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

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студентки *Сак Марії Василівни*
(ПІБ)

академічної групи *121-17-1*
(шифр)

спеціальності *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

освітньої програми *Інженерія програмного забезпечення*
(назва освітньої програми)

на тему: *Розробка веб-орієнтованого програмного забезпечення для продажу смартфонів*

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

Дніпро
2021

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

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

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

« » _____ 2021 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студентки *121-17-1* *Сак Марії Василівни*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка веб-ориентованого програмного забезпечення для продажу смартфонів*

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

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

Завдання видав _____ *проф.Корниенко В.І.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Сак М.В.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2021 р.*

Термін подання кваліфікаційної роботи до ЕК: *16.06.2021 р.*

РЕФЕРАТ

Пояснювальна записка: 86 с., 13 рис., 4 табл., 3 дод., 24 джерела.

Об'єкт розробки: інтернет-магазин з продажу смартфонів та мобільних телефонів на мові програмування PHP.

Мета кваліфікаційної роботи: розробка інтернет-магазину використовуючи сучасну мову програмування PHP.

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

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

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

В економічному розділі визначено трудомісткість розробленого програмного продукту, проведений підрахунок вартості роботи по створенню застосунку та розраховано час на його створення.

Практичне значення полягає у створенні інтернет-магазину з продажів смартфонів та мобільних телефонів.

Актуальність програмного продукту визначається великим попитом на подібні інструменти.

Список ключових слів: ІНТЕРНЕТ-МАГАЗИН, САЙТ, PHP, MVC, HTML, FRONTCONTROLLER, CSS.

ABSTRACT

Explanatory note: 86 pages, 13 figures, 4 table, 3 appendices, 24 sources.

Object of development: online store selling smartphones and mobile phones in the PHP programming language.

The purpose of the qualification work: development of an online store using the modern PHP programming language.

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

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

The second section analyzes the existing solutions, chooses the choice of platform for development, performs design and development of the program, describes the algorithm and structure of the application, defines the input and output data, provides characteristics of the parameters of technical means, describes calling and loading the site, describes the Internet -shop.

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

The practical significance lies in the creation of an online store selling smartphones and mobile phones.

The relevance of the software product is determined by the high demand for such tools.

Keywords: ONLINE STORE, SITE, PHP, MVC, HTML, FRONTCONTROLLER, CSS.

ЗМІСТ

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ

ЕОМ - електронно-обчислювальна машина;

ІС - інформаційна система;

ООП - об'єктно-орієнтоване програмування;

ОС - операційна система;

ПЗ - програмне забезпечення;

ІТ - інформаційні технології;

HTML- Hypertext Markup Language;

CSS- Cascading Style Sheets;

DOM- Document Object Model.

ВСТУП

З розвитком Інтернету в Україні і в усьому світі спостерігається зростання активності в області онлайнової торгівлі. На сьогоднішній день через Інтернет можна придбати практично будь-які товари і послуги.

Електронна комерція (e-commerce) - це прискорення більшості бізнес-процесів за рахунок їх проведення електронним чином. У цьому випадку інформація передається безпосередньо до одержувача, минаючи стадію створення паперової копії на кожному етапі. Таким чином, електронну комерцію можна характеризувати як ведення бізнесу через Інтернет. У сучасному суспільстві все більше компаній переносять значну частину ділового спілкування в Мережу.

Серед відмінних рис онлайн комерції перед традиційною торгівлею можна відзначити:

• відсутність географічних, тимчасових і мовних бар'єрів, що дозволяє просувати товари і послуги на нові ринки збуту;

• нижчий рівень витрат виробництва і обігу, що досягається шляхом впровадження нових технологій в усі сфери діяльності компаній: починаючи від закупівель сировини і матеріалів і закінчуючи дистрибуцією готової продукції і пост-продажним обслуговуванням;

• більш високий рівень конкуренції: відстань між магазинами всього кілька секунд - саме цей час необхідно для завантаження відповідного сайту;

• потенційна ємність електронного магазину значно перевищує ємність традиційних магазинів через відсутність фізичних обмежень на складські та торговельні приміщення.

Інтернет-комерція включає в себе інтернет-магазини, біржі та посередницькі інтернет-аукціони, операції взаємодії між підприємствами, організацію різних каталогів і засобів спілкування користувачів в Інтернеті, проведення рекламних кампаній певних товарів або ресурсів, і т.д. Число

товарів і послуг постійно поповнюється, пропонуючи користувачам все більше число можливостей.

Комерційні програми, в першу чергу, повинні приносити прибуток. В основному, дохід від інтернет-проектів досягається декількома шляхами, серед яких можна виділити:

• надання рекламних місць;

• прямі продажі товарів, послуг і інформації;

• технічна підтримка;

• надання інструментів для ведення бізнесу.

Рекламні місця надаються практично на всіх існуючих інтернет-сайтах. Прибуток від реклами досягається, в першу чергу, за рахунок служби обміну банерами.

Прямі продажі товарів і послуг використовуються для отримання прибутку інтернет-магазинами, банківськими та платіжними організаціями.

Технічна підтримка через Інтернет дозволяє значно економити кошти, пропонуючи користувачам більш швидку і дешеву, в порівнянні з аналогами, систему підтримки дилерів і кінцевих користувачів.

Завдання кваліфікаційної роботи та об'єкт його діяльності безпосередньо пов'язані зі спеціальністю 121 «Інженерія програмного забезпечення» та відповідає узагальненій тематиці кваліфікаційних робіт і переліку зазначених компетенцій.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Загальні відомості з предметної галузі

Електронна торгівля є активно розвивається частиною електронної комерції, а значить, і електронного бізнесу; в загальному розумінні - це комерційна діяльність, коли процес купівлі / продажу товарів і послуг (весь цикл комерційно-фінансової транзакції або її частина) здійснюється електронним чином, тобто із застосуванням інтернет-технологій.

Подібний вид комерції отримав назву системи "Бізнес-Споживач" (Business-to-Customer - B2C) [8]. До систем B2C відносяться:

• web-вітрина - оформлений засобами web-дизайну прайс-лист торгової компанії, який не містить бізнес-логіки торгового процесу;

• інтернет-магазин, який містить, крім web-вітрини, всю необхідну бізнес-логіку для управління процесом інтернет-торгівлі (бек-офіс);

• торгова інтернет-система, яка являє собою інтернет-магазин, бек-офіс якого повністю (в режимі реального часу) інтегрований в торговий бізнес-процес компанії.

З точки зору покупця все три рішення виглядають однаково. Пов'язано це з тим, що покупець має справу лише з зовнішнім оформленням будь-якої системи, а це завжди web-каталог, доповнений системою навігації і, за бажанням продавця, системою оформлення замовлень. Практично переваги покупця залежать тільки від зручності використання web-каталогу та системи навігації [1].

Переваги інтернет-магазинів і торгових інтернет-систем полягають в тому, що в цьому випадку покупцеві можуть запропонувати відразу оформити замовлення, виписати рахунок з урахуванням вартості доставки і страховки, а також надати більш гнучку систему знижок. Крім того, він зможе побачити реальний стан складу і отримати інформацію про проходження свого

замовлення. Покупець, в першу чергу, буде купувати товари на сайтах тих компаній, які нададуть йому кращі ціни і хороший сервіс. Саме на цих конкурентних перевагах будують свої плани по залученню постійної клієнтури інтернет-торговці.

З точки зору продавців ці три рішення різняться досить значно. Web-вітрина обходиться торговельним компаніям недорого, проте, має ряд істотних недоліків:

• web-вітрина не передбачає оформлення замовлення зі складу;

• використання web-вітрини не применшує витрати продавців на утримання штату і операційні витрати;

• web-вітрина являє собою дуже неповоротку рішення з точки зору управління і недостатньо гнучке з точки зору організації маркетингових акцій.

Імідж компанії, що відкрила і підтримуючої просту web-вітрину завжди гірше, ніж у компанії, що організувала торгівлю за допомогою повнофункціонального інтернет-магазину або торгових інтернет-систем.

Таким чином, організація інтернет-торгівлі за допомогою web-вітрини виявляється для торгової компанії малоефективним і часто нерентабельним справою.

Інтернет-магазин істотно більш вигідний торгової компанії, яка хоче реально управляти всім процесом інтернет-торгівлі, маркетинговими акціями, торгувати і на замовлення, і зі складу, зменшити число менеджерів з продажу.

На створення інтернет-магазину буде потрібно більше разових витрат у порівнянні з вітриною, але вони будуть набагато ефективнішими.

Застосування сучасних мережевих технологій в бізнесі не обмежується створенням Web-сайту або електронного каталогу з можливістю оформлення замовлення, а має на увазі глибинну перебудову способів ведення торговельних операцій.

В першу чергу, використання технологій онлайнової торгівлі необхідно компаніям, що мають розвинену регіональну партнерську мережу, тому що дозволить значно знизити вартість обробки замовлень. Після впровадження

методики роботи з регіональними партнерами через Інтернет, компанія зможе скоротити витрати на обробку замовлень більш, ніж в 2 рази.

Підводячи підсумки, виділимо основні переваги інтернет-торгівлі:

• створення альтернативних каналів продажів;

• збільшення оперативності отримання інформації, особливо при міжнародних операціях;

• скорочення циклу виробництва та продажу;

• зниження витрат, пов'язаних з обміном інформацією за рахунок використання більш дешевих засобів комунікацій;

• легке і швидке інформування партнерів і клієнтів про продукти і послуги.

Ще одним важливим достоїнством електронних продажів є широкий вибір засобів оплати. На сьогоднішній день домінуючим платіжним засобом онлайн-покупок є кредитні картки; чималим попитом користуються також смарт-карти, цифрові гроші (digital cash), мікроплатежі, а також електронні чеки.

Незважаючи на видиму легкість здійснення покупок, електронна торгівля має і зворотну сторону: збільшення кількості помилок користувача - це істотні втрати для магазину. Тому, найчастіше, потрібно перевірка менеджером кожного окремого факту замовлення. Виняток становить лише продаж інформаційного продукту, який можна доставити безпосередньо через Інтернет [6].

Основні проблеми електронної комерції лежать на стиках Інтернету і реальної діяльності. У звичайній торгівлі покупець звик до того, що у нього є можливість оцінити товар візуально, визначити його якість і дізнатися характеристики. В електронній торгівлі він цієї можливості позбавлений. Максимум, на що можна розраховувати, це фотографія товару і перерахування його характеристик. Найчастіше цієї інформації достатньо, проте, в більшості випадків, в справу вступають емоційні і психологічні чинники.

Більшість електронних магазинів мають проблеми і з доставкою товарів, особливо якщо ціна товару невелика. Проблеми також виникають при необхідності оплати в електронному магазині. Для цього є безліч причин: недовіру громадян по відношенню до банківської системи, в цілому, і безготівковим платежам, зокрема, невпевненість в безпеці проведення транзакцій через Інтернет.

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

По суті одиночне відвідування сайту, це точка входу в об'єктну модель, через яку відкривається доступ до інших об'єктів моделі.

Типова модель інтернет-магазину складається з наступних функціональних частин:

- каталог товарів;
- Пошукова система;
- призначена для користувача корзина;
- реєстраційна форма;
- форма відправки замовлення.

Каталог товарів являє собою складну і багаторівневу структуру даних, яка повинна простим і зрозумілим способом виробляти упорядкування товарів. Простіше за все такий каталог представити у вигляді дерева об'єктів, верхній рівень якого складається зі списку розділів. Розділи можуть містити підрозділи або посилання на конкретний товар. Таке впорядкування просто необхідно для зручного і швидкого пошуку і замовлення товарів.

Пошукова система є обов'язковим елементом динамічного каталогу і реалізується на стороні сервера. Вона дає користувачеві можливість швидкого пошуку інформації, що особливо важливо в тому випадку, коли каталог являє

собою досить розгалужену структуру даних з великою кількістю розділів, підрозділів і товарів, а користувач погано уявляє, в якому розділі може перебувати цікавить його товар і чи є він в каталозі взагалі. Пошукова система, в деяких випадках, дозволяє значно скоротити кількість переходів між сторінками каталогу для доступу до інформації, що цікавить.

Призначена для користувача корзина являє собою деякий масив даних, який служить для зберігання замовленого користувачем товару.

Реєстраційна форма служить для введення персональних даних користувачів. Надалі ця інформація використовується для їх ідентифікації між сеансами роботи з інтернет-магазином. Дана інформація може зберігатися як на стороні сервера, так і на стороні клієнта.

Форма відправки замовлення служить для введення контактної інформації замовника і відправки її на електронну скриньку організації.

Реєстрація покупця виробляється або при оформленні замовлення, або при вході в магазин. Після вибору товару від покупця потрібно заповнити форму, в якій вказується, яким чином буде здійснено оплату і доставка. Для захисту персональної інформації взаємодія має здійснюватися по захищеному каналу. Після закінчення формування замовлення і реєстрації вся зібрана інформація про покупця надходить з електронної вітрини в торгову систему інтернет-магазину. У торговельній системі здійснюється перевірка наявності затребуваного товару на складі, ініціюється запит до платіжної системи. При відсутності товару на складі направляється запит постачальнику, а покупцю повідомляється про час затримки [4].

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

При можливості оплати через Інтернет, підключається платіжна система. Після повідомлення про проведення онлайн-платежу торговельною системою формується замовлення для служби доставки.

Веб-дизайн (англ. Web design, веб проектування) - різновид дизайну, який займається проектуванням призначених для користувача веб-інтерфейсів для сайтів і веб-додатків [2]. Поняття веб-дизайну включає в себе два аспекти: технічний і творчий. З технічної точки зору, веб-дизайн є проектуванням, що використовують верстку за допомогою гіпертекстової розмітки (HTML), мови опису зовнішнього вигляду документа (CSS), мов програмування (PHP, JavaScript, Ajax). З творчої точки зору веб-дизайн є художнім оформленням електронних ресурсів. Обидва ці аспекти є неподільним цілим, так як неможливо виконання веб-розробки без художнього оформлення, і навіть неможливо розробити функціональний дизайн сайту без проектування. Таким чином, веб-дизайн можна умовно назвати художнім конструюванням [3]. Якість створеного веб-дизайну визначається багатьма факторами, які необхідно враховувати при створенні сайту, тут важливі не тільки технічні, але також соціальні, психологічні та інші аспекти.

Підводячи підсумки дослідження, що стосується питань веб-дизайну, слід виділити ключові моменти, на які розробник, в першу чергу, повинен загострити свою увагу.

Структура сторінки:

• простота є основоположним принципом веб-дизайну;

• мінімум непотрібних елементів;

• наявність назви і пояснювальній фрази (слогану) для того, щоб користувач відразу розумів галузь діяльності веб-ресурсу;

• інформація і структура веб-ресурсу повинна максимально відповідати зазначеній сфері діяльності;

• веб-сайт повинен бути структурований таким чином, щоб мінімізувати зорові маршрути по екрану;

• важлива інформація і елементи меню повинні розташовуватися в лівій і лівої верхньої частинах сторінки, найменш важливі - в нижній і в нижній правій, змістовна частина - в центрі;

• стандартні елементи сайту слід розташовувати в областях сторінки, традиційно для того використовуваних: меню зазвичай розташовується зліва і / або під шапкою сайту; пошук розташовується в шапці сайту, під шапкою справа, рідше - внизу правої / лівої колонок; панель навігаційних іконок зазвичай розташована вгорі шапки праворуч або ліворуч; над або під основною частиною сторінки розташовується рядок навігації (вказівка рубрики, номера сторінки);

• пов'язана за змістом інформація і елементи сайту повинні розташовуватися поруч;

• однотипні дані на різних сторінках повинні розташовуватися в одній і тій же області;

• сторінка не повинна бути захаращена великою кількістю інформації, обов'язково повинні бути присутніми білі поля;

• об'єкти, які несуть порівняно самостійну, відмінну від інших інформацію, слід графічно розділити, що сприяє полегшенню сприйняття і запам'ятовування.

Кольори на сторінках:

• при виборі колірної схеми варто враховувати фірмовий стиль, тематику ресурсу і цільову аудиторію;

• краще використовувати невелику кількість квітів;

• не слід використовувати поєднання близьких кольорів для оформлення текстових блоків і фону;

• найбільш переважно використання безпечних кольорів.

Текст на сторінках магазину:

• по можливості слід використовувати найбільш простий шрифт з досить великим кеглем;

• інформація повинна бути максимально чіткою, досить короткою, і в доступній в формі, що забезпечить її правильне розуміння;

• заголовки і підзаголовки повинні бути інформативними і чіткими, а також відповідати позначається тексту;

• заголовок не повинен бути сильно відірваний від тексту, який розташовується під ним;

• для заголовків і підзаголовків переважно використання більш жирного накреслення;

• статті найкраще починати з анонсу, який буде володіти найбільшою інформативністю;

• ширина рядка документа в ідеальному випадку повинна вміщати близько 50-70 знаків;

• тексти повинні бути невеликого обсягу;

• великі блоки тексту краще розбивати на параграфи і абзаци;

• при необхідності текст потрібно розбивати по пунктам;

• найважливішу інформацію, ключові слова, цитати, посилання потрібно окремо виділяти в тексті;

• при великому кеглі слід коригувати занадто великі розриви між літерами за допомогою трекінгу;

• при великій щільності тексту необхідно зменшувати кількість слів у рядку за допомогою збільшення відстані між словами, коригувати щільність тексту можна за допомогою зміни інтерліньяжу;

• бажано, щоб текст був виконаний темним кольором на світлому фоні і був досить контрастним, не слід використовувати яскраві підкладки під тексти великого обсягу;

• слід уникати надмірного форматування тексту. Надмірні візуальні ефекти відволікають від змісту;

• найкращий порядок слів у реченні - прямий;

• слід уникати складних речень з довгим рядом послідовних підпорядкуванні;

• рекомендована довжина коротких повідомлень 7-11 значущих слів.

Шрифт тексту на сторінках:

• рекомендується ставити набір альтернативних шрифтів для відображення тексту при відсутності на комп'ютері користувача потрібного шрифту;

• не слід використовувати занадто багато гарнітур на сайті (як правило не більше трьох);

• при підборі сполучуваності шрифтів слід керуватися такими принципами: комбінування шрифтів сімейства Serif і Sans Serif; комбінування шрифтів, які були розроблені за схожими принципами (однакові геометричні форми); комбінування дуже різних шрифтів, що створюють протиставлення; не рекомендується комбінувати декоративні та / або рукописні шрифти;

• для виділення заголовка використовувати велике значення кегля, збільшення насиченості, зміна накреслення;

• потрібно враховувати відсутність у деяких шрифтів курсивного або жирного накреслення.

Графіка:

• графіка повинна бути максимально простою, а також відповідати тематиці ресурсу, створювати єдиний стиль оформлення сторінки, доповнювати і ілюструвати текст;

• не рекомендується використовувати велику кількість графіки;

• необхідно вказувати розмір картинок при верстці;

• не рекомендується масштабувати зображення засобами html;

• необхідно завжди вказувати альтернативний текст;

• фоновий малюнок краще вибирати нейтральний, щоб не втрачалася можливість вільного читання тексту;

• рекомендується використовувати навігаційні іконки;

• зображення завжди необхідно обробляти і оптимізувати;

• для маленьких навігаційних картинок використовувати формат PNG, для фотографій формат JPEG;

• рекомендується з великою обережністю використовувати анімацію (тільки для забезпечення інтерактивності);

• не слід включати анімаційні зображення в логотип;

• головне правило в питаннях впровадження графіки в веб-ресурс - завжди пам'ятати про закони авторського права!

Навігація:

• слід виділяти посилання кольором і підкресленням;

• рекомендується виділяти кольором відвідані посилання;

• кольору переглянутих і непереглянутих посилань повинні відрізнятися;

• при наведенні на область посилання курсор повинен змінюватися;

• не рекомендується використання підкреслення тексту, якщо він не є посиланням;

• при виборі назв розділів і категорій слід орієнтуватися на мову користувача і уникати професійного сленгу;

• навігаційні елементи повинні бути згруповані по будь-яким ознаками.

1.2. Призначення розробки та область застосування

Розроблений інтернет-магазин може бути використано при організації продажів смартфонів та мобільних телефонів. Також може бути використано як інформаційну систему, в якфй наведено технічні характеристики сучасних смартфонів та телефонів.

1.3. Підстава для розробки

Підставою для розробки кваліфікаційної роботи на тему «Розробка веб-орієнтованого програмного забезпечення для продажу смартфонів» є наказ ректора по Національному технічному університету «Дніпровська політехніка» від 07.06.2021р. № 317-с.

1.4. Постановка завдання

Розробити в кваліфікаційній роботі інтернет-магазин з продажу смартфонів та мобільних телефонів.

Були поставлені наступні вимоги та завдання:

– розробити інтернет-магазин з продажу смартфонфв та мобільних телефонів;

– розробити базу даних для інтернет-магазину;

– сайт повинен мати зручний інтуїтивно-зрозумілий інтерфейс;

– необхідно розробити функціонал для користувача та адміністратора;

– сайт має бути веб-орієнтованим та бути адаптований під всі види електронних пристроїв: смартфон, комп’ютер, планшет.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Інтернет-магазин з продажу смартфонфв та мобільних телефонів, що розробляється в кваліфікаційній роботі, повинен мати наступні функціональні можливості:

– ведення бази даних інтернет-магазину;

– сайт повинен мати зручний інтуїтивно-зрозумілий інтерфейс;

– зручний функціонал для користувача та адміністратора;

– сайт має бути веб-орієнтованим та бути адаптований під всі види електронних пристроїв: смартфон, комп’ютер, планшет.

1.5.2 Вимоги до інформаційної безпеки

Для уникнення некоректної роботи програми необхідно реалізувати:

– контроль вхідних даних;

– обробку виняткових ситуацій;

– виведення повідомлень про помилки;

– можливість повторного введення даних;

– можливість безперервної роботи протягом не менше 120 годин (5 діб);

– платформну незалежність;

– вірогідність виникнення не більше 2 логічних помилок на 1000 операторів за 1 рік експлуатації;

– забезпечення неушкодженого стану даних, що зберігаються в базі даних, у випадку відмови застосунку.

1.5.3 Вимоги до складу та параметрів технічних засобів

Програмний продукт не є вимогливий до складу та параметрів технічних засобів та може завантажуватись на ПК, ноутбуках, планшетах чи смартфонах різних типів та конфігурацій під управлінням різних ОС.

Для завантаження сторінки необхідно лише веб-браузер та доступ до мережі Інтернет.

Для нормального функціонування програми рекомендовано, наприклад, для ПК чи планшету, щоб обчислювальна машина, відповідала наступним вимогам:

– процесор класу IntelXeon з тактовою частотою не менш 2.4 ГГц;

– не менше 2 GB оперативної пам'яті;

– монітор з діагоналлю не менше 12";

– 2 Гб вільного місця на жорсткому диску;

– доступ до мережі Internet;

– клавіатура;

– маніпулятор «миша».

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

1.5.4 Вимоги до інформаційної та програмної сумісності

Програмний продукт не є вимогливий до інформаційної та програмної сумісності та може завантажуватись на ПК, ноутбуках, планшетах чи смартфонах різних типів та конфігурацій під управлінням різних ОС.

Для завантаження сторінки необхідно лише веб-браузер.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1. Функціональне призначення програми

Інтернет-магазин з продажу смартфонфв та мобільних телефонів, що розробляється в кваліфікаційній роботі, має наступні функціональні можливості:

- ведення бази даних інтернет-магазину;
- сайт має зручний інтуїтивно-зрозумілий інтерфейс;
- зручний функціонал для користувача та адміністратора;
- сайт ає веб-орієнтованим та адаптований під всі види електронних пристроїв: смартфон, комп’ютер, планшет.

2.2. Опис застосованих математичних методів

Оскільки особливості предметної галузі розв'язуваної задачі не передбачають застосування математичних методів, при розробці сайту інтернет-магазину, то математичні методи не використовувалися.

2.3. Опис використаної архітектури та шаблонів проектування

Model-View-Controller (MVC, «Модель-Представлення-Контролер», «Модель-Вид-Контролер») - схема поділу даних програми, інтерфейсу користувача і керуючої логіки на три окремих компоненти: модель, уявлення і контролер - таким чином, що модифікація кожного компонента може здійснюватися незалежно.

Модель (Model) надає дані і реагує на команди контролера, змінюючи свій стан.

Подання (View) відповідає за відображення даних моделі користувачеві, реагуючи на зміни моделі.

Контролер (Controller) інтерпретує дії користувача, сповіщаючи модель про необхідність змін. DOM (від англ. Document Object Model - «об'єктна модель документа») - це незалежний від платформи і мови програмний інтерфейс, що дозволяє програмам і скриптам отримати доступ до вмісту HTML-, XHTML- і XML-документів, а також його анулювання, структуру і оформлення таких документів.

Модель DOM не накладає обмежень на структуру документа. Будь-який документ відомої структури за допомогою DOM може бути представлений у вигляді дерева вузлів, кожен вузол якого є елементом, атрибут, текстовий, графічний або будь-який інший об'єкт. Вузли зв'язані між собою відносинами «батьківський-дочірній».

Спочатку різні браузери мали власні моделі документів (DOM), несумісні з іншими. Для забезпечення взаємної і зворотної сумісності фахівці міжнародного консорціуму W3C класифікували цю модель за рівнями, для кожного з яких була створена своя специфікація. Всі ці специфікації об'єднані в загальну групу, що носить назву W3C DOM.

Паттерн Front controller. Единая точка входа (англ. Front controller) – обеспечивает унифицированный интерфейс для интерфейсов в подсистеме. Front Controller определяет высокоуровневый интерфейс, упрощающий использование подсистемы.

В сложных веб-сайтах есть много одинаковых действий, которые надо производить во время обработки запросов. Это, например, контроль безопасности, многоязычность и настройка интерфейса пользователя. Когда поведение входного контроллера разбросано между несколькими объектами, дублируется большое количество кода. Помимо прочего возникают сложности смены поведения в реальном времени.

Паттерн Front Controller объединяет всю обработку запросов, пропуская запросы через единственный объект-обработчик. Этот объект содержит общую

логику поведения, которая может быть изменена в реальном времени при помощи декораторов. После обработки запроса контроллер обращается к конкретному объекту для отработки конкретного поведения.

2.4. Опис використаних технологій та мов програмування

Основою будь-якої web-сторінки є розмітка. Технології розмітки, такі як HTML, XHTML і XML, визначають структуру і можливе значення вмісту сторінки. Зовнішній же вид сторінки, насправді, повинен досягатися за допомогою таблиць стилів.

HTML (англ. HyperText Markup Language, мова гіпертекстової розмітки) є першорядної важливості технологією розмітки, яка застосовується на web-сторінках. Мова HTML інтерпретується браузером і відображає інформацію у вигляді документа, в зручній для людини формі.

Традиційний HTML існує в трьох основних версіях (HTML 2, HTML 3.2, HTML 4 і HTML 5). HTML 5 є найбільш сучасною і поки остаточною версією HTML [13].

Хоча багато теги і правила HTML досить добре визначені, більшість виробників браузерів надають розширення до цієї мови, що виходять за рамки опису стандартного узагальненого мови розмітки. Однак, незважаючи на це, HTML слід використовувати, перш за все, для структурування документа; функції HTML, пов'язані з форматуванням, в кінцевому підсумку, будуть повністю витіснені каскадними таблицями стилів (Cascading Style Sheets, CSS).

XHTML (eXtensible Markup Language, розширювана мова розмітки) - це нова редакція HTML, виконана за допомогою XML. XHTML дозволяє дві основні проблеми, пов'язані з HTML. По-перше, XHTML, приділяючи велику увагу застосуванню таблиць стилів, продовжує чинити тиск на дизайнерів, з тим, щоб вони відокремлювали зовнішній вигляд документа від його структури. По-друге, XHTML привносить набагато більш сувора вимога про дотримання правил розмітки web-сторінок.

Синтаксична строгість XHTML є одночасно його найбільшим перевагою і найгіршим недоліком. Правильно складеними сторінками простіше управляти і замінювати їх за допомогою програми, але людині їх створювати важче. Перехід на XHTML відбувається повільно саме через його суворості. Зайва негнучкість XHTML робить його менш зручним, ніж HTML, який набагато більш поблажливий по відношенню до новачків. Таким чином, поки не з'явиться більшу кількість інструментальних засобів, які виконують коректний код XHTML, ймовірно, в масштабах всього web-спільноти, мова буде прийматися так само повільно.

XML (англ. Extensible Markup Language, розширювана мова розмітки) розхвалює як революційна технологія розмітки, здатна змінити вигляд web-сторінок. Проте, не дивлячись на цю рекламу, лише деякі в точності розуміють, що насправді являє собою XML. Якщо коротко, XML є різновидом SGML (Standard Generalized Markup Language - стандартний узагальнений мову розмітки), модифікованої для Web; таким чином, він дозволяє розробникам задавати їх власну мову розмітки. Значить, за допомогою XML можна винайти YML (Your Markup Language, мову розмітки).

До сих пір негативний вплив винаходи занадто великої кількості індивідуальних мов на базі XML було обмежено, і більшість web-розробників згодні користуватися широко загальноприйнятими мовами начебто XHTML.

CSS (англ. Cascading Style Sheets, каскадні таблиці стилів) - мова опису зовнішнього вигляду документа, написаного з використанням мови розмітки. CSS використовується для опису параметрів оформлення розмітки електронного документа. Вони можуть задавати колір, шрифт, розмір, розташування об'єктів [12].

CSS вирішує проблему відділення вмісту документа (написаного на HTML) від його уявлення (заданого таблицями стилів). Подібне розділення збільшує доступність документа, надає більше можливостей для управління візуальним оформленням документа, зменшує складність структурного змісту. Завдяки таблиць стилів інтерпретація зовнішнього вигляду стає більш

універсальною: тепер можна окремо описати зовнішній вигляд документа для показу на пристроях з різними технічними можливостями, тим самим зберігши стабільність якості надання інформації.

Джерела таблиць стилів для відображення документа різні:

• авторські стилі (інформація стилів, що надається автором сторінки) у вигляді: локальних стилів: в HTML-документі інформація стилю для одного елемента вказується в його атрибуті <style>; вбудованих стилів-блоків CSS всередині самого HTML-документа, укладених в теги <style> </ style>; зовнішніх таблиць стилів - окремого CSS-файлу, на який дається посилання в документі;

• призначені для користувача стилі: локальний CSS-файл, вказаний користувачем в настройках браузера, переобумовленої авторські стилі, і застосовуваний до всіх документів;

• стиль браузера: стандартний стиль, який використовується браузером за замовчуванням для представлення елементів.

Для реалізації інтерактивних функцій на сайті необхідно використовувати, так звані, технології web-програмування. Технології web-програмування можна розділити на дві базисні групи: працюючі на стороні клієнта і працюють на стороні сервера. Технології на стороні клієнта запускаються на комп'ютері користувача сайту в контексті браузера. Програми, які запускаються на веб-сервері, відносяться до технологій програмування на стороні сервера.

Проблема, пов'язана з web-програмуванням, полягає в тому, щоб забезпечити точний вибір тієї технології, яка потрібна для виконання завдання. У кожної технології є свої переваги і недоліки. Зазвичай технології програмування на стороні клієнта і на стороні сервера мають характеристики, що роблять їх швидше поєднуються, ніж протистоять один одному. Наприклад, при додаванні на web-сайт форми для збору даних і їх збереження в базі даних, очевидно, має сенс перевіряти цю форму на стороні клієнта, щоб переконатися, що користувач ввів правильну інформацію, оскільки при цьому для перевірки

вхідних даних кругового звернення до сервера і назад не буде потрібно. Програмування на стороні клієнта зробить перевірку достовірності форми більш реактивної. З іншого боку, з приміщенням даних в базу краще впорається технологія на стороні сервера, з огляду на, що база даних знаходиться в серверній частині цього рівняння. Кожен основний тип програмування знаходиться на своєму місці, і їх суміш часто є кращим рішенням.

У наш час найбільш часто використовують програмування на стороні сервера, що реалізовується за допомогою мови PHP. Для написання програм на стороні клієнта, зазвичай застосовують мову JavaScript.

PHP (англ. Hypertext Preprocessor, Personal Home Page Tools (устар.), Препроцесор гіпертексту) - скриптова мова програмування загального призначення, інтенсивно застосовується для розробки веб-додатків [10].

PHP - це потужний багатоплатформовий набір засобів, який розташовується на сервері і призначається для обробки коду, вбудованого в html-документи. В даний час підтримується переважною більшістю хостинг-провайдеров.представляет собою мову з відкритим вихідним кодом для виконання на сервері сценаріїв, що створюють динамічні web-сторінки. Крім незалежності від браузерів, він пропонує просте і незалежне від платформи рішення для електронної комерції і складних web-додатків, в тому числі керованих базами даних [7].

У порівнянні з базовим HTML, що представляє собою систему з досить обмеженими можливостями, мова PHP має набагато більшу гнучкість і динамічністю. Він дозволяє додати більше індивідуальності та персоналізувати звичайні статичні HTML-сторінки. За допомогою PHP можна створювати привабливі оригінальні Web-сторінки на основі будь-яких задаються критеріїв (наприклад, часу доби або операційної системи користувача). На відміну від HTML, мова PHP також може взаємодіяти з базами даних і файлами, з його допомогою обробляється електронна пошта і виконуються багато інших операцій.

Структура PHP складається з ядра і підключаються модулей-"розширень", що представляють собою динамічні бібліотеки. Розширення дозволяють доповнити базові можливості мови, надаючи можливості для роботи з базами даних, сокетами, динамічною графікою, криптографічними бібліотеками, документами формату PDF, а також розробити і підключити своє власне розширення може будь-хто.

Синтаксис PHP для роботи програми не потрібно описувати будь-які змінні і використовувані модулі. Будь-яка програма може починатися безпосередньо з оператора PHP.исполняет код, що знаходиться всередині обмежувачів, таких як <? Php?>. В основному, це використовується для вставки PHP-коду в HTML-документ. Крім обмежувачів <? Php?>, Допускається використання

додаткових варіантів, таких як <? ? > I <script language = "php"> </ script>. Все, що знаходиться поза обмежувачів, виводиться без змін.

Імена змінних починаються з символу $, тип змінної оголошувати не потрібно. Імена змінних, функцій і класів з урахуванням регістру. Константи також чутливі до регістру.рассматрівает перехід на новий рядок як пробіл, так само як HTML і інші мови з вільним форматом. Інструкції поділяються за допомогою крапки з комою (;), за винятком деяких випадків, після оголошення конструкції if / else і циклів.

2.5. Опис структури програми та алгоритмів її функціонування

Опис структури бази даних

Таблиця - це деяка регулярна структура, що складається з кінцевого набору полів по вертикалі і однотипних записів по горизонталі. У реляційної моделі даних таблиця називається відношенням.

Атрибут - це інформаційне відображення властивостей об'єкта. Кожен об'єкт характеризується рядом основних атрибутів.

Первинний ключ (Primary Key) - це атрибут (або група атрибутів), які єдиним чином ідентифікують кожен рядок в таблиці.

Альтернативний (Foreign Key) ключ - це атрибут (або група атрибутів), неспівпадаючий з первинним ключем і унікально ідентифікує екземпляр об'єкта.

Наприклад, модель користувачі характеризується ідентифікатором, ім'ям, прізвищем, електронною поштою і т.д.

Таблиця 2.1.

Опис полів таблиці користувачів

Користувачі (users)			
№ п/п	**Поле**	**Примітка**	**Тип**
1.	Id	Ідентифікатор користувача	Number
2.	Name	Ім’я	Varchar
3.	Lastname	Прізвище	Varchar
4.	Email	Электронна пошта	Varchar
5.	password	Пароль	Varchar
6.	City	Місто	Varchar
7.	Role	Роль	Varchar

Таблиця 2.2.

Опис полів таблиці товарів

Товары (product)			
№ п/п	**Поле**	**Примітка**	**Тип**
1.	Id	Ідентифікатор товару	Number
2.	Name	Назва	Varchar
3.	Category_id	Ідентифікатор категорії	Number
4.	code	Код товару	Number

Товары (product)			
№ п/п	**Поле**	**Примітка**	**Тип**
5.	price	Ціна	Number
6.	availability	Доступність	Number
7.	brand	Виробник	Varchar
8.	description	Опис	Text
9.	is_new	Показчик новизни	Number
10.	Is_recommended	Показчик рекомендації	Number
11.	status	Статус	Number

Таблиця 2.3.

Опис полів таблиці замовлень

Заказы (product_order)			
№ п/п	**Поле**	**Примітка**	**Тип**
1.	Id	Ідентифікатор замовлень	Number
2.	user_name	Ім’я користувача	Varchar
3.	user_phone	Телефон користувача	Varchar
4.	user_comment	Коментар користувача	Text
5.	user_id	Ідентифікатор користувача	Number
6.	Date	Дата	Date
7.	products	Товари	Varchar
8.	status	Статус	Number

Таблиця 2.4.

Опис полів таблиці категорії

Категории (category)			
№ п/п	**Поле**	**Примітки**	**Тип**
1.	Id	Ідентифікатор категорії	Number
2.	Name	Ім’я категорії	Varchar
3.	sort_order	Сортування	Number
4.	status	Статус	Number

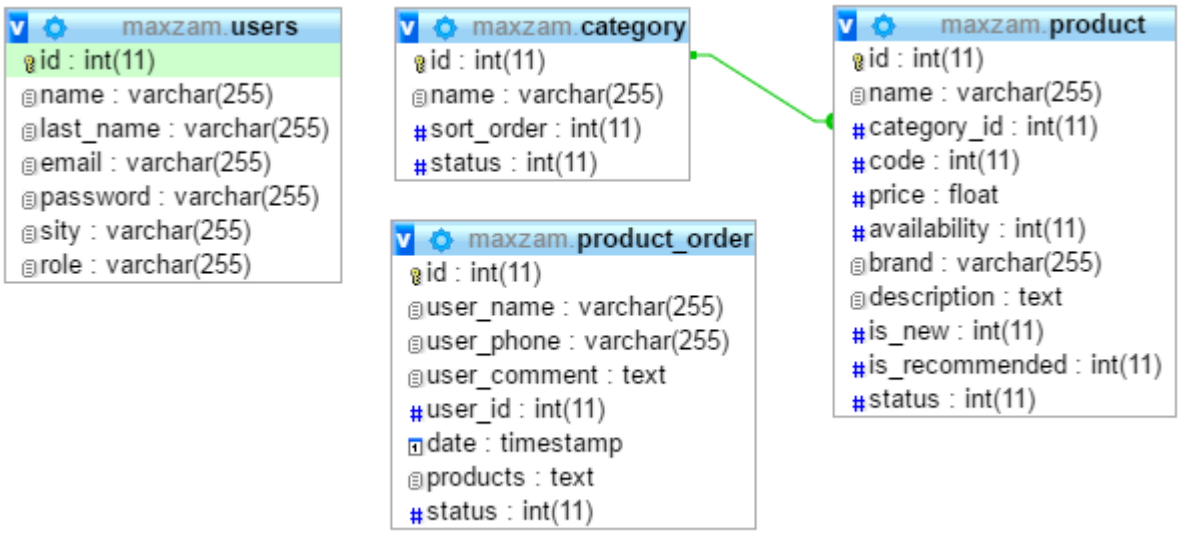

Рис. 2.1. ER-діаграма бази даних

Реляційна ER-діаграма на етапі фізичного проектуваня представлена на рисунку 2.1.

Оскільки було реалізовано архітектурний патерн MVC, то зберігання файлів буде такою, як на рисунку 2.2.

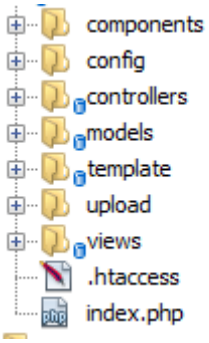

Рис. 2.2. Структура додатку

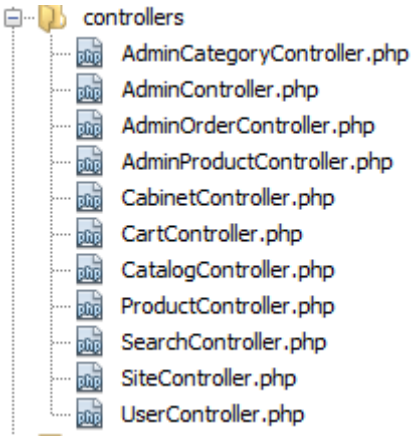

Рис. 2.3. Структура директории Controllers

Короткий опис кожної директорії:

• Components - зберігання головних компонентів додатка

• Config - зберігання налаштувань програми

• Controllers - зберігання котроллера додатки

• Models - зберігання моделей додатки

• Template - зберігання css і javascript файлів

• Upload - зберігання зображень товарів, завантажених з адмін панелі

• Views - зберігання відображення для додатка

Особливо цікавлять три основні директорії, які відносяться до MVC це - Controllers, Models, Views.

В директорії Controllers ми маємо 11 контролерів, головне правило іменування контролерів це назва самого контролера з великої літери і потім

приставка -Controller, наприклад, SiteController. На рисунку 2.3 представлені всі контролери додатку.

В директорії Models є 6 моделей, головне правило йменування моделей це назва модулів з великої літери. На рис. 2.4 представлені всі моделі програми.

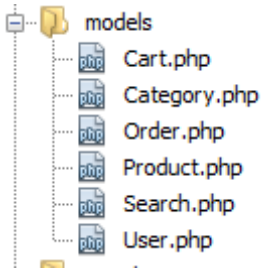

Рис. 2.4. Структура директорії Models

В директорії Views маємо 11 директорій в яких зберігаються окремі розділи сайту. На рис. 2.5 представлені всі подання додатка. У Views зберігаються html шаблони сторінок, логічно розташовані по папках, назви яких збігаються з ім'ям контролера, всі ці шаблони викликаються з контролера і наповнюються даними з моделі.

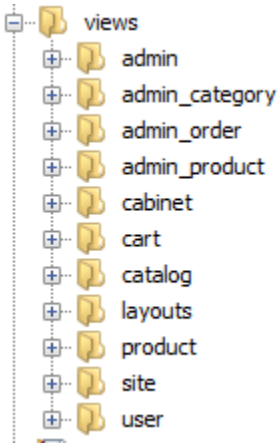

Рис. 2.5. Структура директории Views

Так само в проекті є дві важливі директорії це компоненти та конфігурація. У компонентах зберігаються файли прямо не пов'язані з архітектурі MVC, але без них невозмножно большенство дій. Наприклад, файл

Autoload.php завантажує класи в автоматичному режимі і немає потреби в кожному файлі писати конструкції підключення файлів типу require або include. Файл Db.php має подлючення до бази даних і бере дані з конфігурації, яка повертає параметри підключення до БД. Структура папок паказана на рисунку 2.6.

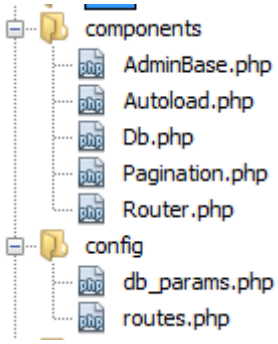

Рис. 2.6. Структура директорій components та config

Перше, що слід розглянути це цикл виконання програми. Спочатку користувач надсилає запит нашому додатку, наприклад він хоче отримати новини ресурсу, він вводить в рядку браузера http://maxzam.com/about і хоче отримати розділ про компанію. Цей запит приймає на себе FrontController і він створює об'єкт Router який парсит рядок запиту отримуючи важливі параметри. Далі роутер шукає по заданих в ньому шляхах назву контролера і дії які будуть виконувати запит. Роутер знаходить рядок « 'about' => 'site / about'» і дізнається, що запит потрібно відправити на SiteController і actionAbout (), який виводить сторінку з views та показується сраніцу користувачеві. На рис. 2.7. показаний цей цикл роботи для розділу новини.

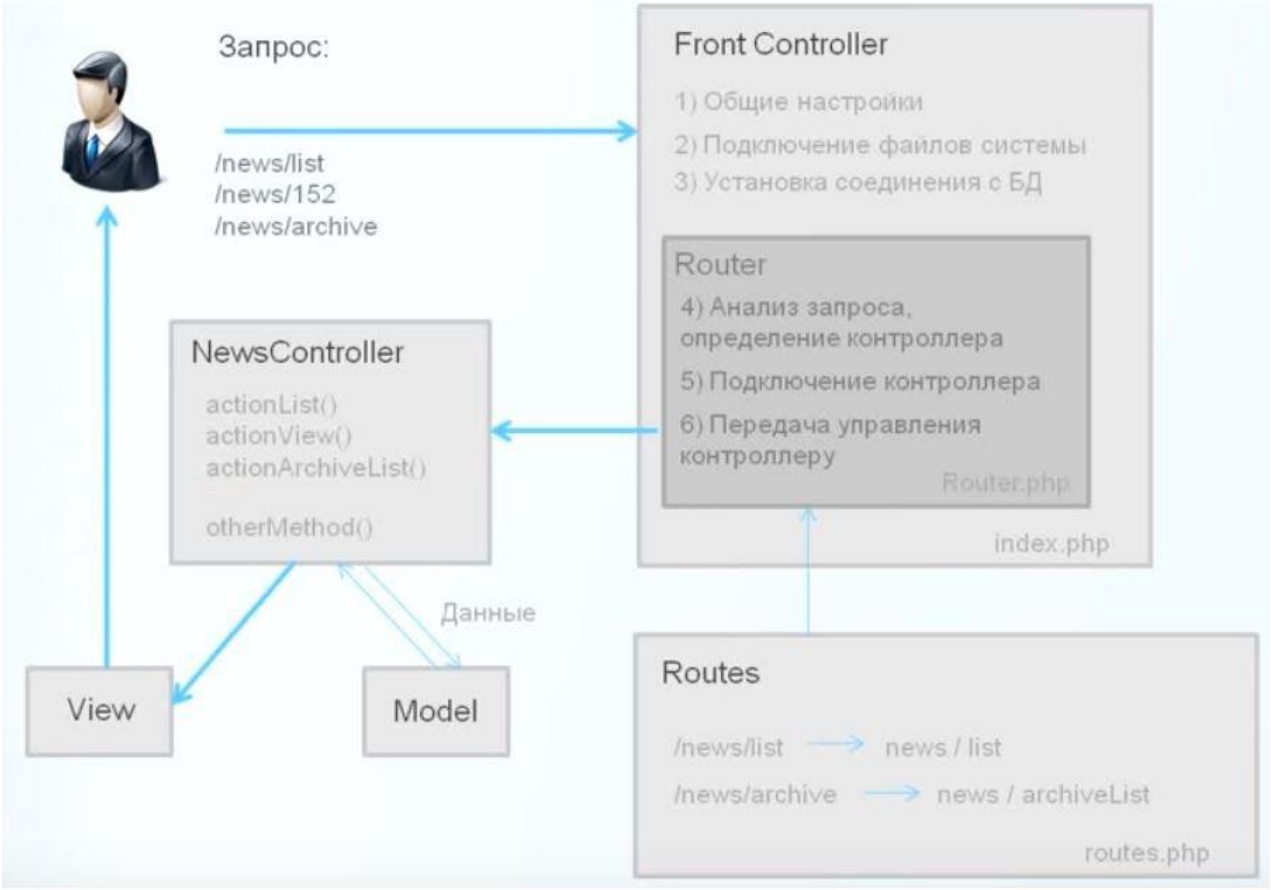

Рис. 2.7. Цикл роботи додатку для розділу новини

2.6. Обгрунтування та організація вхідних та вихідних даних програми

Вхідні дані:

- натискання на візуальні елементи управління застосунку;
- інформація, що вводиться користувачем із клавіатури.

Вихідні дані:

- інформація, що виводиться на екран.

2.7. Опис роботи розробленого програмного продукту

2.7.1. Використані технічні засоби

Програмний продукт не є вимогливий до складу та параметрів технічних засобів та може завантажуватись на ПК, ноутбуках, планшетах чи смартфонах різних типів та конфігурацій під управлінням різних ОС.

Для завантаження сторінки необхідно лише веб-браузер.

Програма створена в середовищі Sublime Text 3 на мові HTML з використанням бібліотеки Math.js.

Сайт розроблено на ноутбуці Lenovo IdeaPad 320-15IKB з такими характеристиками: екран 15.6" (1920x1080) Full HD, матовий / Intel Core i5-7200U (2.5 - 3.1 ГГц) / RAM 8 ГБ / HDD 1 ТБ + SSD 128 ГБ / nVidia GeForce GT 940MX, 2 ГБ / LAN / Wi-Fi / Bluetooth / веб-камера.

2.7.2. Використані програмні засоби

У якості середовища програмування було використано текстовий редактор Sublime Text 3, адаптований під синтаксис JavaScript та HTML для зручності.

Sublime Text - багатофункціональний текстовий редактор з широким набором зручних інструментів для виділення, маркування та обробки текстових фрагментів коду.

Інтерфейс цього редактора дуже лаконічний. Швидкість роботи і відгуку на всі дії на досить високому рівні. Підтримує величезну кількість мов (C ++, Dylan, Erlang, HTML, Haskell, Java, JavaScript, Lua, Markdown, MATLAB, Perl, PHP, Python, Ruby, SQL, XML та ін.) і пропонує на вибір близько 20 колірних схем. Вельми зручно, що реалізований повноекранний режим - дуже корисно, якщо не хочете, щоб користувача щось відволікало від повноцінної роботи. Крім того, в редакторі є так звана Мультіпанель: можна паралельно працювати з декількома файлами в одному вікні, що набагато зручніше, ніж використовувати кілька окремих вікон.

Хочеться виділити таку функцію, як мінікарта - в правій частині редактора розташована колонка, в якій в мініатюрі розташовані приблизно 5-6 екранів з текстом. Це дозволяє швидко переміщатися по коду і знаходити потрібні елементи.

Хто працює з кодом знає, як важлива в текстовому редакторі підсвічування. Так ось в додатку є можливість знайти і підсвітити найближчі парні дужки просто поставивши курсор всередину якої-небудь складної функції - насправді дуже зручно і практично.

Що ж стосується роботи безпосередньо з текстом, то і тут утиліта має багато переваг: підтримка сніпетів, автозавершення введення функцій, робота з макросами, зручний пошук, перевірка орфографії, мультівиделеніе, закладки потрібних місць, функція автозбереження і інше. Якщо і цього функціонала мало, то його легко розширити за рахунок плагінів.

Програма дуже гнучка в налаштуванні, практично всі опції редактора налаштовуються вручну в текстових файлах - одразу видно всі параметри і опис їх призначень. Для часто виконуваних операцій доступні комбінації гарячих клавіш. Загалом, переваги редактора гідно оцінять всі, хто так чи інакше пов'язаний з редагування коду або розмітки - програмісти, адміністратори і т.д.

2.7.3. Виклик та завантаження програми

Для завантаження сторінки необхідно лише веб-браузер.

Програма не є вимогливою до виклику та завантаженню і може завантажуватись на ПК, ноутбуках, планшетах чи смартфонах різних типів та конфігурацій під управлінням різних ОС.

2.7.4. Опис інтерфейсу користувача

При розробці сайту завжди необхідно враховувати апаратні можливості і особливості відтворення на екранах моніторів. Слід враховувати і те, що сучасний ринок обладнання досить широкий і різноманітність моделей, що випускаються моніторів має на увазі собою і різноманітність можливостей подання інформації на них. Відповідно до цього, в першу чергу, при створенні

макета дизайну сайту необхідно враховувати використання різного дозволу екрану користувачами.

Ще однією важливою характеристикою монітора є можливість передачі кольору. Сукупність усіх колірних відчуттів, які може відтворити пристрій, називають колірним охопленням цього пристрою. Сьогодні всі монітори відповідають стандарту sRGB. Діапазон кольорів sRGB дуже малий у порівнянні з видимим оком діапазоном, а тому багато кольору на етапі отримання зображення виявляються за його межами.

Тональність зображення на моніторі визначається колірною температурою. Чим нижче температура, тим тепліше кольору. Необхідність в колірній температурі виникає тому, що немає універсального білого кольору, який очей завжди б сприймав як білий. Залежно від умов, очей підлаштовується під певний колірний діапазон. Відтінок білого кольору на екрані монітора буде злегка змінюватися в залежності від зовнішнього освітлення, під яке підлаштовується і очей. Рекомендується встановлювати на екрані монітора таку колірну температуру, при якій білий колір на екрані не має будь-яких додаткових відтінків [5].

Для того, щоб звести до мінімуму всі можливі кольорові спотворення, виробляють калібрування монітора.

Види калібрування моніторів:

• програмна калібрування не вимагає колориметра і годиться лише на людський зір;

• апаратно-програмний метод (калібрації та характеризація монітора виконується колориметром, решта - відкритий);

• апаратне калібрування передбачає підключення колориметра до самого монітора.

Розроблюваний інтернет-магазин побутової техніки належить деякій комерційної організації і є однією зі складових власного бізнесу; його основне призначення полягає в інформаційному і маркетинговому просуванні товару, збільшення обсягу продажів, а також надання інформації про шуканої компанії.

З огляду на обґрунтовані раніше в дипломному проекті рекомендації та особливості, що стосуються питань сайтобудування і веб-дизайну, мною був створений оптимізований макет сайту, покликаний зайняти власну нішу в сфері онлайнової торгівлі.

Розробка починається з головної сторінки сайту. Ця сторінка має велике значення і виконує важливі завдання: допомагає побачити структуру ресурсу і його цінність для відвідувача (тобто елементи і інформацію, яку ресурс в себе включає), а також зорієнтувати користувача. Головна сторінка, найчастіше, визначає загальне оформлення ресурсу в цілому.

На початку роботи при розробці дизайну рекомендується створювати ескіз для того, щоб визначитися з кількістю колонок, елементів, їх взаиморасположением на сторінці. У нашому випадку веб-ресурс має наступні основні елементи: шапка, що включає в себе логотип компанії, верхнє меню, ліва і центральна колонки, підвал.

У верхньому меню розташовані посилання на розділи, які безпосередньо відносяться до функціонування інтернет-магазину: "Головна", "Каталог", "Про магазині", "Зворотній зв'язок".

Верхня права зона включає в себе функції управління кошиком і управління профілем користувача, а також інструмент пошуку в межах даного веб-ресурсу.

Зона зліва від контенту включає в себе каталог товару і форма підписки на новини даного інтернет ресурсу.

У навігаційних меню, як видно на малюнку 2.8, використовуються розділові вертикальні і горизонтальні смуги, що допомагають оку візуально відокремити пункти один від одного.

Розташування меню в загальноприйнятих областях дозволяє не оформляти пункти меню у вигляді посилань з підкресленням. В даному випадку працює динамічний стереотип, і користувач може зорієнтуватися без додаткових підказок.

Під шапкою сайту ми розмістили привітання для користувача, для того щоб вітати користувача і головна сторінка сайту завжди відрізнялася б від всіх інших сторінок даного ресурсу.

Центральна частина головної сторінки є вибіркою останнього доданого в магазин асортименту, яка надає не тільки основну інформацію про продукт (фотографія, назва, ціна), а й можливість швидкого переходу до потрібного товару.

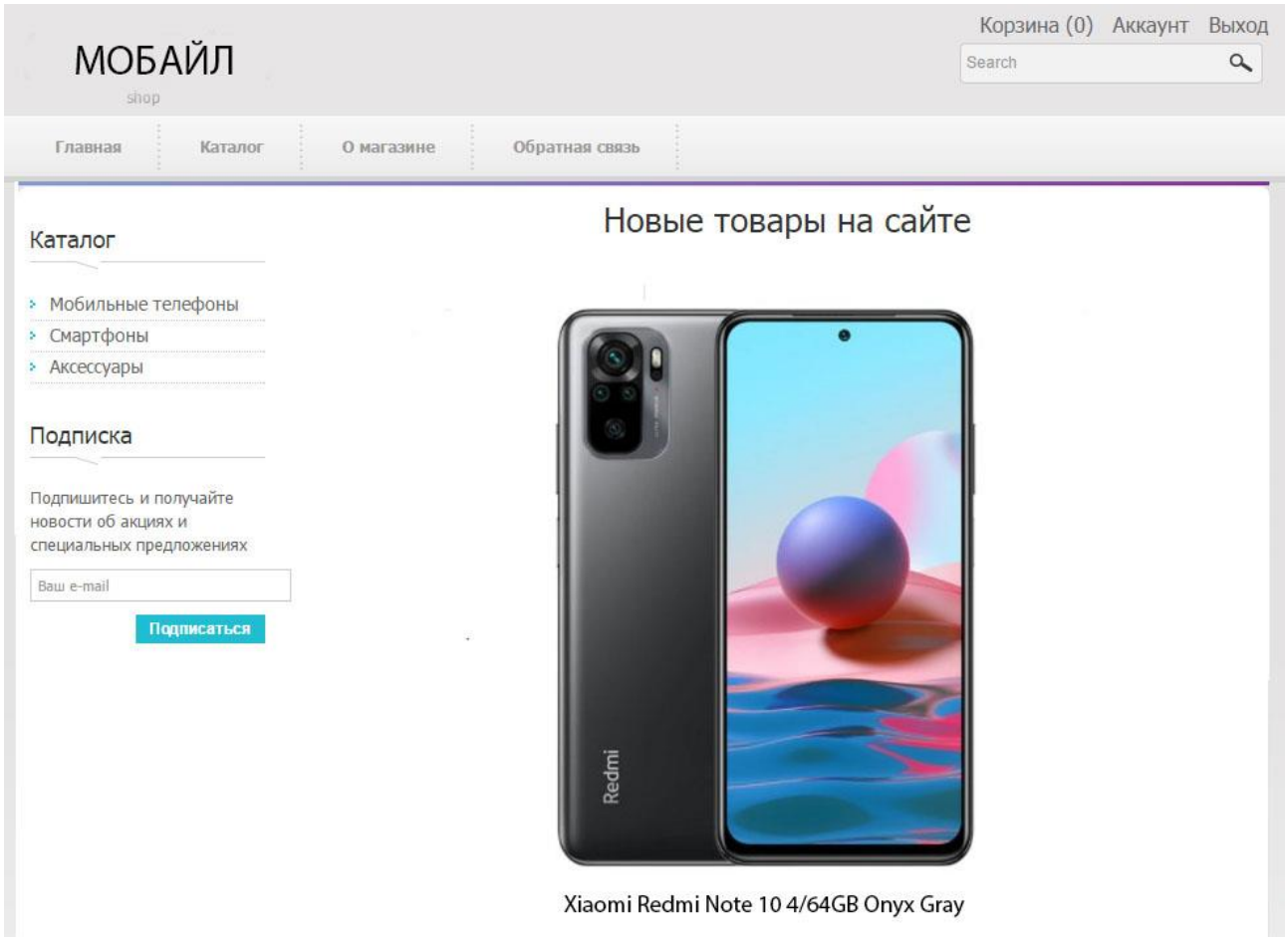

Рис. 2.8. Головна сторінка сайту

Колірна схема інтернет-магазину була вибрана стандартна: сірий текст на білому тлі. Подібний вибір пов'язаний з тим, що основна маса відвідувачів даного веб-ресурсу - чоловіки (рідше жінки) середнього і більш зрілого віку, зазвичай сімейні, чия основна мета - придбання продукції, а зовсім не оцінка художнього дизайну. Тому перед розробником, перш за все, стояло завдання

мінімізації можливих незручностей потенційного покупця за рахунок строгості і зрозумілості створюваного макета.

Грамотна структура каталогу продукції грає першорядне значення в роботі всього інтернет-магазину. Каталог знаходиться завжди зліва від контенту, що дозволяє швидко завжди перейти в потрібну категорію товару і вибрати потрібний товар. Надаючи користувачеві можливість швидкої і зручної навігації по товарному асортименту, а також полегшену форму замовлення, що продає організація залучає потенційних покупців і, як наслідок, набуває додаткові джерела доходу.

З огляду на вищесказане, саме каталогу (рис.2.9.) було приділено найбільшу увагу: пропонований асортимент розділений за видами продукції і брендам; кожен товар має візуальне (фотографія) і текстовий опис (короткий і розширене), а також інформацію про ціну і кнопку покупки.

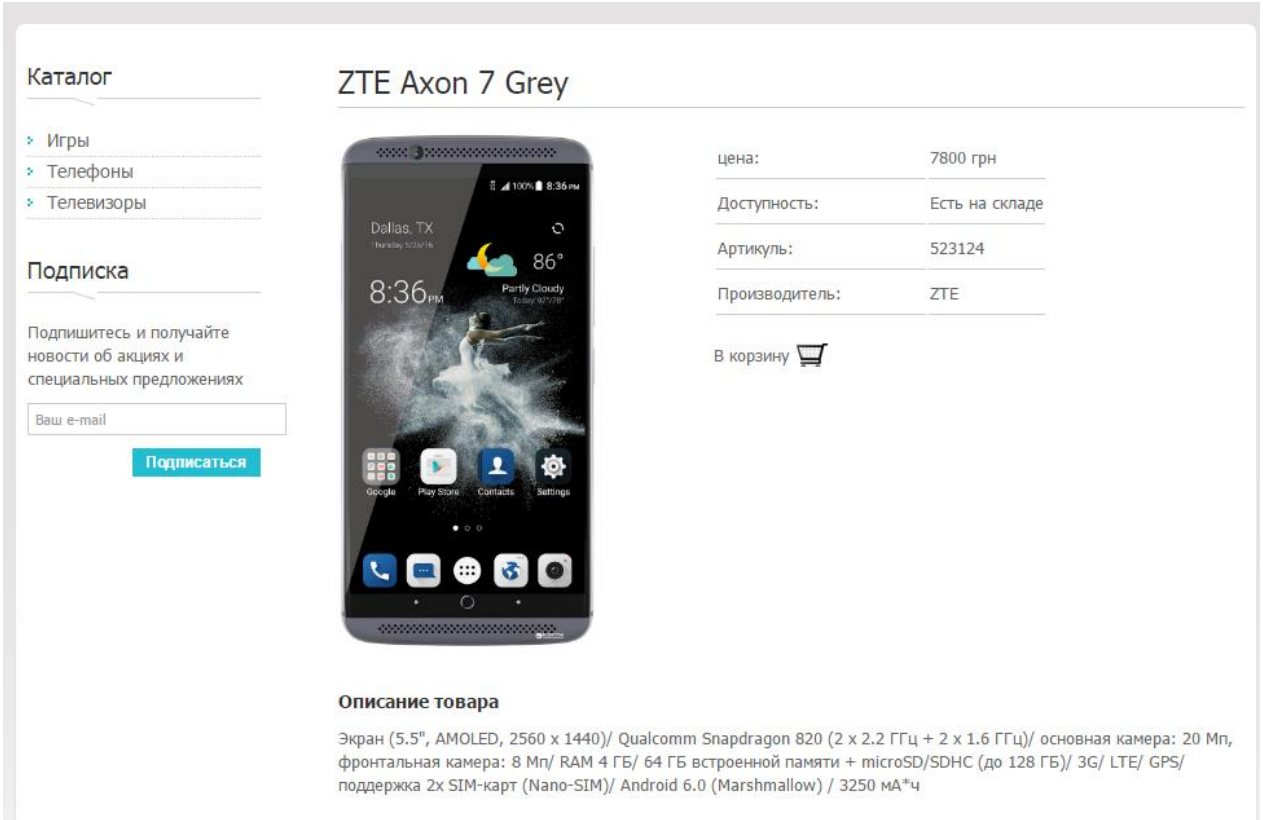

Рис. 2.9. Сторінка перегляду товару

Кошик (рис. 2.10) являє собою форму з інформацією про обрані товари (назва, кількість, індивідуальна і загальна ціни). Після натискання кнопки "Оформити", переходимо на форму заповнення персональний даних і після заповнення цього дані на замовлення відправляються на пошту клієнта і на сервер оператору інтернет-магазину.

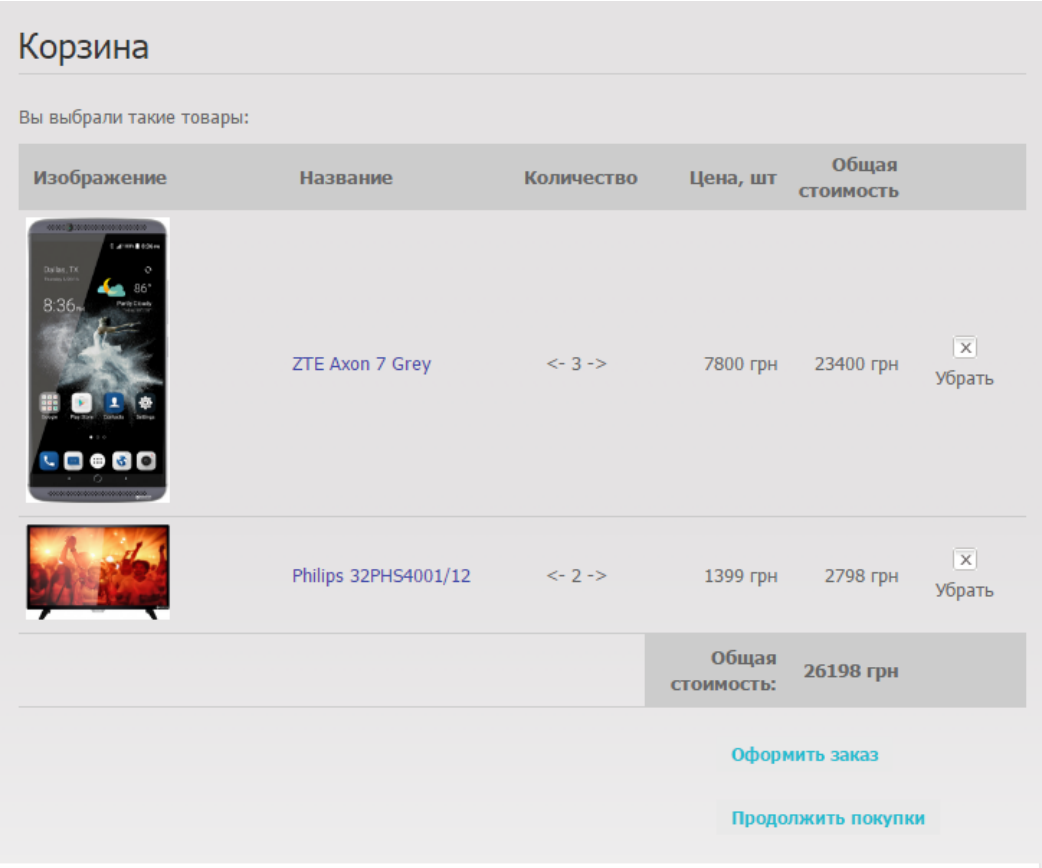

Рис. 2.10. Кошик покупця

Форма авторизації стандартна, представляє два поля для введення інформації, а саме email і пароль користувача (рис. 2.11).

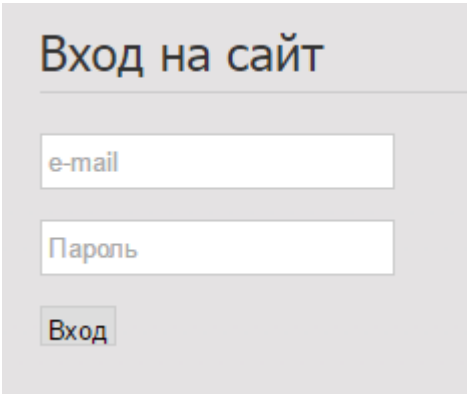

Рис. 2.11. Форма авторизації

Форма реєстрації подає складається з 5 полів введення інформації персональних даних користувача (рис. 2.12).

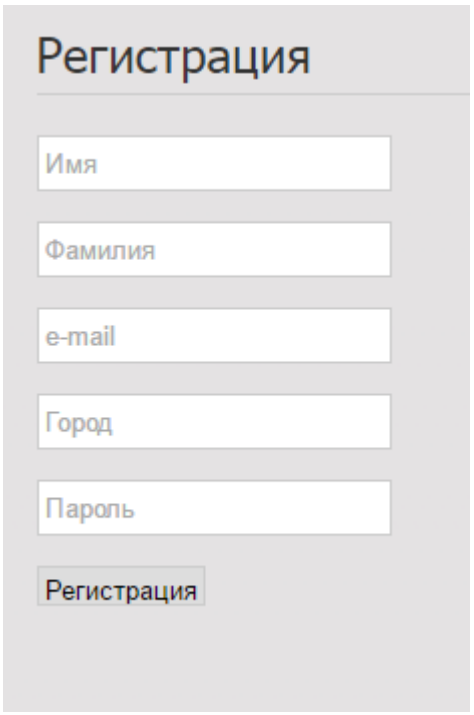

Рис. 2.12. Форма реєстрації

Форма зворотнього зв'язку зроблена для зв'язку з адміністратором для з'ясування будь-яких ситуацій, або для побажань або критики (рис. 2.13).

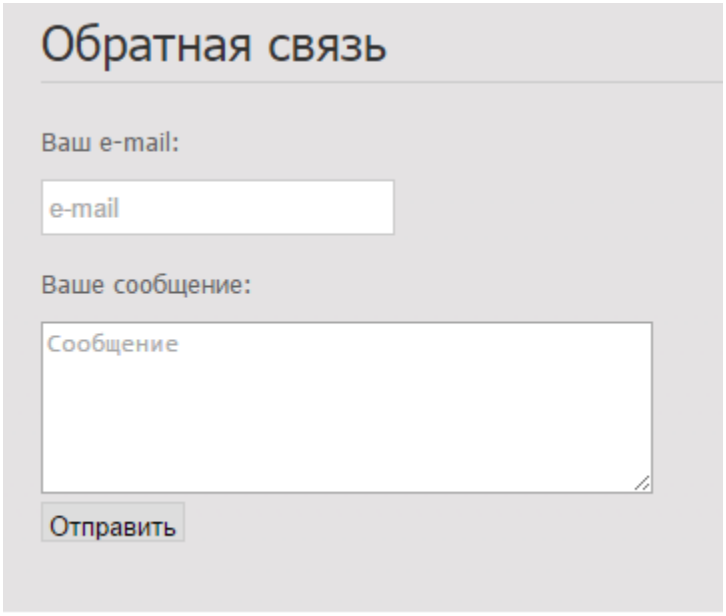

Рис. 2.13. Форма зворотнього зв'язку

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

Вихідні дані розробки програмного забезпечення:

1) передбачуване число операторів – 1500
2) коефіцієнт складності програми – 1,2
3) коефіцієнт кореляції програми в ході її розробки - 0,1
4) середня годинна заробітна плата програміста, грн/год - 90
5) вартість машино-години ЕОМ, грн/год – 7

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

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

Трудомісткість розробки ПЗ можна розрахувати за формулою:

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

де t_o – витрати праці на підготовку й опис поставленої задачі (приймається 50);

t_u – витрати праці на дослідження алгоритму рішення задачі;

t_a – витрати праці на розробку блок-схеми алгоритму;

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

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

$t_д$ – витрати праці на підготовку документації.

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

Умовне число операторів (підпрограм):

$$Q = q \cdot C(1 + p), \text{ де} \tag{3.2}$$

q– передбачуване число операторів;

C– коефіцієнт складності програми;

p – коефіцієнт кореляції програми в ході її розробки.

$$Q = 1500 \cdot 1{,}2 \cdot (1 + 0{,}1) = 1980;$$

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

$$t_u = \frac{Q \cdot B}{(75...85)K}, \text{ людино-годин,} \tag{3.3}$$

де B – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі, В=1.2 ... 1.5;

K – коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності. до 2 – 0,8;

$$t_u = \frac{1980 \cdot 1{,}2}{85 \cdot 0{,}8} = 35, \text{ людино-годин.}$$

Витрати праці на розробку алгоритму рішення задачі:

$$t_a = \frac{Q}{(20...25)K}; \tag{3.4}$$

$$t_a = \frac{1980}{20 \cdot 0{,}8} = 123{,}75 \text{, людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20...25)K}; \tag{3.5}$$

$$t_n = \frac{1980}{25 \cdot 0{,}8} = 99 \text{, людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

- за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4...5)K}; \tag{3.6}$$

$$t_{отл} = \frac{1980}{5 \cdot 0{,}8} = 495 \text{, людино-годин,}$$

- за умови комплексного налагодження завдання:

$$t_{отл}^{к} = 1{,}2 \cdot t_{отл}; \tag{3.7}$$

Витрати праці на підготовку документації:

$$t_{\partial} = t_{\partial р} + t_{\partial о}; \quad (3.8)$$

де $t_{\partial р}$ – трудомісткість підготовки матеріалів і рукопису

$$t_{\partial р} = \frac{Q}{(15...20)K}; \quad (3.9)$$

$$t_{\partial р} = \frac{1980}{20 \cdot 0{,}8} = 123{,}75, \text{ людино-годин.}$$

$t_{\partial о}$ – трудомісткість редагування, печатки й оформлення документації

$$t_{\partial о} = 0{,}75 \cdot t_{\partial р}; \quad (3.10)$$

$$t_{\partial о} = 0{,}75 \cdot 123{,}75 = 58, \text{ людино-годин.}$$

$$t_{\partial} = 123{,}75 + 58 = 181{,}75, \text{ людино-годин.}$$

Отримаємо трудомісткість розробки програмного забезпечення:

t = 50 +35 + 123,75 +99 + 495 +181,75 = 984,5, людино-годин.

У результаті ми розрахували, що в загальній складності необхідно 984,5 людино-годин для розробки даного програмного забезпечення.

3.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ Кпо включають витрати на заробітну плату виконавця програми Ззп і витрат машинного часу, необхідного на налагодження програми на ЕОМ.

$$К_{ПО} = З_{ЗП} + З_{МВ}\text{, грн,} \quad (3.11)$$

де $З_{ЗП}$ – заробітна плата виконавців, яка визначається за формулою:

$$З_{ЗП} = t \cdot С_{ПР}\text{, грн,} \quad (3.12)$$

де t – загальна трудомісткість, людино-годин;

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

$$З_{ЗП} = 984{,}5 \cdot 90 = 39380\text{,грн.}$$

$З_{МВ}$– Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{МВ} = t_{отл} \cdot С_{М}\text{, грн,} \quad (3.13)$$

де $t_{отл}$ – трудомісткість налагодження програми на ЕОМ, год.

$С_{МЧ}$ – вартість машино-години ЕОМ, грн/год.

$$З_{МВ} = 495 \cdot 7 = 3465\text{, грн.}$$

$$К_{ПО} = 39380 + 3465 = 42845\text{, грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p}\text{, мес.} \quad (3.14)$$

де B_k - число виконавців;

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

$$T = \frac{984{,}5}{1 \cdot 176} = 5{,}6 \text{ міс.}$$

Висновок: програмне забезпечення призначене для роботи інтернет-магазину з продажів смартфонів та мобільних телефонів. Час розробки даного програмного забезпечення складає 984,5 людино-годин. Таким чином, очікувана тривалість розробки складе 5,6 місяця при 40 годинному робочому тижні (місячний фонд робочого часу 176 годин), а витрати на створення програмного забезпечення 42845 грн.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи був реалізований інтернет-магазин з продажу смартфонів та мобільних телефонів на мові програмування PHP, також були використані два патерни проектування: MVC і FrontController.

В ході виконання роботи була створена база даних під зберігання даних користувачів, даних про товари, даних про категорії і дані про замовлення з цього магазину. Розроблено макет на мові розмітки html і впроваджений в проект інтернет-магазину.

В результаті виконання кваліфікаційної роботи був створений інтернет-магазин, в якому реалізовані основні можливості інтернет-магазину, а саме: реєстрація, авторизація, корзина, зворотний зв'язок, каталог товарів, пошук за ключовим словом, оформлення замовлення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Алексунин В.А., Родигина В.В. Электронная коммерция и маркетинг в Интернете. Учебное пособие. – М.: "Дашков и К^{O}", 2005. – 255 с.
2. Гото Г., Котлер Э. Web-дизайн. – СПб: Символ Плюс, 2009. – 340 с.
3. Дж.Н. Роббинс. Web-дизайн. Справочник. – М.: Кудиц-Пресс, 2008. – 170 с.
4. Дж.Н. Роббинс. Разработка приложений для электронной коммерции. - СПб: Питер Бук, 2001. – 233 с.
5. Леонтьев А.А. Web-дизайн. Руководство пользователя. - М.: Центр, 2000. – 126 с.
6. Орлов Л. Как создать электронный магазин в Интернет, 2-е изд., М.: Бук. пресс, 2006. - 384 с.
7. А. Рубен, А. Горев, С. Макшарипов. Эффективная работа с СУБД. - СПб.: Питер, 2009. – 822 с.
8. Business-To-Consumer (Электрон. ресурс). – Способ доступа: URL: http://www.ru.wikipedia.org/wiki/B2C.
9. MySQL (Электрон. ресурс). – Способ доступа: URL: http://ru.wikipedia.org/wiki/MySQL.
10. PHP язык программирования (Электрон. ресурс). – Способ доступа: URL: http://www.ru.wikipedia.org/wiki/MySQL.
11. Web-сервер (Электрон. ресурс). – Способ доступа: URL: http://ru.wikipedia.org/wiki/Веб-сервер.
12. Введение в CSS (Электрон. ресурс). – Способ доступа: URL: http://htmlbook.ru/samcss/vvedenie-v-css.
13. Введение в HTML (Электрон. ресурс). – Способ доступа: URL: http://htmlbook.ru/samhtml/vvedenie-v-html.
14. Графика для Web (Электрон. ресурс). – Способ доступа: URL: http://www.webimg.ru.

15. Интернет и украинский рынок электронной коммерции (Электрон. ресурс). – Способ доступа: URL: www.crime-research.org

16. Понятие и функции интернет-магазина (Электрон. ресурс). – Способ доступа: URL: http://www.ecomrus.ru.

17. Проблемы и решения в Web-дизайне (Электрон. ресурс). – Способ доступа: URL: http://www.rotorweb.ru.

18. Web Database Application with PHP and MySQL, 2nd Edition By David Lane, Hugh E. Williams. © O'Reilly, May 2004. – 401 с.

19. О системах управления сайтом (Электрон. ресурс). – Способ доступа: URL: http://ru.wikipedia.org/wiki/CMS

20. Content management system (Электрон. ресурс). – Способ доступа: URL: http://www.brutto.ru/informacija/uznat-bolshe/content-management-system

21. CMS обзор: CMS, движок сайта, система управления сайтом, mambo, php nuke (Электрон. ресурс). – Способ доступа: URL: http://cmsobzor.ru/news.php

22. Каратыгин С., Тихонов А., Долголаптев В. Базы данных: простейшие средства обработки информации, электронные таблицы, системы управления базами данных: В 2 т. - М.: АВР, 1995. – 156 с.

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

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

ДОДАТОК А

КОД ПРОГРАМИ

```
CREATE TABLE IF NOT EXISTS `category` (
  `id` int(11) NOT NULL,
  `name` varchar(255) NOT NULL,
  `sort_order` int(11) NOT NULL DEFAULT '0',
  `status` int(11) NOT NULL DEFAULT '1'
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- --------------------------------------------------------

--
-- Структура таблицы `product`
--

CREATE TABLE IF NOT EXISTS `product` (
  `id` int(11) NOT NULL,
  `name` varchar(255) NOT NULL,
  `category_id` int(11) NOT NULL,
  `code` int(11) NOT NULL,
  `price` float NOT NULL,
  `availability` int(11) NOT NULL,
  `brand` varchar(255) NOT NULL,
  `description` text NOT NULL,
  `is_new` int(11) NOT NULL DEFAULT '0',
  `is_recommended` int(11) NOT NULL DEFAULT '0',
  `status` int(11) NOT NULL DEFAULT '1'
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- --------------------------------------------------------

--
-- Структура таблицы `product_order`
--

CREATE TABLE IF NOT EXISTS `product_order` (
  `id` int(11) NOT NULL,
  `user_name` varchar(255) NOT NULL,
  `user_phone` varchar(255) NOT NULL,
  `user_comment` text NOT NULL,
  `user_id` int(11) DEFAULT NULL,
  `date` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `products` text NOT NULL,
  `status` int(11) NOT NULL DEFAULT '1'
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- --------------------------------------------------------

--
-- Структура таблицы `users`
--

CREATE TABLE IF NOT EXISTS `users` (
  `id` int(11) NOT NULL,
  `name` varchar(255) NOT NULL,
  `last_name` varchar(255) NOT NULL,
  `email` varchar(255) NOT NULL,
  `password` varchar(255) NOT NULL,
```

```
  `sity` varchar(255) NOT NULL DEFAULT 'неизвесно',
  `role` varchar(255) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- Индексы сохранённых таблиц
--

--
-- Индексы таблицы `category`
--
ALTER TABLE `category`
  ADD PRIMARY KEY (`id`),
  ADD KEY `id` (`id`);

--
-- Индексы таблицы `product`
--
ALTER TABLE `product`
  ADD PRIMARY KEY (`id`),
  ADD KEY `category_id` (`category_id`),
  ADD KEY `category_id_2` (`category_id`);

--
-- Индексы таблицы `product_order`
--
ALTER TABLE `product_order`
  ADD PRIMARY KEY (`id`),
  ADD KEY `user_id` (`user_id`),
  ADD KEY `user_id_2` (`user_id`);

--
-- Индексы таблицы `users`
--
ALTER TABLE `users`
  ADD PRIMARY KEY (`id`);

--
-- AUTO_INCREMENT для сохранённых таблиц
--

--
-- AUTO_INCREMENT для таблицы `category`
--
ALTER TABLE `category`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;
--
-- AUTO_INCREMENT для таблицы `product`
--
ALTER TABLE `product`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;
--
-- AUTO_INCREMENT для таблицы `product_order`
--
ALTER TABLE `product_order`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;
--
-- AUTO_INCREMENT для таблицы `users`
--
ALTER TABLE `users`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;
--
-- Ограничения внешнего ключа сохраненных таблиц
```

Файл /components /Router.php :

```
<?php

class Router
{
	private $routes;

	public function __construct()
	{
		$routesPath = ROOT.'/config/routes.php';
		$this->routes = include($routesPath);
	}

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

	public function run()
	{
		$uri = $this->getURI();

		foreach ($this->routes as $uriPattern => $path) {

			if(preg_match("~$uriPattern~", $uri)) {

				$internalRoute = preg_replace("~$uriPattern~", $path, $uri);

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

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

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

				$parameters = $segments;

				$controllerFile = ROOT . '/controllers/' .$controllerName. '.php';
				if (file_exists($controllerFile)) {
					include_once($controllerFile);
				}

				$controllerObject = new $controllerName;
				$result = call_user_func_array(array($controllerObject, $actionName),
$parameters);

				if ($result != null) {
					break;
				}
			}

		}
	}
}
```

Файл /components /Db.php :

```
<?php
```

```
class Db
{

                public static function getConnection()
                {
                        $paramsPath = ROOT . '/config/db_params.php';
                        $params = include($paramsPath);
                        $dsn = "mysql:host={$params['host']};dbname={$params['dbname']}";
                        $db = new PDO($dsn, $params['user'], $params['password']);
                $db->exec("set names utf8");

                        return $db;
                }
}
```

Файл /components /Autoload.php :

```
<?php

function __autoload($class_name)
{
   // Массив папок, в которых могут находиться необходимые классы
   $array_paths = array(
      '/models/',
      '/components/',
      '/controllers/',
   );
   // Проходим по массиву папок
   foreach ($array_paths as $path) {
      // Формируем имя и путь к файлу с классом
      $path = ROOT . $path . $class_name . '.php';
      // Если такой файл существует, подключаем его
      if (is_file($path)) {
         include_once $path;
      }
   }
}
```

Файл /components/AdminBase.php :

```
<?php

abstract class AdminBase {

   public static function checkAdmin(){

      $userId = User::checkLogged();

      $user = User::getUserById($userId);

      if($user['role'] == 'admin'){
         return true;
      }
      die('Доступ закрыт');
   }
}
```

Файл /config/db_params.php :

```
<?php

return array(
                        'host' => 'localhost',
                        'dbname' => 'maxzam',
                        'user' => 'root',
```

```
                                'password' => '',
);
```

Файл /config/routes.php :

```
<?php
return array(

        'product/([0-9]+)' => 'product/view/$1',

    'catalog' => 'catalog/index',

    'category/([0-9]+)/page-([0-9]+)' => 'catalog/category/$1/$2',
    'category/([0-9]+)' => 'catalog/category/$1',

    'user/register' => 'user/register',
    'user/login' => 'user/login',
    'user/logout' => 'user/logout',

    'cabinet/history/([0-9]+)' => 'cabinet/historyId/$1',
    'cabinet/history' => 'cabinet/history',
    'cabinet/edit' => 'cabinet/edit',
    'cabinet' => 'cabinet/index',

    'cart/checkout' => 'cart/checkout',
    'cart/delete/([0-9]+)' => 'cart/delete/$1',
    'cart/add/([0-9]+)' => 'cart/add/$1',
    'cart/upper/([0-9]+)' => 'cart/upper/$1',
    'cart/downper/([0-9]+)' => 'cart/downper/$1',
    'cart' => 'cart/index',

    'admin/product/create'=> 'adminProduct/create',
    'admin/product/update/([0-9]+)'=> 'adminProduct/update/$1',
    'admin/product/delete/([0-9]+)'=> 'adminProduct/delete/$1',
    'admin/product'=> 'adminProduct/index',

    'admin/category/create'=> 'adminCategory/create',
    'admin/category/update/([0-9]+)'=> 'adminCategory/update/$1',
    'admin/category/delete/([0-9]+)'=> 'adminCategory/delete/$1',
    'admin/category'=> 'adminCategory/index',


    'admin/order/update/([0-9]+)'=> 'adminOrder/update/$1',
    'admin/order/delete/([0-9]+)'=> 'adminOrder/delete/$1',
    'admin/order/view/([0-9]+)'=> 'adminOrder/view/$1',
    'admin/order'=> 'adminOrder/index',

    'admin'=> 'admin/index',

    'search' => 'search/search',
    'about' => 'site/about',
    'contacts' => 'site/contact',
        '' => 'site/index',


        );
```

Файл / controllers/CabinetController.php :

```
<?php

class CabinetController {

   public function actionIndex(){
```

```
    $userId = User::checkLogged();

    $user = User::getUserById($userId);

    require_once (ROOT.'/views/cabinet/index.php');
    return true;
}

public function actionEdit(){

    $userId = User::checkLogged();
    $user = User::getUserById($userId);
    $name = $user['name'];

    $password = $user['password'];
    $last_name = $user['last_name'];
    $sity = $user['sity'];

    $result = false;


    if (isset($_POST['submit'])) {
        $name = $_POST['name'];
        $last_name = $_POST['last_name'];
        $password = $_POST['password'];
        $sity = $_POST['sity'];

        $errors = false;

        if (!User::checkName($name)) {
           $errors[] = 'Имя не должно быть короче 3 символов';
        }
        if (!User::checkName($last_name)) {
           $errors[] = 'Фамилия не должна быть короче 3 символов';
        }

         if (!User::checkPassword($password)) {
           $errors[] = 'Пароль слишком короткий, он должен быть не меньше 6 символов';
        }

        if($errors == false){
           $result = User::edit($userId, $name,$last_name,$sity, $password);
        }
    }
    require_once '/views/cabinet/edit.php';

    return true;

}

 public function actionHistory(){

      $userId = User::checkLogged();
      $ordersList = Order::getOrderByUserId($userId);
      require_once '/views/cabinet/history.php';

    return true;

 }

 public function actionHistoryId($id){
```

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

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

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

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

        require_once '/views/cabinet/historyView.php';

        return true;

    }
}
```

Файл / controllers/CartController.php :

```
<?php


class CartController {

    public function actionAdd($id){

        Cart::addProduct($id);

        $referrer = $_SERVER['HTTP_REFERER'];
        header("Location: {$referrer}");

    }

    public function actionIndex(){
        $categories = array();
        $categories = Category::getCategoriesList();

        $productsInCart = false;

        $productsInCart = Cart::getProducts();

        if($productsInCart){
            $productIds = array_keys($productsInCart);

            $products = Product::getProductByIds($productIds);

            $totalPrice = Cart::getTotalPrice($products);
        }
        require_once (ROOT.'/views/cart/index.php');
        return true;
    }

     public function actionCheckout(){

        $categories = array();
        $categories = Category::getCategoriesList();
        $result = false;

        if(isset($_POST['submit'])){


            $userName = $_POST['userName'];
```

```
    $userPhone = $_POST['userPhone'];
    $userComment = $_POST['userComment'];

    $errors = false;

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

       $productsInCart = Cart::getProducts();
       if (User::isGuest()){
           $userId = 0;
       } else {
           $userId = User::checkLogged();
       }


       $result = Order::save($userName,$userPhone,$userComment,$userId,$productsInCart);

       if ($result) {
           $adminEmail = 'max_dev@inbox.ru';
           $message = 'новый заказ';
           $subject = 'новый заказ';
           mail($adminEmail,$message,$subject );

           Cart::clear();
       }
     }else {
         $productsInCart = Cart::getProducts();
         $productIds = array_keys($productsInCart);
         $products = Product::getProductByIds($productIds);
         $totalPrice = Cart::getTotalPrice($products);
         $totalQuantity = Cart::countItems();

     }

} else {

  $productsInCart = Cart::getProducts();
    if($productsInCart == false){

        header("Location: /");
    } else {
        $productIds = array_keys($productsInCart);
        $products = Product::getProductByIds($productIds);
        $totalPrice = Cart::getTotalPrice($products);
        $totalQuantity = Cart::countItems();

         $userName = false;
         $userPhone = false;
         $userComment = false;

         if (User::isGuest()) {

         } else {

             $userId = User::checkLogged();
             $user = User::getUserById($userId);
             $userName = $user['name'];
         }
    }
```

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

    public function actionDelete($id){

        Cart::deleteProduct($id);


        $referrer = $_SERVER['HTTP_REFERER'];
        header("Location: {$referrer}");
    }

     public function actionUpper($id){

        Cart::upperCart($id);

        $referrer = $_SERVER['HTTP_REFERER'];
        header("Location: {$referrer}");

   }

    public function actionDownper($id){

        Cart::downperCart($id);

        $referrer = $_SERVER['HTTP_REFERER'];
        header("Location: {$referrer}");

   }
}
```

Файл / controllers/CatalogController.php :

```
<?php

class CatalogController {

    public function actionIndex(){

        $categories = array();
        $categories = Category::getCategoriesList();

        $latestProducts = array();
        $latestProducts = Product::getLatestProducts();

        $recommendedList = array();
        $recommendedList = Product::getRecomendedProducts();


        require_once (ROOT.'/views/catalog/index.php');

        return true;
   }

    public function actionCategory($categoryId, $page = 1){

        $categories = array();
        $categories = Category::getCategoriesList();

         $categoryProducts = array();
         $categoryProducts = Product::getProductsListByCategory($categoryId,$page);
```

```
        $recommendedList = array();
        $recommendedList = Product::getRecomendedProducts();

         $total = Product::getTotalProductsInCategory($categoryId);
         $pagination = new Pagination($total,$page, Product::SHOW_BY_DEFAULT,'page-');

        require_once (ROOT.'/views/catalog/category.php');

        return true;
    }
}
```

Файл / controllers/ProductController.php :

```
<?php

class ProductController {

     public function actionView($productId){

        $categories = array();
        $categories = Category::getCategoriesList();

        $product = Product::getProductById($productId);
        require_once (ROOT.'/views/product/view.php');

        return true;
    }
}
```

Файл / controllers/SearchController.php :

```
<?php

class SearchController {
    public function actionSearch()
    {
        $categories = Category::getCategoriesList();

        if (isset($_POST['submit'])) {
            $search = $_POST['search'];
            $list = Search::getProductByName($search);

        }

        require_once (ROOT.'/views/site/search.php');
        return true;
    }
}
```

Файл / controllers/SiteController.php :

```
<?php

class SiteController {

    public function actionIndex(){

        $categories = array();
        $categories = Category::getCategoriesList();

        $latestProducts = array();
        $latestProducts = Product::getLatestProducts(8);

        $recommendedList = array();
        $recommendedList = Product::getRecomendedProducts();
```

```
        require_once (ROOT.'/views/site/index.php');

        return true;
    }

    public function actionContact(){

        $userMail = '';
        $userText = '';
        $result = false;
        if (isset($_POST['submit'])) {

            $userMail = $_POST['userMail'];
            $userText = $_POST['userText'];

            $errors= false;

            if (!User::checkEmail($userMail)) {
                $errors[] = 'email не ok';
            }

            if($errors == false){
                $adminEmail = 'max_dev@inbox.ru';
                $subject='Тема письма';
                $message="Текст: {$userText}. от {$userMail} ";
                $result = mail($adminEmail,$subject,$message);
                $result = true;
            }
        }
        require_once (ROOT.'/views/site/contact.php');
       return true;
    }
       public function actionAbout(){
           require_once (ROOT.'/views/site/about.php');
           return true;
       }
}
```

Файл / controllers/UserController.php :

```
<?php


class UserController {

    public function actionRegister(){
        $categories = array();
        $categories = Category::getCategoriesList();

        $name = '';
        $last_name = '';
        $email = '';
        $password = '';
        $sity = '';
        $result = false;


        if (isset($_POST['submit'])) {
            $name = $_POST['name'];
            $email = $_POST['email'];
            $password = $_POST['password'];
            $last_name = $_POST['last_name'];
            $sity = $_POST['sity'];
            $errors = false;
```

```
        if (!User::checkName($name)) {
            $errors[] = 'Имя не должно быть короче 3 символов';
        }
        if (!User::checkName($last_name)) {
            $errors[] = 'Фамилия не должна быть короче 3 символов';
        }
         if (!User::checkEmail($email)) {
            $errors[] = 'Неправильный e-mail';
        }

         if (!User::checkPassword($password)) {
            $errors[] = 'Пароль слишком короткий, он должен быть не меньше 6 символов';
        }

         if (User::checkEmailExists($email)) {
            $errors[] = 'такой email уже существует';
        }

        if($errors == false){
            $result = User::register($name,$last_name, $email,$sity, $password);
            if($result){
                User::auth($result);

            }

        }
    }
    require_once ROOT.'/views/user/register.php';

    return true;

}

public function actionLogin(){
    $categories = array();
    $categories = Category::getCategoriesList();


    $email = '';
    $password = '';


    if (isset($_POST['submit'])) {

        $email = $_POST['email'];
        $password = $_POST['password'];

        $errors = false;

         if (!User::checkEmail($email)) {
            $errors[] = 'email не ok';
        }

         if (!User::checkPassword($password)) {
            $errors[] = 'password ne ok';
        }

         $userId = User::checkUserData($email,$password);

         if ($userId == false){
             $errors[] = 'Неправильные данные для входа на сайт';
         } else {
```

```
                User::auth($userId);

                header("Location: /cabinet/");
            }

        }

        require_once ROOT.'/views/user/login.php';
        return true;
    }

    public function actionLogout(){

        unset($_SESSION['user']);
        header("Location: /");
    }
}
```

Файл / controllers/AdminCategoryController.php :

```
<?php

class AdminCategoryController extends AdminBase
{
    public function actionIndex()
    {
        // Проверка доступа
        self::checkAdmin();

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

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

    public function actionCreate()
    {
        self::checkAdmin();


        if (isset($_POST['submit'])) {

            $name = $_POST['name'];
            $sortOrder = $_POST['sort_order'];
            $status = $_POST['status'];

            $errors = false;

            if (!isset($name) || empty($name)) {
                $errors[] = 'Заполните поля';
            }


            if ($errors == false) {
                Category::createCategory($name, $sortOrder, $status);

                header("Location: /admin/category");
            }
        }

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

```
    }

    public function actionUpdate($id)
    {
        self::checkAdmin();

        $category = Category::getCategoryById($id);

        if (isset($_POST['submit'])) {
            $name = $_POST['name'];
            $sortOrder = $_POST['sort_order'];
            $status = $_POST['status'];

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

            header("Location: /admin/category");
        }

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

    public function actionDelete($id)
    {
        // Проверка доступа
        self::checkAdmin();

        // Обработка формы
        if (isset($_POST['submit'])) {
            Category::deleteCategoryById($id);

            header("Location: /admin/category");
        }

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

}
```

Файл / controllers/AdminController.php :

```
<?php

class AdminController extends AdminBase
{
    public static function actionIndex(){
        self::checkAdmin();
        require_once ROOT.'/views/admin/index.php';
        return true;
    }

}
```

Файл / controllers/AdminOrderController.php :

```
<?php

class AdminOrderController extends AdminBase
{

    public function actionIndex()
    {
        self::checkAdmin();
```

```
        $ordersList = Order::getOrdersList();

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

    public function actionUpdate($id)
    {
        self::checkAdmin();

        $order = Order::getOrderById($id);

        if (isset($_POST['submit'])) {
            $userName = $_POST['userName'];
            $userPhone = $_POST['userPhone'];
            $userComment = $_POST['userComment'];
            $date = $_POST['date'];
            $status = $_POST['status'];

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

            header("Location: /admin/order/view/$id");
        }

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

    public function actionView($id)
    {
        self::checkAdmin();

        $order = Order::getOrderById($id);

        $productsQuantity = json_decode($order['products'], true);
        $productsIds = array_keys($productsQuantity);
        $products = Product::getProductByIds($productsIds);

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

    public function actionDelete($id)
    {
        self::checkAdmin();

        if (isset($_POST['submit'])) {
            Order::deleteOrderById($id);

            header("Location: /admin/order");
        }

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

}
```

Файл / controllers/AdminProductController.php :

```
<?php

class AdminProductController extends AdminBase{
```

```
    public function actionIndex(){
        self::checkAdmin();
        $productsList = Product::getProductsList();

        require_once (ROOT.'/views/admin_product/view.php');
        return true;

    }
       public function actionCreate()
    {
        self::checkAdmin();

        $categoryList = Category::getCategoriesListAdmin();

        if (isset($_POST['submit'])) {
            $options['name'] = $_POST['name'];
            $options['code'] = $_POST['code'];
            $options['price'] = $_POST['price'];
            $options['category_id'] = $_POST['category_id'];
            $options['brand'] = $_POST['brand'];
            $options['availability'] = $_POST['availability'];
            $options['description'] = $_POST['description'];
            $options['is_new'] = $_POST['is_new'];
            $options['is_recommended'] = $_POST['is_recommended'];
            $options['status'] = $_POST['status'];

            $errors = false;

               if (!isset($options['name']) || empty($options['name'])) {
                $errors[] = 'Заполните поля';
            }

            if ($errors == false) {
                $id = Product::createProduct($options);

                if ($id) {

                    if (is_uploaded_file($_FILES["image"]["tmp_name"])) {
                                        move_uploaded_file($_FILES["image"]["tmp_name"],
$_SERVER['DOCUMENT_ROOT'] . "/upload/images/products/{$id}.jpg");
                    }
                };

                header("Location: /admin/product");
            }
        }

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

    public function actionUpdate($id)
    {
        self::checkAdmin();


        $categoryList = Category::getCategoriesListAdmin();

        $product = Product::getProductById($id);

        if (isset($_POST['submit'])) {
```

```
            $options['name'] = $_POST['name'];
            $options['code'] = $_POST['code'];
            $options['price'] = $_POST['price'];
            $options['category_id'] = $_POST['category_id'];
            $options['brand'] = $_POST['brand'];
            $options['availability'] = $_POST['availability'];
            $options['description'] = $_POST['description'];
            $options['is_new'] = $_POST['is_new'];
            $options['is_recommended'] = $_POST['is_recommended'];
            $options['status'] = $_POST['status'];

            if (Product::updateProductById($id, $options)) {

                if (is_uploaded_file($_FILES["image"]["tmp_name"])) {

                    move_uploaded_file($_FILES["image"]["tmp_name"], $_SERVER['DOCUMENT_ROOT'] .
"/upload/images/products/{$id}.jpg");
                }
            }


            header("Location: /admin/product");
        }

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


    }

    public static function actionDelete($id){
        self::checkAdmin();

        if (isset($_POST['submit'])) {
            Product::deleteProductById($id);

            header("Location: /admin/product");
        }

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

Файл /models/Cart.php :

```
<?php

class Cart {

    public static function addProduct($id){
        $id = intval($id);

        $productsInCart = array();

        if(isset($_SESSION['products'])){
            $productsInCart = $_SESSION['products'];

        }

        if(array_key_exists($id, $productsInCart)){
            $productsInCart[$id] ++;
        } else {
```

```
            $productsInCart[$id] = 1;
        }
        $_SESSION['products'] = $productsInCart;


        return true;
    }

    public static function downperCart($id){
        $id = intval($id);


        if(isset($_SESSION['products']["{$id}"])&&($_SESSION['products']["{$id}"]!=0)){
            $_SESSION['products']["{$id}"] --;
        }
        return true;
    }
    public static function upperCart($id){
        $id = intval($id);


        if(isset($_SESSION['products']["{$id}"])){
            $_SESSION['products']["{$id}"]++;

        }

        return true;
    }

     public static function countItems(){
         if(isset($_SESSION['products'])){
             $count = 0;
             foreach ($_SESSION['products'] as $id => $quantity) {
                 $count = $count + $quantity;
             }
             return $count;
         } else {
             return 0;
         }
     }

     public static function getProducts(){
         if (isset($_SESSION['products'])) {
             return $_SESSION['products'];
         }
         return false;
     }

     public static function getTotalPrice($products){
         $productsInCart = self::getProducts();

         $total = 0;
         if ($productsInCart) {
             foreach ($products as $item){
                 $total += $item['price'] * $productsInCart[$item['id']];
             }
         }
         return $total;
     }

       public static function deleteProduct($id){
        $id = intval($id);
```

```
        if(isset($_SESSION['products'])){
            unset( $_SESSION['products'][$id]);
        }

        return true;
    }
}
```

Файл /models/ Category.php :

```
<?php
class Category {

    public static function getCategoriesList(){

        $db = Db::getConnection();

        $categoryList = array();

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

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

    public static function getCategoriesListAdmin(){

        $db = Db::getConnection();

        $categoryList = array();

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

        $i=0;
        while ($row = $resalt->fetch()) {
            $categoryList[$i]['id'] = $row['id'];
            $categoryList[$i]['name'] = $row['name'];
            $categoryList[$i]['sort_order'] = $row['sort_order'];
            $categoryList[$i]['status'] = $row['status'];
            $i++;
        }
        return $categoryList ;
    }

     public static function deleteCategoryById($id)
    {
        $db = Db::getConnection();

        $sql = 'DELETE FROM category WHERE id = :id';

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

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

```
    $db = Db::getConnection();

        $sql = "UPDATE category
        SET
            name = :name,
            sort_order = :sort_order,
            status = :status
        WHERE id = :id";

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

public static function getCategoryById($id)
{
    // Соединение с БД
    $db = Db::getConnection();

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

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

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

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

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

public static function getStatusText($status)
{
    switch ($status) {
        case '1':
            return 'Отображается';
            break;
        case '0':
            return 'Скрыта';
            break;
    }
}

public static function createCategory($name, $sortOrder, $status)
{
    // Соединение с БД
    $db = Db::getConnection();

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

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

```
        $result->bindParam(':sort_order', $sortOrder, PDO::PARAM_INT);
        $result->bindParam(':status', $status, PDO::PARAM_INT);
        return $result->execute();
    }

}
```

Файл /models/ Order.php :

```
<?php

class Order {
    public static function save($user_name, $user_phone, $user_comment, $user_id, $products){

        $products = json_encode($products);

        $db = Db::getConnection();

        $sql = 'INSERT INTO product_order (user_name, user_phone, user_comment, user_id, products) '.
               ' VALUES (:user_name, :user_phone, :user_comment, :user_id, :products)';

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

        return $result->execute();

    }

     public static function getOrdersList()
    {
        // Соединение с БД
        $db = Db::getConnection();

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

    public static function getStatusText($status)
    {
        switch ($status) {
            case '1':
                return 'Новый заказ';
                break;
            case '2':
                return 'В обработке';
                break;
            case '3':
```

```
            return 'Доставляется';
            break;
        case '4':
            return 'Закрыт';
            break;
    }
}

public static function getOrderById($id)
{
    // Соединение с БД
    $db = Db::getConnection();

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

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

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

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

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

public static function deleteOrderById($id)
{
    // Соединение с БД
    $db = Db::getConnection();

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

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

public static function updateOrderById($id, $userName, $userPhone, $userComment, $date, $status)
{
    // Соединение с БД
    $db = Db::getConnection();

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

    $result = $db->prepare($sql);
    $result->bindParam(':id', $id, PDO::PARAM_INT);
    $result->bindParam(':user_name', $userName, PDO::PARAM_STR);
    $result->bindParam(':user_phone', $userPhone, PDO::PARAM_STR);
    $result->bindParam(':user_comment', $userComment, PDO::PARAM_STR);
```

```
        $result->bindParam(':date', $date, PDO::PARAM_STR);
        $result->bindParam(':status', $status, PDO::PARAM_INT);
        return $result->execute();
    }

    public static function getOrderByUserId($user_id)
    {
        // Соединение с БД
        $db = Db::getConnection();

        // Текст запроса к БД
       $result = $db->query('SELECT id, user_id,user_name, user_phone, date, status FROM product_order
WHERE user_id = '.$user_id.' ORDER BY id DESC');
        $ordersList = array();
        $i = 0;
        while ($row = $result->fetch()) {
            $ordersList[$i]['id'] = $row['id'];
            $ordersList[$i]['user_id'] = $row['user_id'];
            $ordersList[$i]['user_name'] = $row['user_name'];
            $ordersList[$i]['user_phone'] = $row['user_phone'];
            $ordersList[$i]['date'] = $row['date'];
            $ordersList[$i]['status'] = $row['status'];
            $i++;
        }
        return $ordersList;

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

Файл /models/ Product.php :

```
<?php

class Product {
    const SHOW_BY_DEFAULT = 12;

    public static function getLatestProducts($count = self::SHOW_BY_DEFAULT){
        $count = intval($count);

        $db = Db::getConnection();

        $productList = array();

        $resalt  = $db->query('SELECT id,name,price,is_new FROM product'.
              ' WHERe status = "1" ORDER BY id DESC LIMIT ' . $count);

        $i=0;
        while ($row = $resalt->fetch()) {
            $productList[$i]['id'] = $row['id'];
            $productList[$i]['name'] = $row['name'];
            $productList[$i]['price'] = $row['price'];
            $productList[$i]['is_new'] = $row['is_new'];
            $i++;
        }
        return $productList ;

    }

    public static function getProductsListByCategory($categoryId = false, $page = 1){
        if($categoryId) {
            $categoryId = intval($categoryId);
            $offset = ($page-1) * self::SHOW_BY_DEFAULT;
```

```
            $db = Db::getConnection();
            $product = array();
            $resalt  = $db->query('SELECT id,name,price,is_new FROM product'.
                ' WHERe status = "1" AND category_id = '.$categoryId
                    .' ORDER BY id ASC LIMIT ' . self::SHOW_BY_DEFAULT.
                    ' OFFSET '.$offset);
        $i=0;
        while ($row = $resalt->fetch()) {
            $product[$i]['id'] = $row['id'];
            $product[$i]['name'] = $row['name'];
            $product[$i]['price'] = $row['price'];
            $product[$i]['is_new'] = $row['is_new'];
            $i++;
        }
        return $product ;
        }
    }

    public static function getProductById($id){

        $id = intval($id);
        if($id){
        $db = Db::getConnection();

        $resalt  = $db->query('SELECT * FROM product WHERE id = '.$id);
        $resalt->setFetchMode(PDO::FETCH_ASSOC);

        return $resalt->fetch();
        }
    }
    public static function getProductByIds($idsArray){

        $products = array();

        $db = Db::getConnection();

        $idsArray = implode(',',$idsArray);

        $sql = "SELECT * FROM product WHERE status = '1' AND id IN ({$idsArray})";
        $result  = $db->query($sql);
        $result->setFetchMode(PDO::FETCH_ASSOC);

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

        return $products;
        }


    public static function getTotalProductsInCategory($categoryId){
         $db = Db::getConnection();
        $resalt  = $db->query('SELECT count(id) AS count FROM product WHERE category_id =
'.$categoryId);
        $resalt->setFetchMode(PDO::FETCH_ASSOC);
        $row = $resalt->fetch();

        return $row['count'];
```

```
 }

public static function getRecomendedProducts(){
      $db = Db::getConnection();

   $recomendedList = array();

   $resalt  = $db->query('SELECT id,name,price,is_new FROM product'.
         ' WHERe status = "1" AND is_recommended = "1" ORDER BY id DESC LIMIT 20');

   $i=0;
   while ($row = $resalt->fetch()) {
      $recomendedList[$i]['id'] = $row['id'];
      $recomendedList[$i]['name'] = $row['name'];
      $recomendedList[$i]['price'] = $row['price'];
      $recomendedList[$i]['is_new'] = $row['is_new'];
      $i++;
   }
   return $recomendedList ;
 }

public static function getProductsList(){

   $db = Db::getConnection();

   $productList = array();

   $resalt  = $db->query('SELECT id,name,price,code FROM product'.
         ' WHERe status = "1" ORDER BY id ASC');

   $i=0;
   while ($row = $resalt->fetch()) {
      $productList[$i]['id'] = $row['id'];
      $productList[$i]['name'] = $row['name'];
      $productList[$i]['price'] = $row['price'];
      $productList[$i]['code'] = $row['code'];
      $i++;
   }
   return $productList ;

}

public static function deleteProductById($id){

   $db = Db::getConnection();

   $sql = 'DELETE FROM product WHERE id = :id';

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

   return $result->execute() ;

}

public static function updateProductById($id, $options)
{
   // Соединение с БД
   $db = Db::getConnection();

   // Текст запроса к БД
   $sql = "UPDATE product
      SET
```

```
            name = :name,
            code = :code,
            price = :price,
            category_id = :category_id,
            brand = :brand,
            availability = :availability,
            description = :description,
            is_new = :is_new,
            is_recommended = :is_recommended,
            status = :status
        WHERE id = :id";

    $result = $db->prepare($sql);
    $result->bindParam(':id', $id, PDO::PARAM_INT);
    $result->bindParam(':name', $options['name'], PDO::PARAM_STR);
    $result->bindParam(':code', $options['code'], PDO::PARAM_STR);
    $result->bindParam(':price', $options['price'], PDO::PARAM_STR);
    $result->bindParam(':category_id', $options['category_id'], PDO::PARAM_INT);
    $result->bindParam(':brand', $options['brand'], PDO::PARAM_STR);
    $result->bindParam(':availability', $options['availability'], PDO::PARAM_INT);
    $result->bindParam(':description', $options['description'], PDO::PARAM_STR);
    $result->bindParam(':is_new', $options['is_new'], PDO::PARAM_INT);
    $result->bindParam(':is_recommended', $options['is_recommended'], PDO::PARAM_INT);
    $result->bindParam(':status', $options['status'], PDO::PARAM_INT);
    return $result->execute();
}

public static function createProduct($options)
{
    // Соединение с БД
    $db = Db::getConnection();

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

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

 public static function getAvailabilityText($availability)
{
    switch ($availability) {
        case '1':
```

```
                return 'В наличии';
                break;
            case '0':
                return 'Под заказ';
                break;
        }
    }

    public static function getImage($id)
    {
        // Название изображения-пустышки
        $noImage = 'no_image.png';

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

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

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

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

}
```

Файл /models/ Search.php :

```
<?php

class Search {
    public static function getProductByName($search){
         $db = Db::getConnection();
        $products = array();
        $result  = $db->query('SELECT * FROM product WHERE name LIKE "%'.$search.'%"');
        $result->setFetchMode(PDO::FETCH_ASSOC);
        $i=0;
        while ($row = $result->fetch()){
          $products[$i]['id']=$row['id'];
          $products[$i]['code']=$row['code'];
          $products[$i]['name']=$row['name'];
          $products[$i]['price']=$row['price'];
          $i++;
        }

        return $products;
    }
}
```

Файл /models/ User.php :

```
<?php

class User
{

    public static function register($name,$last_name, $email,$sity, $password)
    {
        $db = Db::getConnection();
```

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

    // Получение и возврат результатов. Используется подготовленный запрос
    $result = $db->prepare($sql);
    $result->bindParam(':name', $name, PDO::PARAM_STR);
    $result->bindParam(':last_name', $last_name, PDO::PARAM_STR);
    $result->bindParam(':email', $email, PDO::PARAM_STR);
    $result->bindParam(':sity', $sity, PDO::PARAM_STR);
    $result->bindParam(':password', $password, PDO::PARAM_STR);
    if ($result->execute()) {
        // Если запрос выполенен успешно, возвращаем id добавленной записи
        return $db->lastInsertId();
    }
    return 0;
}

public static function edit($id, $name,$last_name,$sity, $password)
{
    // Соединение с БД
    $db = Db::getConnection();

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

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

public static function checkUserData($email, $password)
{
    // Соединение с БД
    $db = Db::getConnection();

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

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

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

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

public static function auth($userId)
```

```
{
    // Записываем идентификатор пользователя в сессию
    $_SESSION['user'] = $userId;
}

public static function checkLogged()
{
    // Если сессия есть, вернем идентификатор пользователя
    if (isset($_SESSION['user'])) {
        return $_SESSION['user'];
    }

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

public static function isGuest()
{
    if (isset($_SESSION['user'])) {
        return false;
    }
    return true;
}

public static function checkName($name)
{
    if (strlen($name) >= 2) {
        return true;
    }
    return false;
}

public static function checkPhone($phone)
{
    if (strlen($phone) >= 10) {
        return true;
    }
    return false;
}


public static function checkPassword($password)
{
    if (strlen($password) >= 6) {
        return true;
    }
    return false;
}

public static function checkEmail($email)
{
    if (filter_var($email, FILTER_VALIDATE_EMAIL)) {
        return true;
    }
    return false;
}

public static function checkEmailExists($email)
{
    // Соединение с БД
    $db = Db::getConnection();

    // Текст запроса к БД
    $sql = 'SELECT COUNT(*) FROM users WHERE email = :email';
```

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

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

    public static function getUserById($id)
    {
        // Соединение с БД
        $db = Db::getConnection();

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

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

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

        return $result->fetch();
    }

}
```

ДОДАТОК Б

Відгук керівника економічного розділу

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

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