе Данных
    //SERVER BD
    define('HOST', 'localhost');
    //polzovatel
    define('USER', 'ishop_user');
    //parol
    define('PASS', '123');
    //BD
    define('DB', 'ishop');
    //mail admina
    define('ADMIN_EMAIL', 'admin@ishop.com');
    //количество товара на страницу
    define('PERPAGE', 9);
//Константа опредиляющая название магазина
    //nazvanie magazina
    define('TITLE', 'Интернет магазин сотовых телефонов');
//подключение к БД и установка кодировки
mysql_connect(HOST, USER, PASS) or die('No connect to server');
mysql_select_db(DB) or die ('No connect to DB');
mysql_query("SET NAMES 'UTF8'") or die('Cant set charset');?>
```