МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ВИПУСКНА РОБОТА

на тему:

«Інформаційна система моніторингу обсягів купівлі продажу для мережі магазинів»

Завідувач
випускаючої кафедри **Довбиш А. С.**

Керівник роботи **Ободяк В. К.**

Студентка групи ІН-64-8 **Шовкопляс Н. Р.**

СУМИ 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютерних наук

Затверджую_____

Зав. кафедри Довбиш А. С.

«_____» _____20___ р.

ЗАВДАННЯ

до випускної роботи

Студентки четвертого курсу, групи ІН-64-8 спеціальності «Інформатика» денної форми навчання Шовкопляс Надії Ростиславівни.

Тема: «Інформаційна система моніторингу обсягів купівлі продажу для мережі магазинів»

Затверджена наказом по СумДУ

№_____ від _____20___ р.

Зміст пояснювальної записки: 1) інформаційний огляд; 2) постановка задачі; 3) методи розроблення веб-додатка; 4) практична реалізація.

Дата видачі завдання «_____»_____ 20___ р.

Керівник випускної роботи _____ Ободяк В. К.

Завдання приняла до виконання _____ Шовкопляс Н. Р.

РЕФЕРАТ

Записка: 61 стор., 37 рис., 1 додаток, 13 джерел.

Об’єкт дослідження – процес моніторингу обсягів купівлі-продажу для мережі магазинів.

Мета роботи – розроблення веб-додатка моніторингу обсягів купівлі-продажу для мережі магазинів.

Методи дослідження – застосування мови розмітки HTML, мови PHP, JavaScript, каскадних таблиць стилів CSS, фреймворку Bootstrap, БД MySQL.

Результати – розроблений веб-додаток моніторингу обсягів купівлі-продажу для мережі магазинів, головним призначенням якого є аналітичний аналіз різних категорій товарів в інших магазинах і їх порівняння з продукцією власної мережі магазинів.

МОНІТОРИНГ, ВЕБ-ДОДАТОК, MYSQL, HTML,
CSS, PHP, JAVASCRIPT, BOOTSTRAP

Зміст

ВСТУП

Internet надав небувалий раніше доступ до колосальної кількості інформаційних ресурсів. Кожен день в Internet створюється величезна кількість даних. Люди, підприємства, пристрої перетворюються на фабрики даних, які щоденно завантажують неймовірну кількість інформації в Internet. На сьогоднішній день зареєстровано сотні мільйонів доменних імен у світі, а окремих веб-сторінок нараховуються мільярди і кожен день з'являються нові. Особливо привабливими можливості Internet виявилися для комерційних організацій. Для багатьох присутність в Internet – питання престижу, але для ряду компаній це тільки перший крок.

На сучасному етапі розвитку людської діяльності важливе місце займає її всебічний моніторинг. Суть його полягає в зборі необхідної інформації і ретельному аналізі. Вручну одній людині відслідковувати її не під силу, а наймати для цих цілей штат з декількох десятків співробітників досить не вигідно.

Регулярне проведення моніторингу забезпечує своєчасне виявлення помилок і, відповідно, їх виправлення в найкоротші терміни.

Основний метод, який використовується в моніторингу – постійне спостереження. Він дозволяє швидко отримувати інформацію про цікаві параметри і в разі чого бути готовим до будь-яких змін.

1 ІНФОРМАЦІЙНИЙ ОГЛЯД

1.1 Аналіз предметної області застосування

Створення сайта є важливим завданням підприємницької діяльності, в тому числі, і мережі Internet.

Веб-сайт або просто сайт (від англ. website – місце в Internet) – комплекс веб-сторінок, що доступні у мережі Internet, об’єднані за змістом, навігацією під єдиним доменним ім’ям. Фізично сайт може бути розміщений на одному або кількох серверах.

Залежно від призначення веб-сторінки класифікують так (рис. 1.1):

– головна сторінка – це домашня сторінка, з якої користувач розпочинає перегляд веб-сайта при переході за URL-адресою. Зазвичай на цій сторінці розкриваються призначення та тематика сайта, можна знайти інформацію про розробників та йде ознайомлення, які матеріали містяться на інших сторінках ресурсу;

– інформаційні сторінки – це веб-сторінки з тематичними розділами, на яких розміщується контент для розуміння теми сайта та його складових: тексти, зображення тощо. Окремим типом інформаційної сторінки є головна сторінка веб-ресурсу;

– сторінки-контейнери – це веб-сторінки зі списками посилань на різні ресурси сайтів. Виділяють веб-каталоги, які посилаються на веб-ресурси, та каталоги файлів, які є посиланнями на файли для завантаження користувачами;

– комунікативні сторінки – це інтерактивні веб-сторінки, призначенням яких є забезпечення зворотного зв’язку користувачів із розробниками сервісу. За формою взаємодії сторінки мають своїм призначенням:

1) організація спілкування засобами форуму;
2) організація спілкування через чати;

3) забезпечення зворотного зв’язку з користувачами за допомогою коментарів та відгуків, які вони залишають для розробників сайта на сторінках гостьової книги;

4) проведення анкетування, вибір послуг або товарів, збір інформації про думку користувачів та інше на сторінках форми [1].

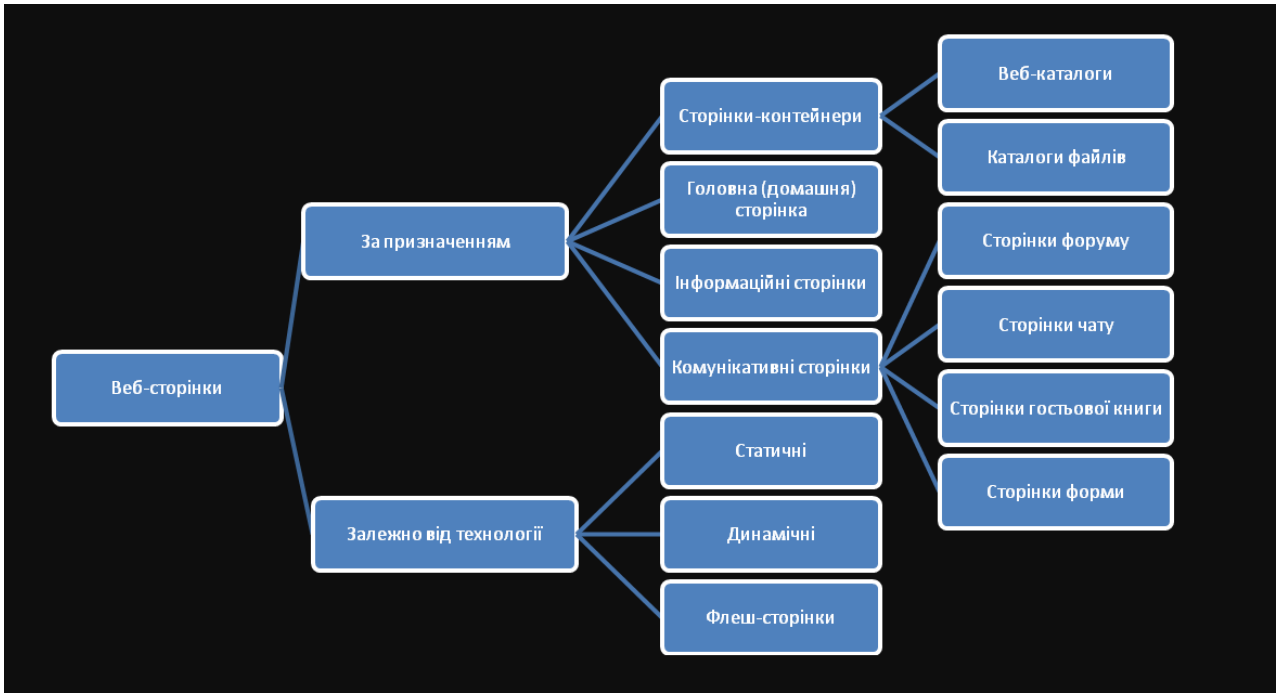

Рисунок 1.1 – Схема класифікації веб-сторінок

Відштовхуючись від використаних технологій при розробленні веб-сторінок їх можна поділити на такі типи:

– статичні сторінки, як правило, створені із застосуванням мови розмітки HTML (англ. Hyper Text Markup Language). Для всіх відвідувачів сайта вміст сторінок залишається незмінним. Для таких сторінок не потрібне постійне оновлення та спеціальні програми, щоб зберігати дані. Прикладом таких сторінок можуть бути сторінки з історичними даними, документацією, сайт-візитка, персональна сторінка, блог тощо;

– динамічні веб-сторінки написані такими мовами, як PHP (англ. Personal Home Page Tools), ASP (англ. Active Server Pages), PERL (англ. Practical Extraction and Report Language) тощо. На динамічних веб-сторінках

вміст сторінок відрізняється для різних відвідувачів. Завантаження займає більше часу, ніж статична веб-сторінка. Динамічні веб-сторінки використовуються там, де інформація часто змінюється, наприклад, ціни акцій, інформація про погоду, новини тощо.

− флеш-сторінки (англ. flash – спалах) це такі веб-сторінки, фундаментальною функцією яких є анімація, необхідна для розширення взаємодії з сайтом. Створюються такі веб-сторінки за допомогою динамічної анімації, руху та відео засобами програмної системи для розробки векторної анімації Adobe Flash. До беззаперечних переваг флеш-сторінок відносяться інтерактивність та кросбраузерна сумісність. Але є причини, щоб узяти під сумнів доцільність використання цього інструмента. Флеш-сайти потребують встановлення плагіну Flash player, пошукові системи не індексують зображення. Обсяги файлів, де зберігаються флеш-сторінки, дуже великі, тому швидкість завантаження таких сторінок досить низька. Крім того, flash-додатки пред’являють високі вимоги до ресурсів процесора. Технологія Adobe Flash допомогла відкрити абсолютно нову сферу для Internet і додатків, і протягом десятиліттями була надійною системою розробки додатків, дозволила отримати ефектну графіку. Однак, як трапляється і з іншими технологіями, настав час для впровадження нової ери створення цифрового контенту [2].

У браузері працює така комп’ютерна програма як веб-додаток.

Веб-додаток – будь-яка комп'ютерна програма, що виконує певну функцію, використовуючи веб-браузер у якості свого клієнта.

Середовище клієнт-сервер - це система, в якій кілька комп’ютерів обмінюються інформацією, наприклад, введенням інформації в базу даних.

"Клієнт" використовується для введення інформації, а "сервер" – програма, що використовується для зберігання інформації.

Веб-додатки можуть забезпечити однакову функціональність та отримати перевагу роботи на кількох платформах. Наприклад, веб-додаток

може діяти як текстовий процесор, зберігаючи інформацію в хмарі і дозволяючи завантажувати документ на свій особистий жорсткий диск.

Швидкість веб-сайта є головним критерієм для користувача. Одна з найбільш неприємних речей – це низька швидкість веб-сайта, яка негативно позначиться на використанні ресурсу.

Високопродуктивні веб-сайти сприяють високому рівню відвідувань, низьких показників відмов та вищої кількості залученості. Повільні веб-сайти будуть коштувати розробникам пошкодженої репутації. Наприклад, скорочення часу завантаження сторінки позитивно вплине на процеси маркетингу та продажу, а результатом буде більший трафік та залучення більш кваліфікованих клієнтів.

Не менш важливим на сьогодні стає відстеження поведінки користувачів, яка описує взаємодію людей із веб-сайтом під час відвідування. Можна побачити, скільки часу вони проводять на сторінці, скільки і яких сторінок вони відвідують, а також кількість дій, які здійснюються, наприклад, при натисканні на посилання або відтворенні відео.

Моніторинг має вирішальне значення для діяльності і розвитку IT-систем. Компанії постійно проводять різні заходи, наприклад, моніторинг мережі, перевірку та моніторинг серверів, а також комп'ютерної техніки. Після проведення всіх необхідних робіт можна зібрати потрібні дані про стан кожного компонента IT-інфраструктури з метою подальшого аналізу, поліпшення і вирішення можливих бізнес-процесів.

Переваги своєчасного моніторингу IT-інфраструктури:

− правильний підхід до системи моніторингу, який можуть забезпечити грамотні фахівці, дозволяє своєчасно виявити можливі поточні проблеми в системах, а також проводити ведення статистики їх стану;

− завдяки моніторингу IT-інфраструктури, у разі виникнення непередбаченої ситуації можна швидко виявити проблему, усунути її в короткі терміни і більше не допустити.

Існуючі види індикаторів моніторингу:

− моніторинг ресурсів – вимір кількості споживаних ресурсів;

− моніторинг продуктів – вимір кількості наданих послуг;

− моніторинг результатів – вимір результатів від реалізації програми для надання послуг;

− моніторинг ефективності – зіставлення витрачених ресурсів і результатів.

На сьогоднішній день систем моніторингу Web-сайтів дуже багато. Більшість призначені для вирішення таких завдань як пошук вірусів на сайтах, відслідковування часу відгуку, моніторинг доступності, також дуже поширені системи моніторингу пошукової видачі сайтів, визначення позиції сайту по певних пошукових запитах. Але для вирішення таких завдань майже жодна з цих систем моніторингу не підходить. Покажемо це, навівши класифікацію систем моніторингу веб-сайтів по об’єкту спостереження і цілі [3].

Як видно з рисунка 1.2, система займає окрему нішу у множині застосувань, призначених для моніторингу сайтів. На сьогодні аналогічні завдання можна вирішити використовуючи опитування пошукових систем, але такий варіант можливий тільки за допомогою автоматизованого виконання запитів до пошукової системи для кожного сайта, проте такий шлях більшістю з подібних систем не приймається.

Також такий метод має певні недоліки:

− неспромога обробки ресурсів, доступ до яких неможливий через Internet;

− від пошукової системи залежить швидкість оновлення даних;

− відсутність повного контролю над процесом формування даних пошуковою системою.

Види веб-сайтів за технологією відображення:

− статичні сайти – зазвичай мають фіксовану кількість сторінок, які мають певний макет. Вміст сторінки є буквально статичним і не змінюється у

відповідь на дії користувача. Статичний веб-сайт, як правило, створюється з допомогою HTML і каскадних таблиць стилів CSS (англ. Cascading Style Sheets) у простих текстових редакторах.

− динамічні сайти – користувач має можливість взаємодіяти з інформацією на сторінці. При створені використовуються база даних, різні скрипти, вміст генерується.

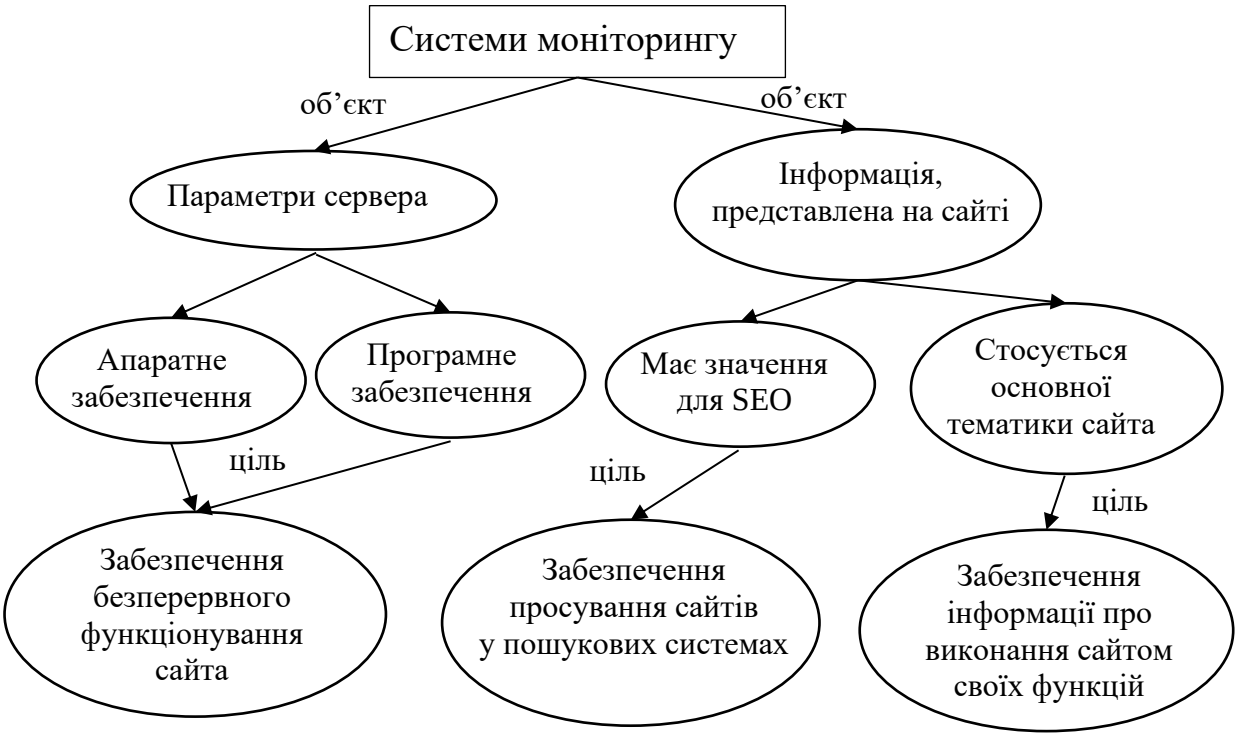

Рисунок 1.2 – Система класифікації систем моніторингу веб-сайтів

Створений веб-сайт має бути корисним, привабливим і максимально зручним для користувача. Існують такі етапи створення веб-орієнтованого інтерфейсу.

Розробка дизайну. Дизайн користувацького інтерфейсу будь-якого програмного продукту має вирішальне значення для його успіху. На даному кроці починається розробка концепції надання контенту користувачу. Важливим є чітке розуміння основних майбутніх цілей веб-сайта та цільової аудиторії, яка потім буде залучатися на сайт. Веб-дизайнер ретельно обмірковує кожну деталь макета, розробляє стратегію. Після чого структура

обговорюється співробітниками студії (SEO-спеціалісти, верстальники, програмісти). Після затвердження структури вона погоджується із замовником і починається створення макетів, які представляють собою кілька версій дизайну. Усі деталі повинні бути опрацьовані: форми, заголовки, списки, кнопки і т.ін. Замовник отримує кінцевий результат. У разі затвердження макети передаються до веб-студії.

Верстка. Верстальник отримує макети, що складаються з шарів, розроблених в графічному редакторі (Ai, Psd). Завданням верстальника є перетворити графічні макети у веб-сторінки. На даному етапі розробники вирішують основне питання щодо кросбраузерної сумісності. Дуже часто веб-сторінка неоднаково відображається у різних браузерах, що є недопустимим. У випадках, коли елементи сторінки інтерпретуються некоректно в різних браузерах, розробники вигадують різні скрипти, які потім впроваджують в HTML чи в CSS. Після перевірки ідентичності роботи веб-сторінки у популярних браузерах, починається етап програмування.

Програмування. На даному етапі створюється програмна частина ресурсу, програмісти поєднують результати робіт дизайнерів і верстальників. Оскільки системи управління сайтом (CMS) щодалі набувають усе більшої популярності, тому зазвичай усі сучасні сайти використовують CMS, аби не розробляти основу сайта з нуля.

Після аналізу технічного завдання, веб-студія визначає тип веб-сайта, обмірковує обсяг роботи і визначається з ціною створення сайта.

Для WEB-розробок традиційно використовують такі мови програмування як Perl, PHP, ASP і інші, вони дозволяють реалізовувати практично будь-які завдання. Але обробляти за їх допомогою великі обсяги даних, що мають до того ж складну структуру, досить важко. Розроблення подібних програм вимагає зростаючих витрат праці програмістів, у геометричній прогресії зростає обсяг програмного коду і кількість помилок, знижується надійність програмного забезпечення. У такій ситуації на

допомогу програмісту приходять бази даних. Згідно із класичним визначенням, база даних – це впорядкована сукупність інформації, що зберігається у вигляді множин, кожне з яких містить записи уніфікованого виду. Системи управління базами даних (СУБД) надають програмісту потужний інструментарій для створення, оновлення та обробки великих обсягів інформації, що має складну структуру.

У класичній теорії виділяють три типи баз даних: ієрархічні, мережні та реляційні БД [4, 5].

− У ієрархічних базах даних дані упорядковують у «деревоподібну» структуру. Це схоже на папки та файли на комп’ютері. Так само, як файл на комп’ютері знаходиться в одній папці, так і кожен запис у базі даних має один "батьківський" запис, що забезпечує цілісність посилань. Ієрархічно впорядковані дані зазвичай описуються у вигляді стосунків предок/нащадок. Якщо видалити запис з дерева, то всі його нащадки теж будуть знищені.

− Мережні бази даних є розширенням ієрархічної моделі – записи можуть мати кількох власників, і тим самим забезпечувати безліч шляхів доступу до даних.

− На відміну від мережної та ієрархічної моделей, реляційна модель має більш високий рівень складності. Реляційна модель не має проблем із відносинами «багато до одного» або «багато до багатьох». Їх записи побудовані у вигляді декількох «таблиць», а не деревних структур, і кожен запис на таблиці має унікальний ідентифікатор. Кожен запис на батьківській таблиці може бути «батьком» для одного (або більше, або жодного) унікального запису «дочірньої таблиці». Запис таких зв’язків – це те, що дає реляційним базам даних їх ім’я.

1.2 Актуальність виконання проєкту

У наш час величезна кількість фірм використовує персональні комп’ютери для збереження і обробки будь-якого виду інформації. Ця інформація міститься в базах даних. Бази даних відіграють важливу роль у

світі розвитку технологій. Робота з базами даних є найважливішим навиком по роботі з комп'ютером, а фахівці даної області стають все більш затребуваними.

База даних – це організована структура, яка призначена для зберігання інформації. У той час, коли відбувався розвиток баз даних, в них зберігалася виключно інформація, проте вже в наші дні багато систем управління базами даних дозволяє розміщувати в своїх структурах і дані, і програмний код, за допомогою якого здійснюється зв'язок із користувачами або з іншими програмно-апаратними комплексами. База даних створюється для збереження і безпосереднього доступу до інформації, що містить відомості про шукану предметну область. Ступінь конкретизації даних обумовлюється групою факторів [6, 7].

Будь-яка сучасна компанія не може обійтися без бази даних. Це навчальні заклади, банки, магазини, заводи, будь-які підприємства і державні установи. Вони використовують їх для перекладу даних в електронний вигляд і об'єднання даних, а також оперативного доступу до них. Це дозволяє економити час і кошти на витрати.

Наприклад, розглянемо задачу формування університетського журналу успішності студентів і роботи з ним. Потрібно упорядкувати великим об'єм однотипних даних про студентів (персональні дані, адреса, телефон, контактні номери телефонів батьків тощо) і про процес навчання (загальна успішність, результати атестацій та підсумкового контролю знань, заняття, додаткові заходи і т.ін.). Використати алгоритмічні мови для автоматизації цього завдання не вийде. Для подібних задач і призначені системи управління базами даних, які дозволяють ефективно знаходити, вставляти, змінювати та видаляти дані. Ці системи застосовуються не тільки для вирішення завдань одного типу.

Потрібно підкреслити, що база даних є головною і найскладнішою частиною інформаційних систем (ІС) – інтегрованих комплексів для збору, зберігання і обробки даних, а також для надання інформації. Спочатку ці

системи були тільки в паперовому вигляді. Для їх зберігання використовували різні папки, приміщення та архіви. Стрімкий розвиток інформаційно-комунікаційних технологій забезпечив широке використання автоматизованих інформаційних систем. Сьогодні ІС обслуговують різні сфери діяльності, системи управління господарських і технічних об’єктів, модельні комплекси для наукових досліджень, системи автоматизації моделювання та виробництва; створюються різні навчальні системи. Сучасні ІС інтегрують дані великих об’ємів зі складною організацією, вони мають забезпечити різноманітні запити різних категорій користувачів. Отже, системою управління базами даних називають програмне середовище, яке призначене для створення на персональному комп’ютері загальної бази даних для безлічі додатків, підтримки її в діючому стані і поліпшення ефективності доступу користувачів до вмісту в ній інформації у рамках наданих їм повноважень [8, 9].

Моніторинг є важливою частиною інформаційної інфраструктури сучасних організацій. Необхідно постійно контролювати правильну роботу інфраструктури, щоб можлива помилка не вплинула на послугу, яка надається користувачам. Для виявлення та запобігання збоїв компанія повинна мати дієвий інструмент моніторингу, який відповідає за управління апаратним забезпеченням, мережею, комунікаціями, операційними системами, програмними додатками тощо, аналізує їх роботу і продуктивність, сповіщає про можливі помилки. Розглянемо електронний магазин як приклад бізнесу, веб-сайт якого став працювати погано або дуже погано. За відсутності системи моніторингу, ймовірно, знадобляться години для з’ясування проблеми, що може призвести до значних збитків. Але за цей час магазин не тільки залишається без продажу, але й зменшує довіру клієнтів, які більше не сприйматимуть магазин як надійний варіант для здійснення покупок в Internet. Більш того, ці користувачі поділяться неприємним досвідом із друзями,

знайомими, або навіть у соціальних мережах, що спричить негативну рекламу, і бізнес зазнає ще більше втрат.

Гарна система моніторингу попередить про виникнення проблеми і дозволить мінімізувати час на її вирішення. Відповідно, обслуговування клієнтів та імідж компанії не постраждають. А кваліфікований персонал приділятиме увагу основним задачам, що також сприятиме підвищенню продуктивності.

Контроль над змінами в інфраструктурі – ще одне важливе завдання. Для цього проводиться інвентаризація програмно-апаратних засобів і забезпечується автоматизована підтримка актуальної інформації про інфраструктуру. Завдяки моніторингу відбувається оперативне виявлення збоїв, а процес внесення змін в інфраструктуру регламентований.

Підвищення ефективності управління мережею магазинів та істотне зниження витрат можливе тільки при організації жорстко централізованої структури – єдиної інформаційної системи. Наприклад, за наявності всього однієї такої системи адміністратору мережі магазинів дуже зручно контролювати прибуток, збитки, продаж, повернення тощо. Кожен магазин має бути авторизованим в цій системі і періодично (кожного дня) вносити потрібні дані. Адміністратор в будь-який момент часу може проконтролювати фінансову ситуацію кожного магазину.

Єдина інформаційна система надає такі переваги:

− управління всіма об’єктами мережі ведеться менеджерами з одного місця, що забезпечує безпосередній їх контакт між собою і з керівництвом. Це дозволяє швидко приймати як тактичні, так і стратегічні рішення. Кожен менеджер в такій схемі може відповідати за свої об’єкти мережі або за свою товарну групу;

− управління запасами проводиться централізовано. Це означає, що один менеджер може управляти декількома об’єктами, маючи оперативний

доступ до інформації по залишках на всіх своїх об’єктах, і може навіть перерозподіляти запаси між магазинами;

− централізоване управління закупками. Інформація про замовлення від магазинів на поповнення запасів відразу ж стає доступною для розподільного центру. Всі замовлення консолідуються і включаються в замовлення постачальникам. Умови поставки для всієї мережі магазинів стають більш вигідними;

− управління ціноутворенням здійснюється з єдиного центру, що дозволяє проводити гнучку цінову політику, встановлюючи різні ціни для магазинів, наприклад залежно від їх територіального розташування;

− реалізація єдиної маркетингової стратегії. Проведення рекламних акцій в рамках мережі. Оперативне отримання об'єктивних результатів маркетингових дій;

− єдність, детальність, оперативність і повнота інформації про всі об’єкти мережі дозволяє проводити актуальні аналітичні оцінки як по кожному об'єкту, так і по всьому підприємству в цілому, що забезпечує найбільш ефективне використання оборотних коштів;

− звільнення персоналу магазинів від функцій управління, таких як поповнення запасів, вирішення питань ціноутворення та інших. Кількість персоналу магазинів може бути зведена до мінімуму, що дозволяє також істотно знизити витрати;

− централізоване управління транспортним господарством, що забезпечує максимальну оптимізацію маршрутів транспортних засобів та істотне скорочення транспортних витрат;

− забезпечує роботу всіх користувачів в реальному масштабі часу і істотне зниження витрат на персонал, необхідний для її обслуговування.

1.3 Постановка задачі

Метою роботи є розроблення веб-додатка моніторингу обсягів купівлі-продажу для мережі магазинів, головним призначенням якого є аналітичний

аналіз різних категорій товарів в інших магазинах і їх порівняння з продукцією власної мережі магазинів.

Таким чином, після проведення аналізу предметної області і з’ясування актуальності проєкту, були поставлені такі задачі:

1) обґрунтування вибору середовища розробки;
2) проєктування структури веб-сайта;
3) розроблення інтерфейсу;
4) створення концептуальної моделі бази даних;
5) розроблення веб-додатка моніторингу обсягів купівлі-продажу для мережі магазинів;
6) тестування роботи веб-додатка.

2 МЕТОДИ РОЗРОБЛЕННЯ ВЕБ-ДОДАТКА

2.1 Середовище розробки

Найпопулярніша служба мережі Інтернет, яка надає доступ до інформації незалежно від місця її розташування – World Wide Web, так звана всесвітня мережа або всесвітня павутина.

Всесвітню павутину утворюють понад мільйон web-серверів. Більшість ресурсів всесвітньої павутини є гіпертекстом, який призначений для перегляду інформації, отриманої від веб-сервера. Для полегшення створення, відображення і зберігання гіпертексту у Всесвітній павутині традиційно використовується мова HTML. Штучну комп'ютерну мову HTML не відносять до мов програмування, тобто вона не має можливості створювати динамічну функціональність. Натомість, це дозволяє організувати та форматувати документи, подібно до Microsoft Word [10].

Основними технологіями для створення веб-сторінок є HTML, CSS та скриптинг.

Структура статичної HTML – сторінки:

```
<!doctype HTML>
<html>
<head>
<title>HTML-документ</title>
</head>
<body>
Усім привіт!
</body>
</html>
```

HTML 5 була створена як єдина мова розмітки, яка могла б поєднувати синтаксичні норми HTML і XHTML. Воно покращує, розширює і раціоналізує розмітку документів. Однак сучасне застосування HTML5 дуже далеко від його початкової задачі, вадою є відсутність динамічності в сторінках. І тому, якщо потрібна динамічність, то в гру вступає PHP мова.

PHP – скриптова мова, що призначена для інтенсивного застосування при розробці веб-додатків. Мова PHP є одною з основних мов, які

використовуються для створення динамічних веб-сайтів, і у наш час підтримується більшістю хостинг-провайдерів.

Сценарій – це набір інструкцій програмування, які інтерпретуються під час виконання. Мова сценаріїв – це мова, яка інтерпретує сценарії під час виконання. Сценарії зазвичай вбудовуються в інші програмні середовища. Метою сценаріїв зазвичай є підвищення продуктивності або виконання рутинних завдань для програми.

PHP-скрипти можна інтерпретувати лише на сервері, на якому встановлено PHP. Для клієнтських комп'ютерів, що отримують доступ до скриптів PHP, потрібен лише веб-браузер.

PHP має вагомі плюси для веб-розробки, починаючи з того, що код – відкритий і безкоштовний. Незалежність від платформи, підтримка усіх серверів, доступ до підтримки, ефективність, рентабельність, – це далеко не всі переваги PHP.

Код PHP традиційний, тому що більшою мірою схожий на Pascal або C. Це неабияк полегшує вивчення PHP новачкам. Також мова PHP поєднує переваги C, Perl і націлена спеціально на роботу в Internet. Переконливою перевагою є простота. PHP починає обробляти файл, коли знаходить відкриваючі та закриваючі теги (?>), що вказує PHP, коли починати, а коли закінчувати обробку коду між ними.

До того ж мова PHP є найбільш безпечним способом розроблення веб-сайтів і веб-додатків, оскільки має високий рівень безпеки для захисту від вірусів і загроз.

Також важливою частиною веб-додатка є обрана система управління базами даних. Навіть не важливо, наскільки добре розроблений код програми і інтерфейс, якщо додаток не здатен швидко отримати, обробити та надати інформацію. Крім того, всі ці дані повинні бути захищені, щоб зловмисники не могли їх отримати. Це можна досягти правильним вибором СУБД.

На ринку існує понад 300 систем управління базами даних. Вибір між такою кількістю інструментів справді надзвичайний. Розглянемо MySQL як найяскравіший приклад реляційних баз даних SQL.

Реляційна база даних ідеально підходить для зберігання структурованих даних (поштові індекси, номери кредитних карт, дати, ідентифікаційні номери). SQL – це зріла технологія: бази даних добре задокументовані, мають велику підтримку та добре працюють із більшістю сучасних середовищ та бібліотек.

Ще одна велика перевага реляційних баз даних – їх безпека. Реляційні бази даних підтримують дозволи доступу, які визначають, кому дозволено читати та редагувати базу даних. Адміністратор бази даних може надати певному користувачу привілеї для доступу, вибору, вставки або видалення даних. Це не дає шансів третім особам викрасти інформацію [11].

Доступ до баз даних MySQL мають всі мови програмування, в яких присутній API (англ. application programming interface).

Інтерфейс прикладного програмування API – це код, який дозволяє двом програмам взаємодіяти між собою. Він також містить умови обміну даними. API-інтерфейси є рівнем абстракції між двома системами, приховуючи складність і робочі деталі останньої. Використовується програмістами при написанні різних додатків.

API застосовують різні системи. API баз даних забезпечують спілкування між додатком та СУБД. Розробники працюють з базами даних, записуючи запити на доступ до даних, змінюючи таблиці тощо.

Простіше кажучи, API надає відповідь користувача системі та надсилає відповідь системи користувачу. API допомагає розробникам швидко доставляти інформацію споживачам і використовується щодня у сучасному світі. Покупки в Internet, перегляд додатків у соціальних мережах або гра на смартфоні. Кожного разу при відвідуванні сторінки в Internet відбувається взаємодія з API.

API також надає широкий набір функцій, а тому часто може використовуватись для малювання вікон на екрані чи іконок. Хороший API полегшує розробку програми, надаючи всі складові. Потім програміст лише з'єднує блоки. API – це потужний інструмент, який може допомогти прискорити різні бізнес-операції, розширити охоплення бренду, підключити покупців до потрібних продуктів та багато іншого.

Об'єм підвантажуваних даних, а також кількість запитів до сервера, можна зменшити за допомогою AJAX (Asynchronous Javascript and XML) – це певна технологія, що дозволяє без перевантаження самої сторінки підвантажувати ряд даних на веб-сторінці.

У класичній моделі веб-додатків користувач звертається до сервера за допомогою HTTP-запиту через браузер. Сервер отримує запит користувача та виконує необхідні дії – взаємодіє зі сховищами даних, потім здійснює обробку, спілкуючись з різними успадкованими системами тощо. У результаті сервер видає HTML сторінку клієнта.

Для опису зовнішнього вигляду документа використовують формальну мову CSS із використанням мови розмітки.

CSS є кращим способом налаштування зовнішнього вигляду веб-сайта. Таблиці стилів визначають колір, розмір, шрифти, положення тексту та інших тегів HTML, тоді як файли HTML визначають вміст та спосіб його організації. Розділення їх дозволяє, наприклад, змінити колірну гамму без необхідності переписувати весь веб-сайт. Такий поділ збільшує доступність документа, надає велику гнучкість, зменшує повторюваність і складність в структурному вмісті.

Каскад означає, що стиль, застосований до батьківського елемента, також застосовуватиметься до всіх дочірніх елементів у батьківському. Наприклад, встановлення для тіла кольору тексту означатиме, що всі заголовки та абзаци всередині тіла також будуть однакового кольору.

CSS надає веб-розробникам більш точний контроль над тим, як виглядатимуть веб-сторінки, ніж HTML. Саме тому сьогодні більшість веб-сторінок містять каскадні таблиці стилів.

Із розвитком технологій функціональність веб-сторінок постійно зростає й наближається до функціональності настільних прикладних програм. Ця зростаюча функціональність реалізується за допомогою JavaScript [12].

JavaScript – це динамічна мова програмування. Вона легка і найчастіше використовується у складі веб-сторінок, реалізація яких дозволяє сценарію на стороні клієнта взаємодіяти з користувачем та створювати динамічні сторінки. Це інтерпретована мова програмування з об’єктно-орієнтованими можливостями. Основні архітектурні особливості: динамічне введення тексту, слабка типізація, автоматичне управління пам’яттю, програмування прототипів, функції як об’єкти першого класу.

Немає сенсу самостійно прописувати всі необхідні ефекти, коли для цього вже написано кілька сотень бібліотек, що допомагають реалізовувати ці ефекти. І серед безлічі цих бібліотек найбільш доступною та зручною для розуміння являється бібліотека jQuery.

Обсяг програмного коду jQuery менше, ніж обсяг стандартного коду JavaScript, завдяки чому скорочуються часові витрати на розробку елементів веб-сторінки. У порівнянні з JavaScript - програмний код більш зрозумілий.

jQuery – це фреймворк, розроблений на JavaScript із метою спрощення написання об'ємного коду. Бібліотека jQuery містить в собі велику кількість вже прописаних функцій, що дозволяють якісно та швидко створювати інтерактивні сторінки веб-сайту. Використовують jQuery як в програмуванні елементів веб-сторінок, так і в створенні різного типу додатків.

Для написання швидкого односторінкового додатка краще всього обрати один з популярних js-фреймворків. Фреймворки JavaScript популярні серед розробників завдяки таким перевагам як ефективність, безпека та вартість.

Bootstrap – це величезний набір зручних, багаторазових інструментів коду, що містить HTML, CSS і додаткові розширення JavaScript. Також це фронтальна структура розробки, яка дозволяє розробникам та дизайнерам швидко створювати гнучкі, динамічні, адаптивні веб-сайти чи веб-додатки.

Bootstrap став важливим інструментом для front-end розробників.

Офіційний веб-сайт Bootstrap описує його як: «Найпопулярніші рамки HTML, CSS та JS для розробки чуйних, мобільних перших проєктів в Internet».

Bootstrap заощаджує програміста від написання великої кількості CSS-коду, даючи більше часу витратити на розробку веб-сторінок, до того ж, безкоштовно.

Основні переваги Bootstrap:

1. Реактивна сітка. Більше не витрачаються зайві години на кодування власної сітки – Bootstrap задається із наперед готовою власною сіткою. Тепер є можливість перейти безпосередньо до наповнення контейнерів вмістом.

2. Адаптивні зображення. Bootstrap має власний код для автоматичної зміни розміру зображень на основі поточного розміру екрана.

3. Компоненти. Bootstrap містить цілу низку компонентів, які можна легко застосувати до веб-сторінки, включаючи: навігаційні панелі, випадаючі списки, індикатори прогресу, ескізи тощо.

4. Bootstrap’s JavaScript. У розробників є можливість скористатися понад десятком користувацьких плагінів JQuery, що дає ще більше інтерактивності, пропонуючи прості рішення для модальних спливаючих вікон, переходів, каруселей зображень та плагіна під назвою scrollspy, який автоматично оновлює панель навігації під час прокрутки сторінки.

5. Документація. Кожен фрагмент коду описується та пояснюється чітко на офіційному веб-сайті. Пояснення включають у себе зразки коду для базової реалізації, спрощуючи процес навіть для початківця. Все, що потрібно зробити розробнику – це вибрати компонент, скопіювати та вставити код на свою сторінку та налаштувати звідти.

6. Налаштування. Одним із основних і критичних питань, що стосується Bootstrap, є розмір. Це дійсно може уповільнити програму при першому завантаженні. Наприклад, поточна версія CSS-файлу Bootstrap – це цілих 119 Кб. Хоча це може не виглядати особливо великим порівняно з файлами зображень та відео, для CSS-файлів.

Bootstrap дозволяє боротися з цим. Перейшовши на сторінку налаштування та завантаження цих файлів, є можливість перевірити непотрібні функції, що зменшить розмір файлу та заощадить користувачам додатковий час завантаження.

7. Спільнота Bootstrap. Як і у багатьох проєктів з відкритим кодом, Bootstrap має велике співтовариство дизайнерів та розробників. Розміщення в GitHub розробникам полегшує зміни та доповнення до кодової бази Bootstrap. Це також дозволяє людям легко співпрацювати, надавати свої поради та взаємодіяти з однолітками та іншими користувачами.

8. Зовнішні шаблони Bootstrap. По мірі зростання популярності Bootstrap люди почали створювати шаблони на його основі, щоб ще більше прискорити процес розробки веб-сторінок. Існує багато веб-сайтів, присвячених обміну та купівлі спеціальних шаблонів на основі Bootstrap.

2.2 Проєктування структури веб-сайта

Правильно організована структура необхідна для зручного користування веб-додатком і пошуку інформації.

Веб-сторінки різних додатків можуть виглядати по-різному, але усі вони мають тенденцію використовувати одні й ті самі стандартні компоненти, якщо тільки у неї немає спеціального призначення (повноекранна гра, відео тощо): заголовок, панель навігації, основний зміст, бічна панель, колонтитул. Веб-додаток – це структурована сукупність таких сторінок, розділів та навігації та зв’язок між ними. Структуру веб-додатка розглядають на зовнішньому і логічному рівнях.

Зовнішня структура визначає, як будуть розташовані основні значущі елементи на кожній сторінці (зміст, меню, авторизація, пошук, головний контент і т.п.).

У кожного сайта є «шапка», де зазвичай розміщують заголовок (підзаголовок) сайта, логотип сайта, навігаційне меню, де розташовані назви рубрик, міток та інша корисна інформація, контентна частина сайта, де публікується основна інформація та футер («підвал»), де показана деяка службова інформація.

Зовнішня структура веб-додатка розміщена на рисунку 2.1.

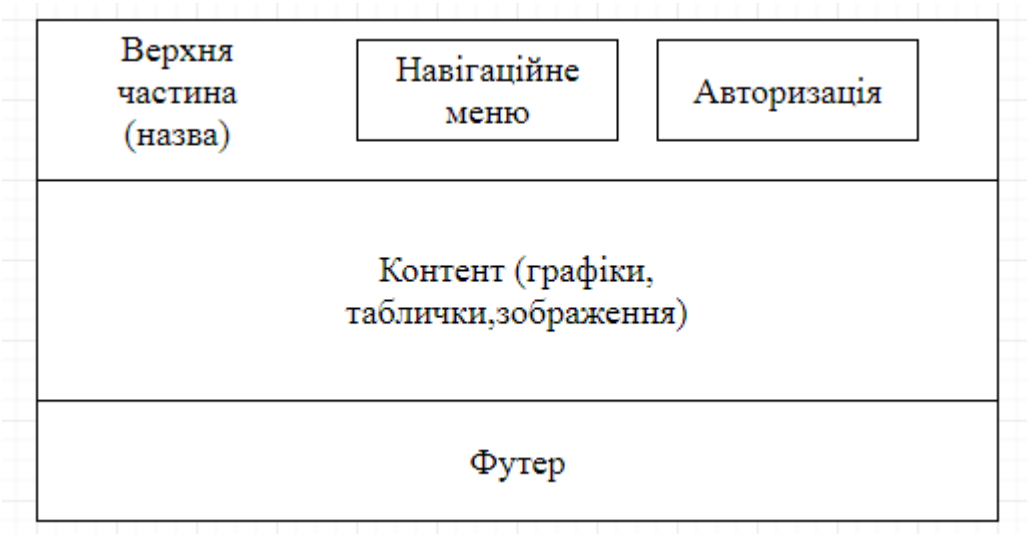

Рисунок 2.1 – Представлення зовнішньої структури веб-додатка

На початковому етапі проєкта перед тим, як приступати до роботи по візуальному дизайну, корисно розробити логічну структуру сайта. Чітко визначена структура дає уявлення про кількість рівнів інформації та задіяний контент. Отже, структура сайта представляє собою комплекс логічно пов’язаних між собою веб-сторінок з унікальним дизайном. Вони об’єднані посиланнями і виконані в одному стилі. Затребуваність сайта залежить від успішного дизайна, який приваблює користувачів і робить пошук ними потрібної інформації зручним і приємним.

Структура навігації сайта надзвичайно впливає на поведінку користувачів. Якщо користувачі не розуміють, де швидко знайти потрібну їм інформацію, вони не будуть користуватись таким ресурсом. Гарна навігація сайта означає, що користувачам абсолютно зрозуміло, де вони знаходяться, як розміщені елементи додатка і як ними користуватися. Правильно реалізована логічна структура сайта дозволяє користувачу з легкістю переходити за посиланнями і бути впевненим в тому, що можна буде легко повернутися до попередніх сторінок.

Структура веб-додатка визначається схемою навігації усередині сайта. Розрізняють звичайну лінійну структуру, ієрархічну, мережну і комбіновану.

Лінійна навігація є найпростішою структурою. У цій структурі кожна сторінка просто зв’язана з попередньою. Лінійну структуру доцільно використовувати тільки якщо на сайті мало сторінок.

Ієрархічна структура передбачає застосування головної сторінки, на якій розміщено меню з посиланнями на розділи веб-додатка, що розташовані на наступних сторінках. Розділи можуть містити в собі посилання на підрозділи чи іншу детальну інформацію. Ієрархічна структура передбачає, що кожна сторінка (за винятком головної), є підрозділом для сторінки більш високого рівня. Початком для такої структури є головна сторінка, з якої найчастіше починається перегляд структури. Але у неї немає кінця, відвідувач сайта може спускатися і підніматися по ієрархічних рівнях структури і проглядати сторінки одну за одною. Цей вид структури найкраще підходить для організації різнорідного і складно упорядкованого матеріалу, наприклад, каталогів, збірників статей, підрозділів організації.

Мережна структура дозволяє користувачам слідкувати за власним потоком інформації, він може бути унікальним для кожного користувача. Для досягнення зручного переходу між логічно пов’язаними сторінками має бути велика кількість посилань між окремими сторінками. Тому використання мережної моделі для сайтів з великою кількістю сторінок стає менш корисним.

Найчастіше розробники використовують складену навігаційну структуру, яка поєднує різні елементи інших структур і є найменш обмеженою. Вона може бути частково лінійною, частково ієрархічною, і особливо корисна, коли ресурс містить багато тем і підтем. При цьому користувачі вільно переходять між сторінками, а для насправді сторінки мають лінійну або мережеву структуру різних частин сайта.

Важливим моментом при розробці логічної структури є розподіл інформаційного матеріалу сайта по окремих сторінках. Розподіл має бути логічним, кожна сторінка повинна бути присвячена одній темі, і кожна тема повинна займати лише одну відведену під неї сторінку.

При розробленні сайта було обрано мережну структуру. Структуру сайта зображено на рисунку 2.2.

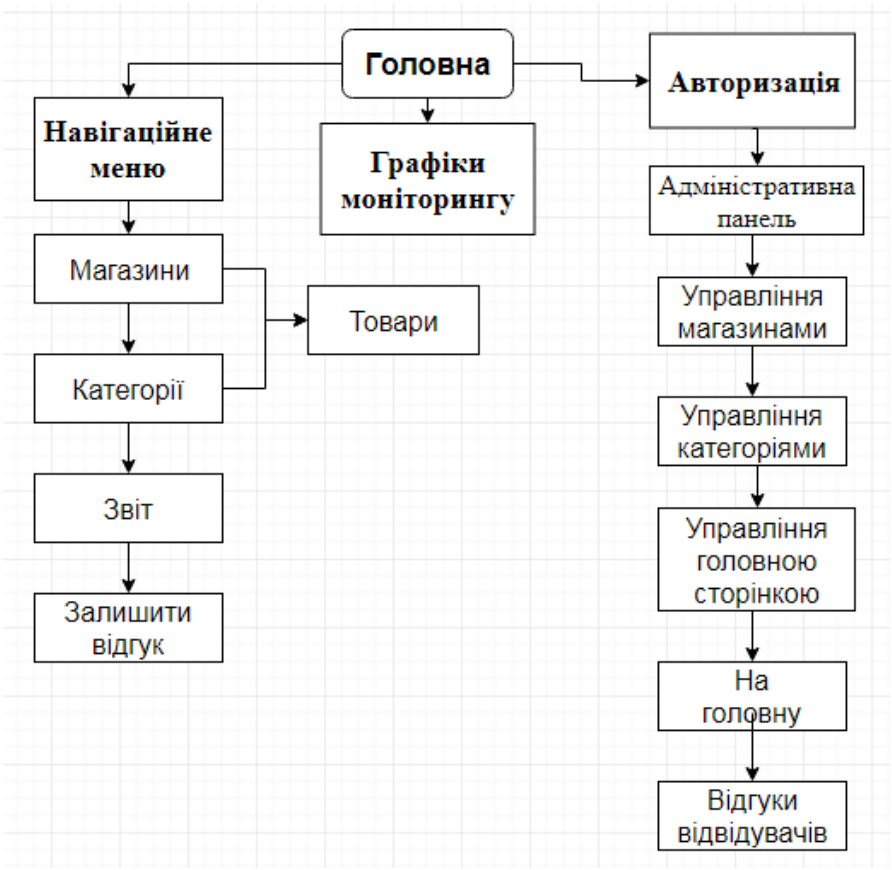

Рисунок 2.2 – Структура веб-додатка

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ

3.1 Розроблення інтерфейсу

Будь-який сайт має свій власний інтерфейс. Це його візуальне відображення на екрані монітора комп'ютера і це є один з найбільш важливих елементів веб-сайта. Саме інтерфейс відповідає за те, наскільки зручно користувачеві буде взаємодіяти з сайтом. Інтерфейс сайта повинен бути простим, зрозумілим, і найголовніше, кросбраузерним. Будь який веб-сайт, який прагне отримати значну користувацьку базу, повинен однаково відображатися у різних браузерах та їх версіях і мати ідентичну функціональність.

При виконанні дипломної роботи інтерфейс сайта було створено із застосуванням мови розмітки HTML, каскадних таблиць стилів CSS.

При створенні шаблону сайта був використаний фреймворк Bootstrap 4.

Для повної функціональності та зручності користування інформаційної системи було розроблено 2 інтерфейси. Перший – для всіх користувачів сайта як головна сторінка, другий – сторінка менеджера для роботи з накладними.

Усі сторінки сайта розбиті на такі основні частини: верхню, основну (контент) та нижню.

У верхній частині головної сторінки розташована «шапка» сайта, яка дублюється на всіх інших сторінках, окрім сторінки, яка інформує користувача про успішну відправку відгуку. Вона створюється за допомогою HTML та PHP, а також компонента bootstrap 4 navbar. Верхня частина містить назву веб-додатка, вибір магазинів, вибір категорій, посилання на сторінку звітів, посилання на сторінку для написання відгуку, привітання та посилання на сторінку для авторизації адміністратора (рис. 3.1).

«Шапка» веб-додатка для відвідувачів відрізняється від «шапки» адміністративної панелі. Вона також створена за допомогою HTML та PHP. Містить такі пункти меню, як назва веб-додатка, налаштування магазинів,

категорій та перегляд відгуків, привітання та посилання на сторінку для авторизації (рис. 3.2).

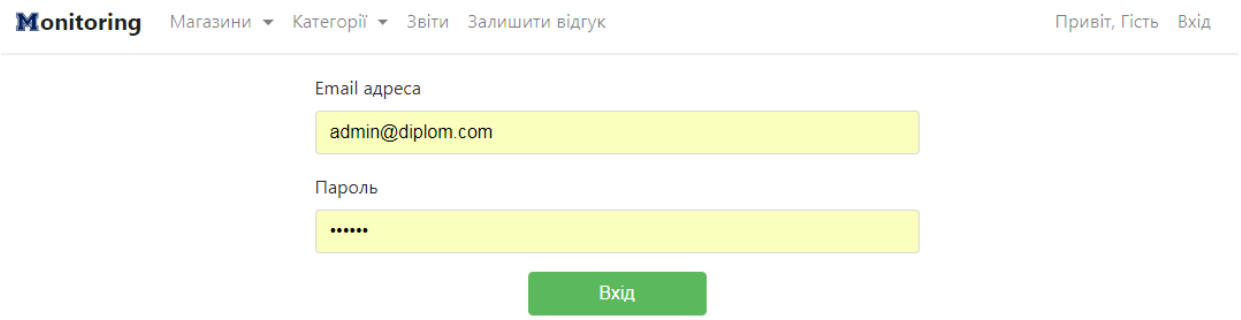

Рисунок 3.1 – «Шапка» веб-сайта і панель авторизації

Рисунок 3.2 – «Шапка» адміністративної панелі веб-сайта

На футері веб-додатка відображений блок із інформацією про авторські права (рис. 3.3).

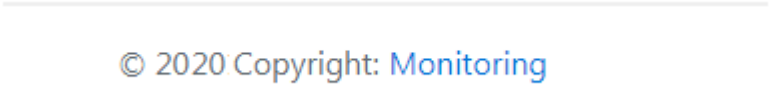

Рисунок 3.3 – Футер веб-додатка

На контентній частині веб-додатка для відвідувача відображаються ті графіки, які обрав адміністратор для перегляду. Для їх побудови використовується JavaScript та бібліотека CanvasJS.

Наприклад, оберемо графік, який показує сумарний прибуток від реалізації товару за кожний місяць за вибраним користувачем найменуванням товару за певний час (рис. 3.4).

Даний сайт орієнтований на потенційних клієнтів та адміністратора. Кожен із них виконує свої функції.

Для управління контентом на сайті, розроблена адміністративна панель, яка надає можливість адміністратору додавати, видаляти та змінювати її без

участі виконавця (програміста). Адміністратор може обирати, які категорії будуть відображатися для користувачів, проглядати відгуки, налаштовувати інформацію про магазини та інформацію на головній сторінці (рис. 3.5). Для створення кожної картки був використаний компонент bootstrap 4 jumbotron.

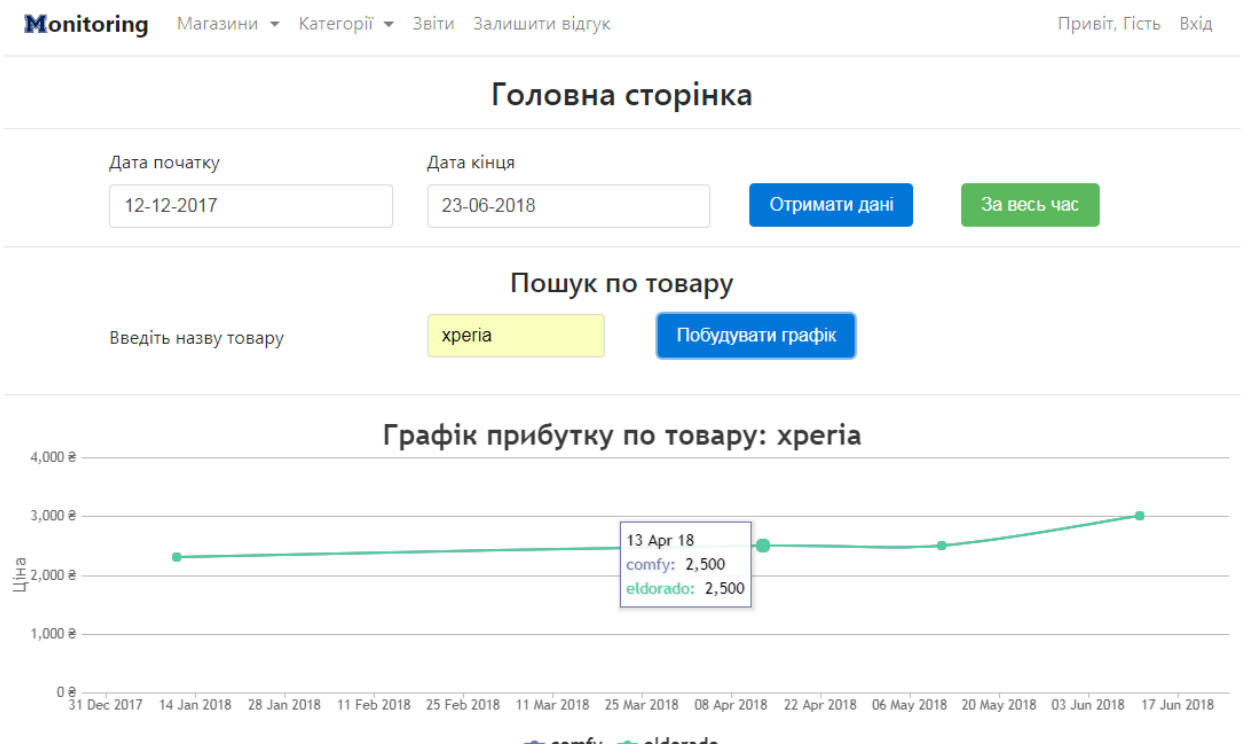

Рисунок 3.4 – Прибуток по певному товару

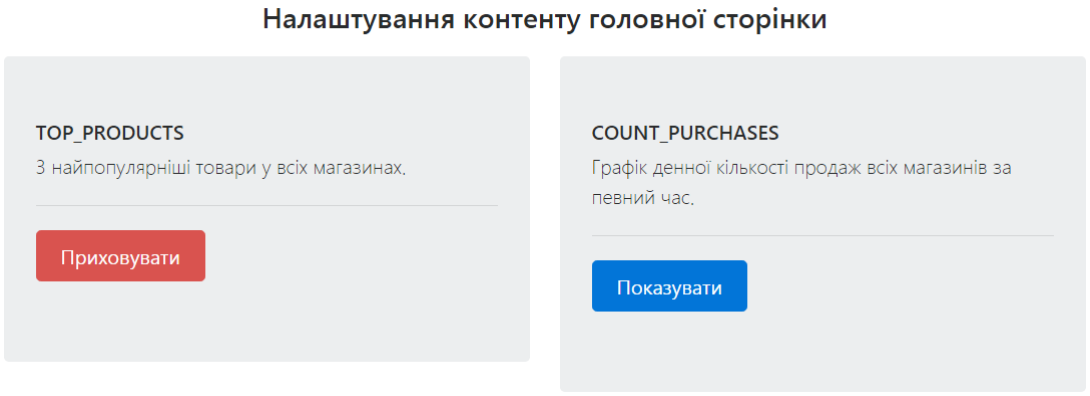

Рисунок 3.5 – Налаштування контенту головної сторінки

При переході на сторінку «Магазини» відображається список магазинів (рис. 3.6) та можливість налаштування інформації про них (рис. 3.7).

Відображення списку магазинів здійснюється за допомогою компонента bootstrap 4 table.

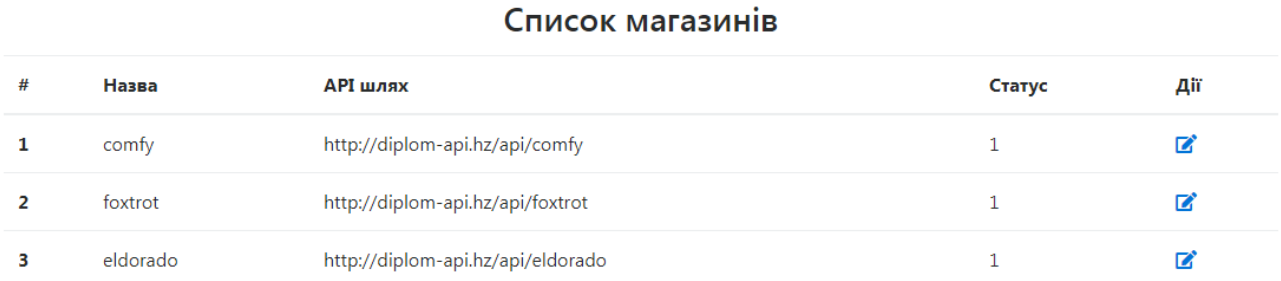

Рисунок 3.6 – Інформація про магазини

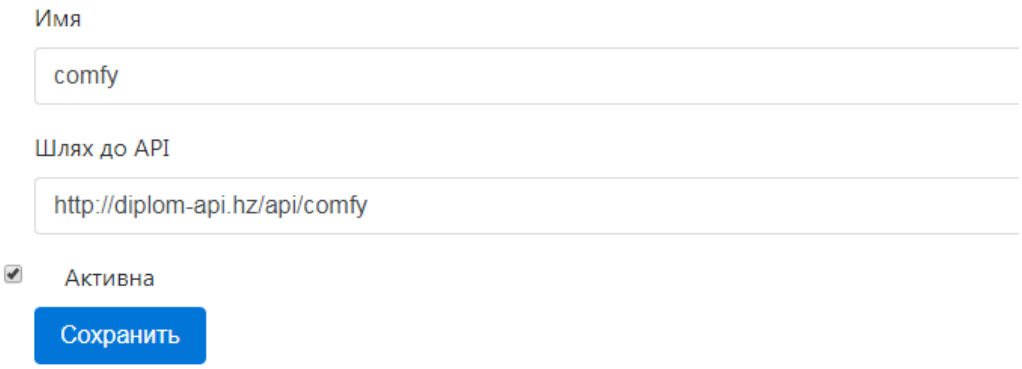

Рисунок 3.7 – Налаштування інформації про магазин

При переході на сторінку «Категорії» відображаються всі категорії кожного магазину та можливість додати їх до обраних категорій (рис. 3.8).

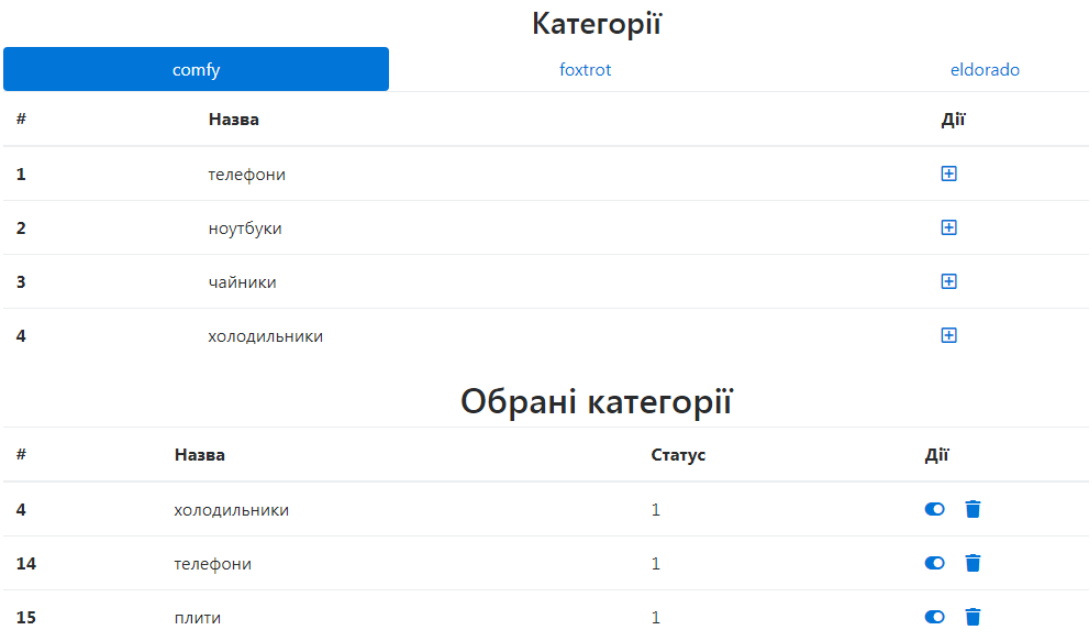

Рисунок 3.8 – Налаштування категорій

В обраних категоріях знаходяться ті категорії, які будуть відображатися у меню користувача. Відображення інформації по кожному магазину виконується за допомогою компонента bootstrap 4 tabs.

При переході на сторінку «Відгуки» відображаються відгуки від користувачів та інформація про них (рис. 3.9). За відображення кожного відгуку відповідає компонент bootstrap 4 card.

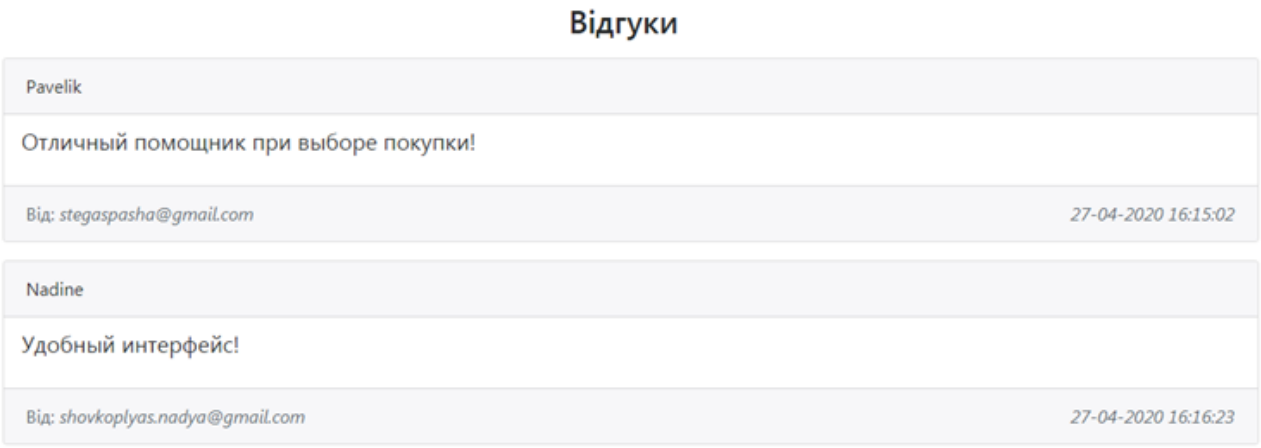

Рисунок 3.9 – Перегляд відгуків

На сторінці користувача є такі пункти панелі, як вибір магазинів, вибір категорій, посилання на сторінку звітів, посилання на сторінку для написання

відгуку. При переході на сторінку «Магазини» відображаються товари та інформація про них (рис. 3.10).

Щоб переглянути інформацію про кількість продажу та ціни конкретного товару в конкретному магазині, треба натиснути на кнопку з іконкою корзини (рис. 3.11). Для відображення всіх іконок використовуємо бібліотеку fontawesome. Для побудови графіків використовується JavaScript та бібліотека CanvasJS.

Щоб переглянути інформацію про кількість продажу та ціни конкретного товару у всіх магазинах треба натиснути на кнопку із зображенням посудини (рис. 3.12). Відображення основної інформації по товару виконується за допомогою компонента bootstrap 4 card, а для перегортання інформації по кожному магазину – компонент bootstrap 4 carousel.

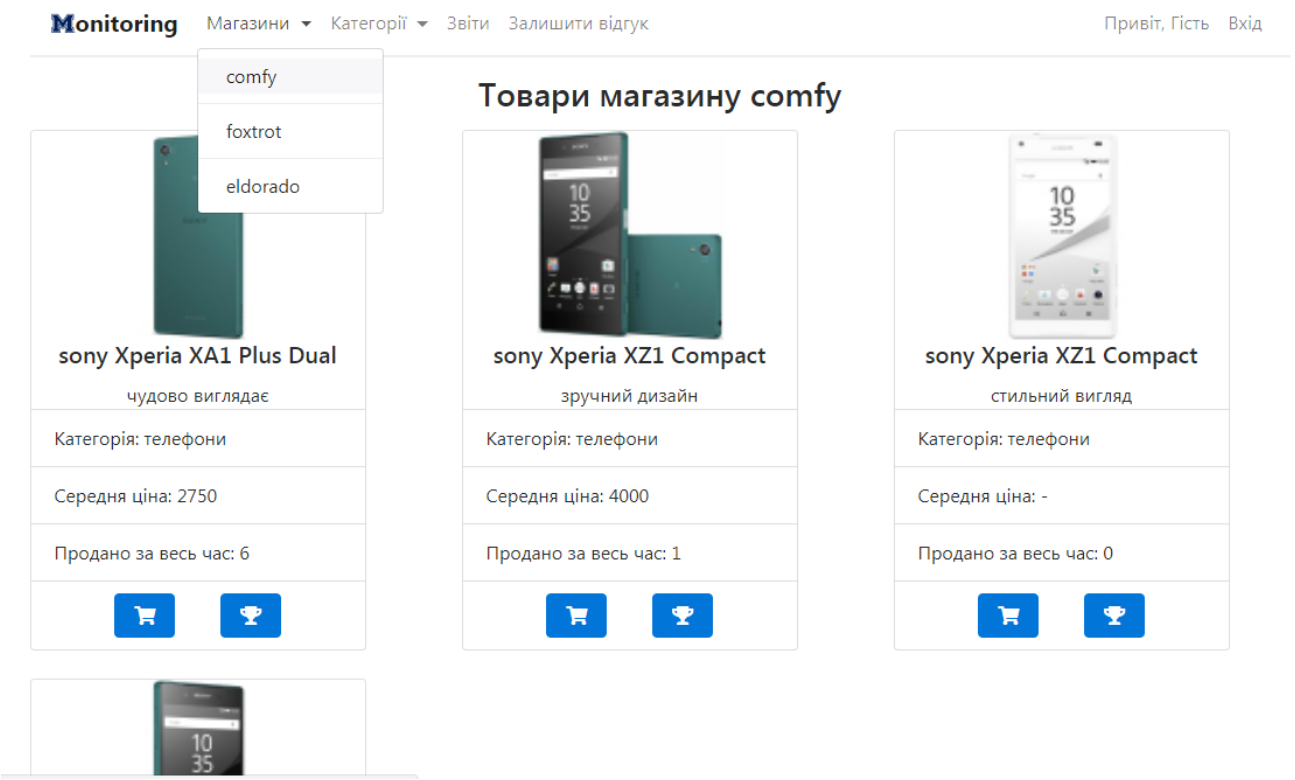

Рисунок 3.10 – Товари магазину

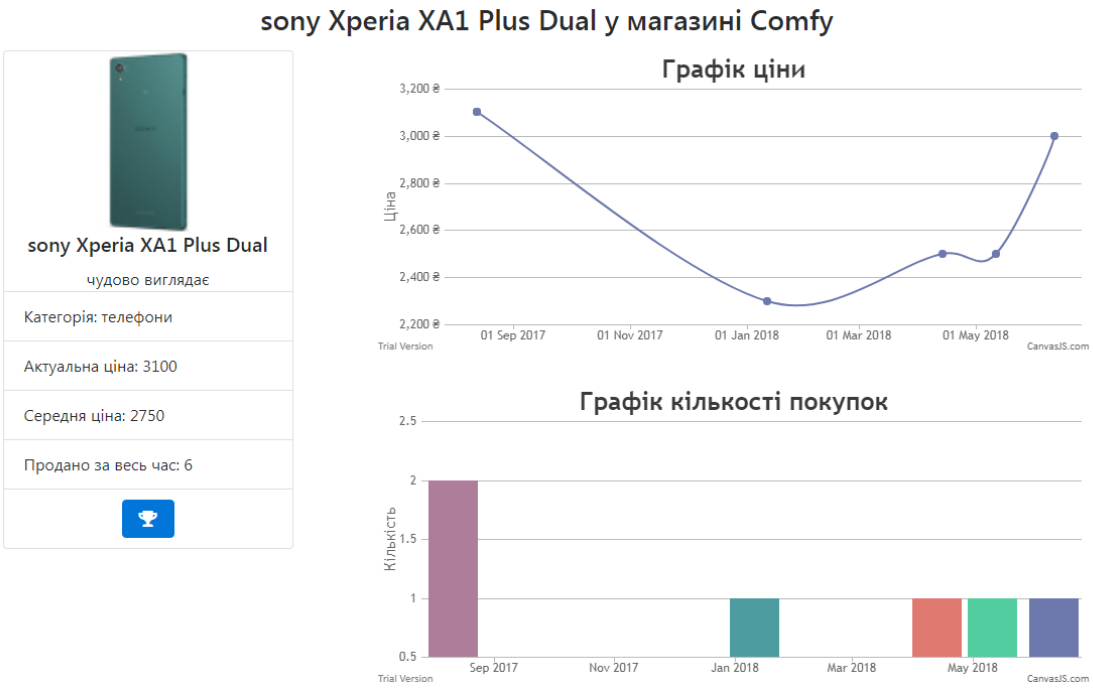

Рисунок 3.11 – Інформація про конкретний товар у конкретному магазині

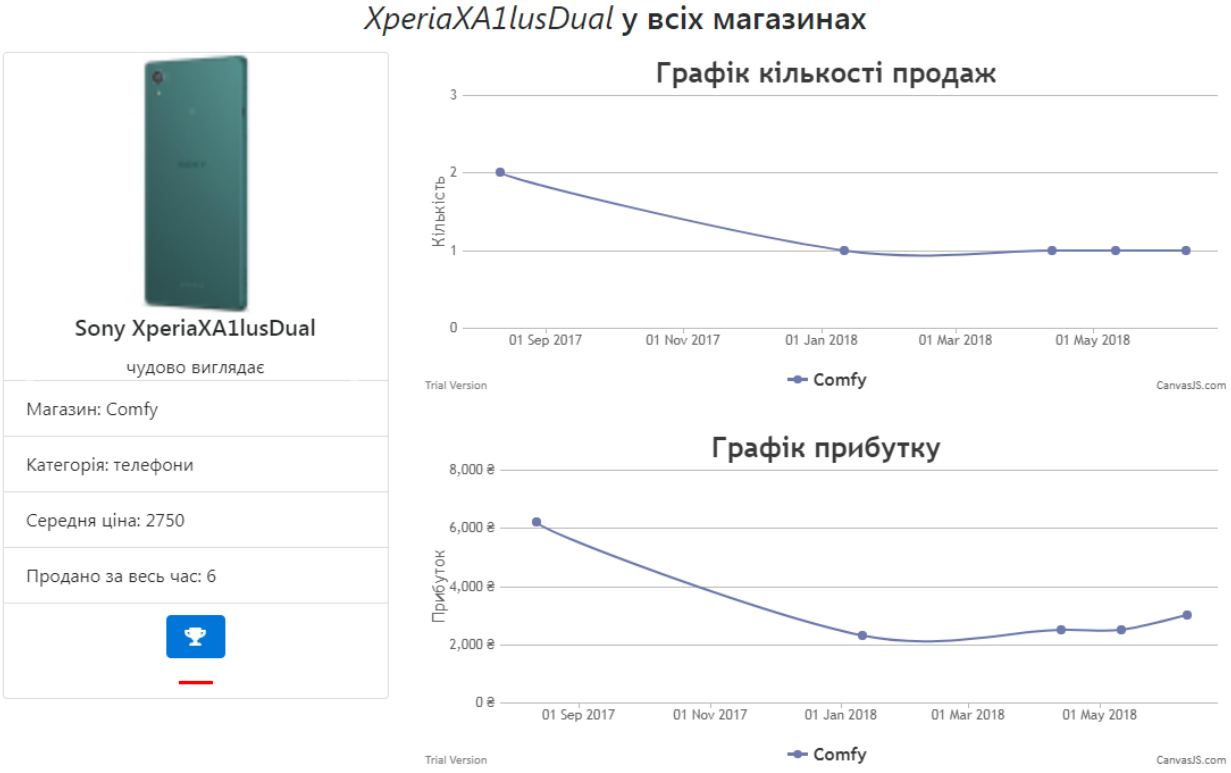

Рисунок 3.12 – Інформація про конкретний товар у всіх магазинах

При переході на сторінку «Категорії» відображаються всі товари з магазинів, у яких є ця категорія (рис. 3.13). Для відображення кожної вкладки

з товарами магазину був використаний компонент bootstrap 4 tabs. Для відображення інформації про самі товари – компонент bootstrap 4 card.

При переході на сторінку «Звіти» користувач має можливість завантажити звіт собі на комп'ютер у форматі .csv (рис. 3.14). Для відображення кожного блока з інформацією про звіт використовується компонент bootstrap 4 card.

При переході на сторінку «Залишити відгук» користувач має можливість залишити свій відгук (рис. 3.15). На цій сторінці було використано звичайну html-форму, яка стилізована за допомогою css-правил, описаних у bootstrap 4. Кожний input має клас form-control та розташований разом з label у блоці, який має клас form-group.

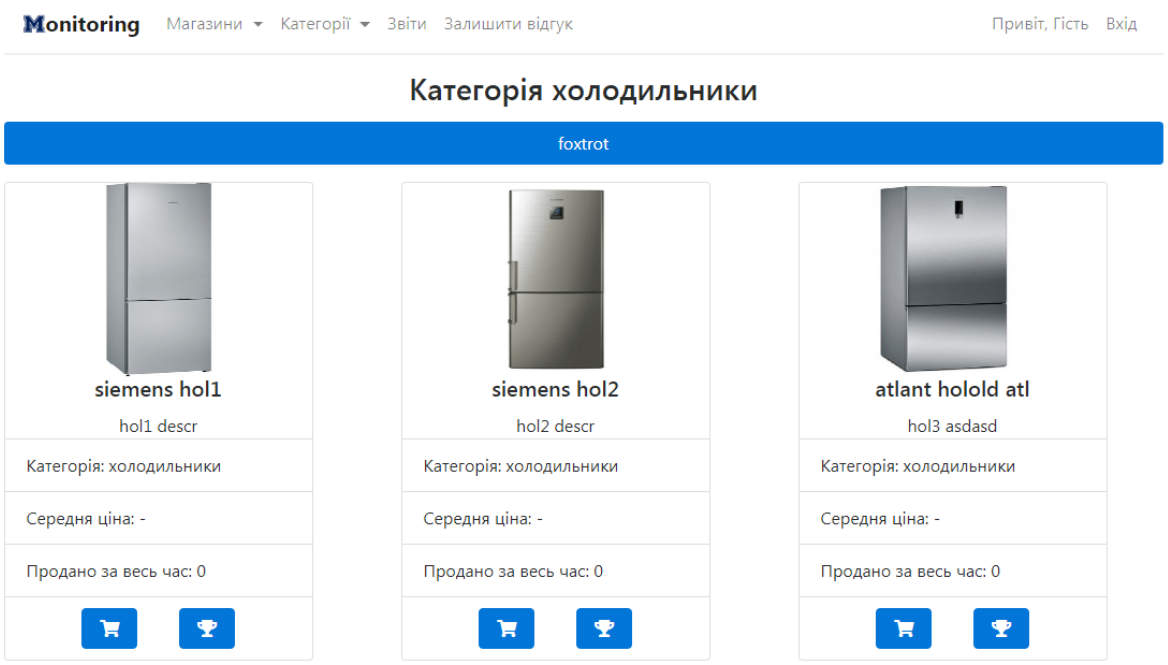

Рисунок 3.13 – Категорії

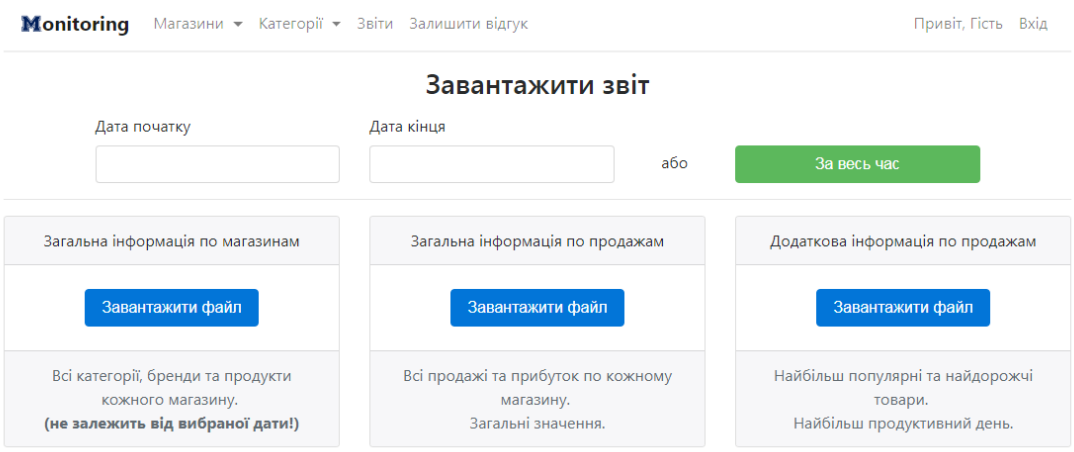

Рисунок 3.14 – Завантаження звіту

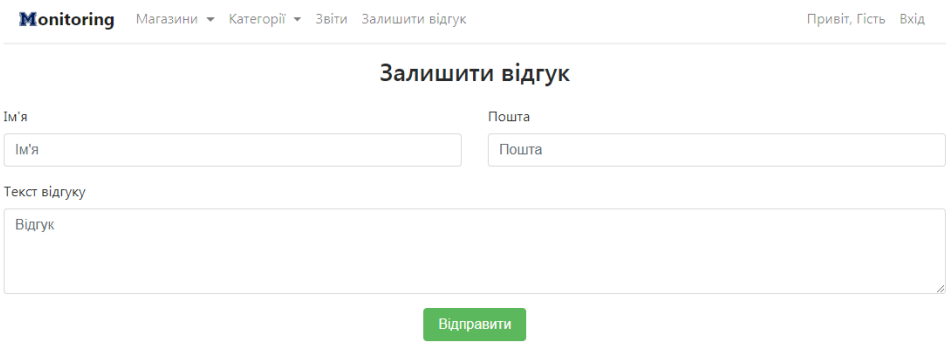

Рисунок 3.15 – Форма для відправки відгуку

3.2 Розроблення бази даних

У веб-додатку використовується система управління реляційними базами даних MySQL, що надає багатий набір функціональних можливостей, які підтримують безпечне середовище для зберігання, обслуговування і отримання даних.

«Diplom» є основною базою даних веб-додатка, концептуальна схема зображена на рисунку 3.16.

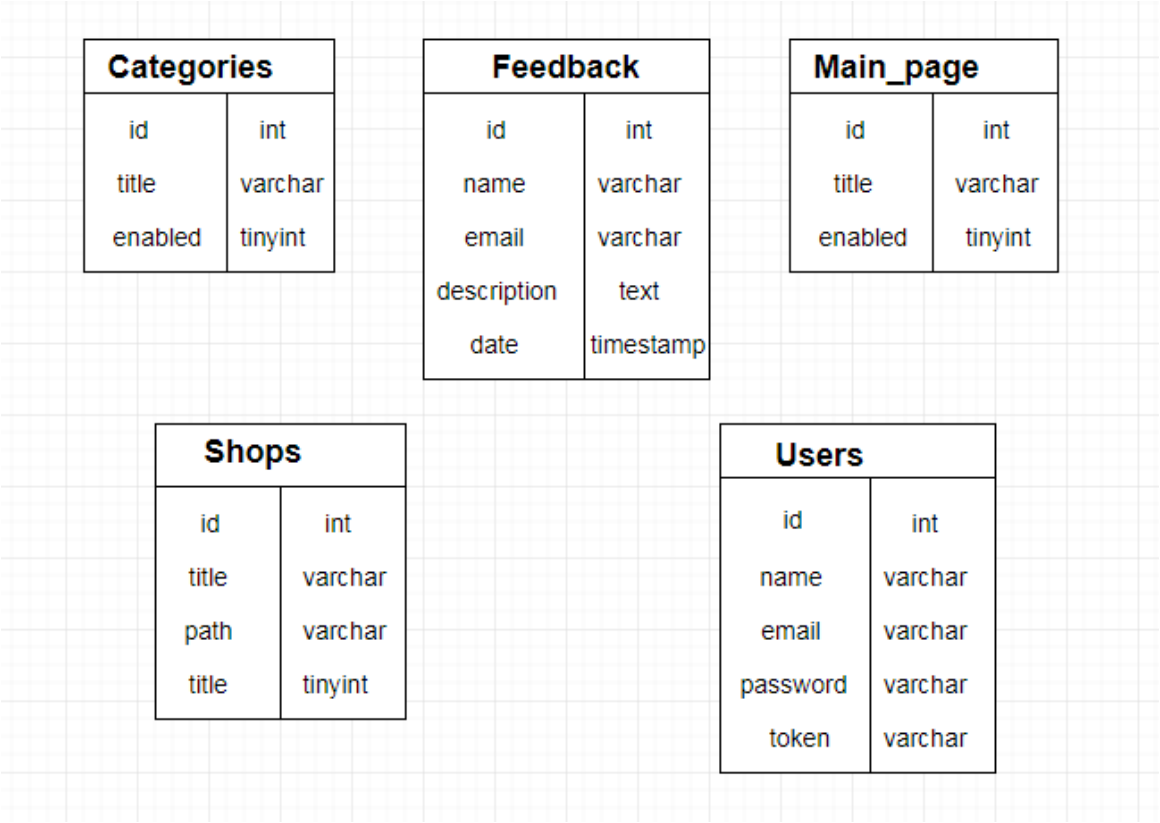

Рисунок 3.16 – Концептуальна схема бази даних «Diplom»

Таблиця categories – категорії товарів, які будуть відображатися в меню навігації. Таблиця наведена на рисунку 3.17 та має такі поля:

id – унікальній ідентифікатор,

title – назва категорії,

enabled – дає можливість відключати або включати категорії. Якщо поле набуває значення 0, то категорія не активна, якщо 1 – навпаки.

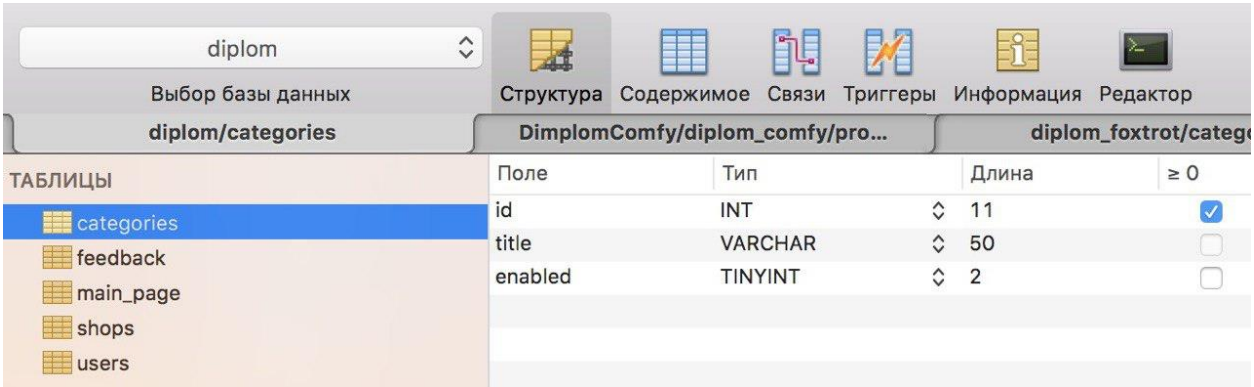

Рисунок 3.17 – Таблиця categories

Таблиця feedback – зберігає в собі відгуки про інтернет-додаток від людей, які його відвідали. Таблиця наведена на рисунку 3.18 та має такі поля:

id – унікальний ідентифікатор,

name – ім’я людини, яка залишила відгук,

email –електронна адреса людини, яка залишила відгук,

description – текст відгуку,

date – дата, коли залишили відгук.

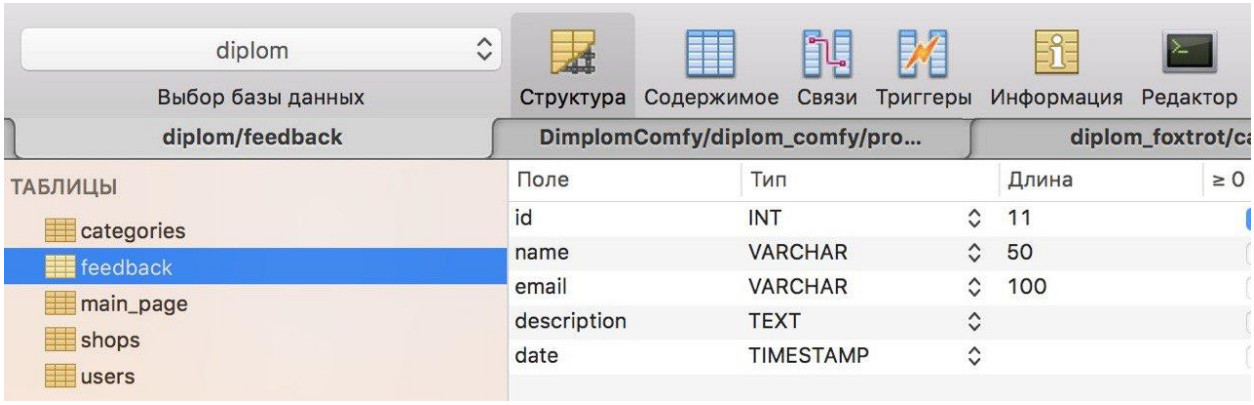

Рисунок 3.18 – Таблиця feedback

Таблиця main_page – зберігає в собі контент головної сторінки (графіки). Таблиця наведена на рисунку 3.19 та має такі поля:

id – унікальний ідентифікатор,

title – назви частин контенту,

description – опис частин контенту,

enabled – дає можливість відключати або включати частини контенту. Якщо поле набуває значення 1, то магазин активний і з нього можна брати дані, якщо 0 – навпаки.

Таблиця shops – зберігає в собі дані про всі магазини, які моніторить система (comfy,foxtrot …). Таблиця наведена на рисунку 3.20 та має такі поля:

id – унікальній ідентифікатор

title – назва магазину

path – шлях до API, по якому потрібно звертатися, щоб отримати потрібну інформацію.

enabled – дає можливість відключати або включати магазини. Якщо поле набуває значення 1, то магазин активний і з нього можна брати дані, якщо 0 – навпаки.

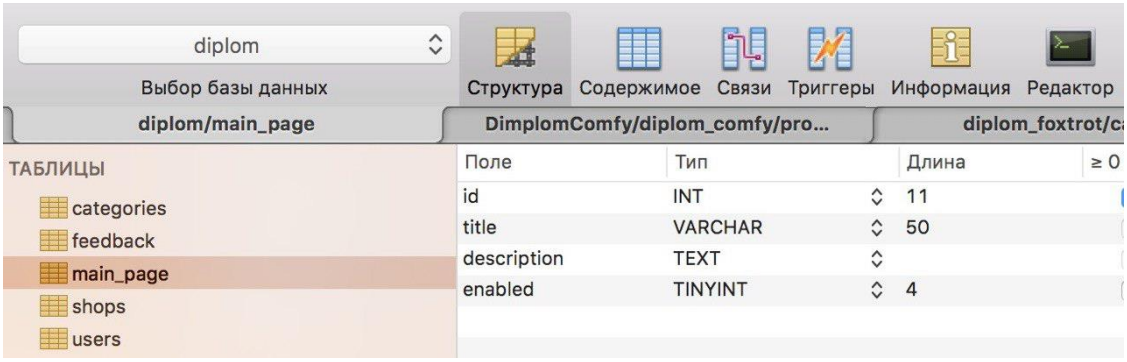

Рисунок 3.19 – Таблиця main_page

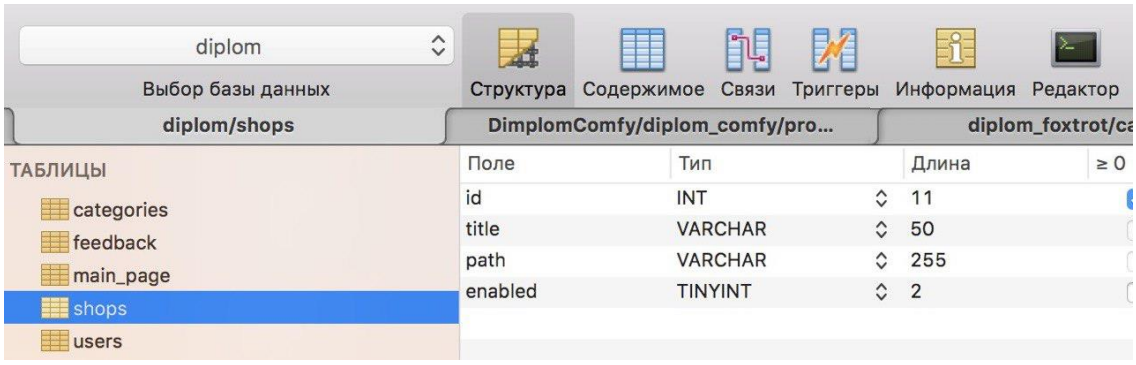

Рисунок 3.20 – Таблиця shops

Таблиця users – інформація про користувачів системи (адміністратор). Таблиця наведена на рисунку 3.21 та має такі поля:

id – унікальний ідентифікатор,

name – ім’я користувача,

email – електронна адреса користувача, що являється його логіном при вході до системи,

password – пароль у зашифрованому виді (алгоритм MD5),

token – унікальний номер сесії користувача. Коли користувач заходить під своїм логіном і паролем, система дає йому унікальний набір символів і зберігає його в браузері і коли той звертається до сайта, сайт пам’ятає, що цей користувач вже заходив до системи.

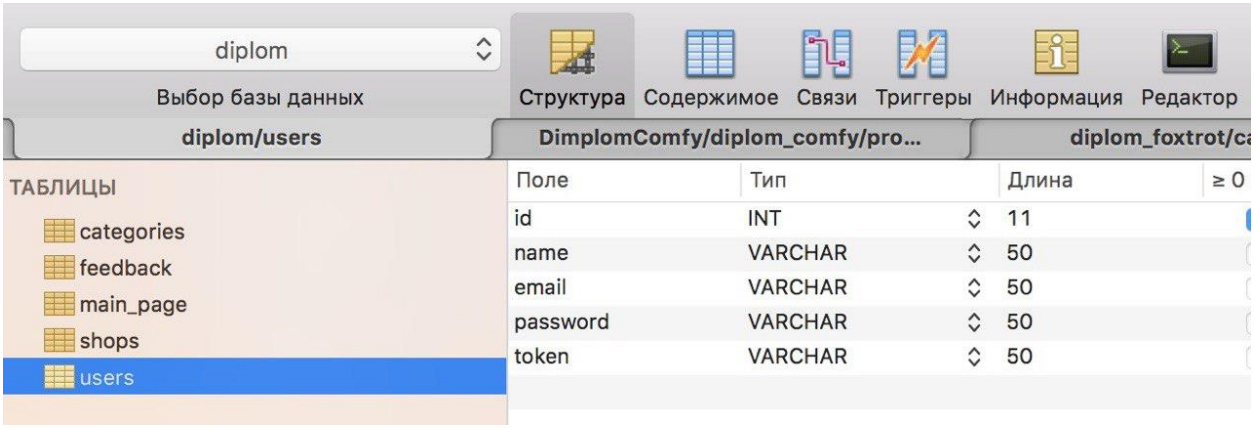

Рисунок 3.21 – Таблиця users

База даних «diplom_comfy» – це імітація справжньої БД інтернет-магазину Comfy, через спеціальний прошарок (API), що дає змогу з веб-додатку звертатися в цю БД і брати потрібну інформацію.

База даних використовує кілька взаємопов’язаних таблиць. Між ними встановлені зв’язки (рис. 3.22).

Таблиця brands – бренди яким належать товари (sony, apple, samsung, braun…). Таблиця наведена на рисунку 3.23 та має такі поля:

id – унікальний ідентифікатор,

title – назва бренда.

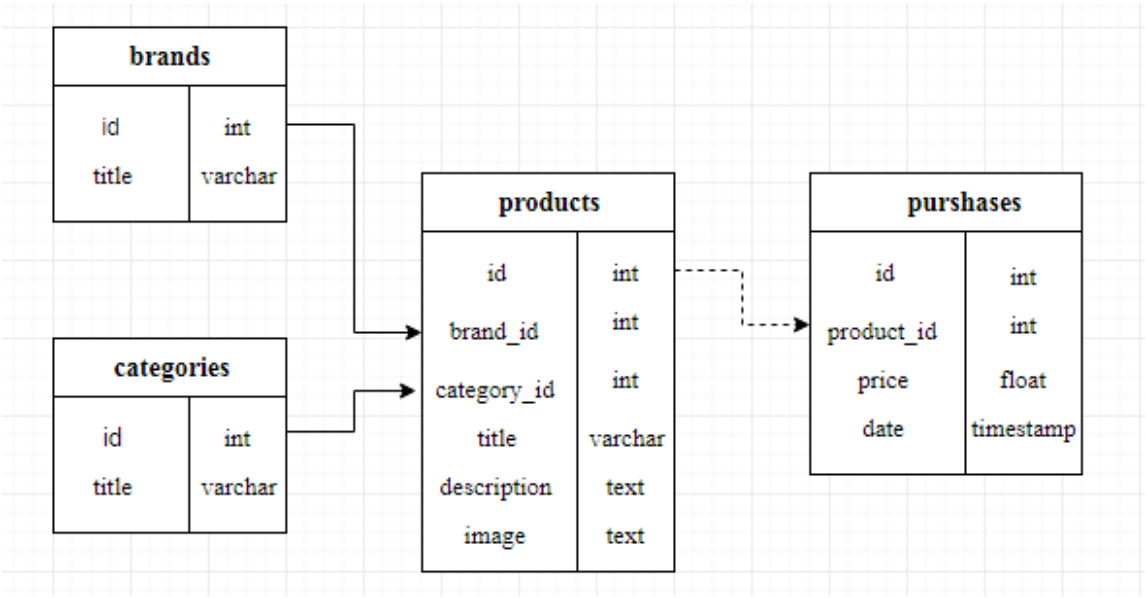

Рисунок 3.22 – концептуальна схема бази даних «diplom_comfy»

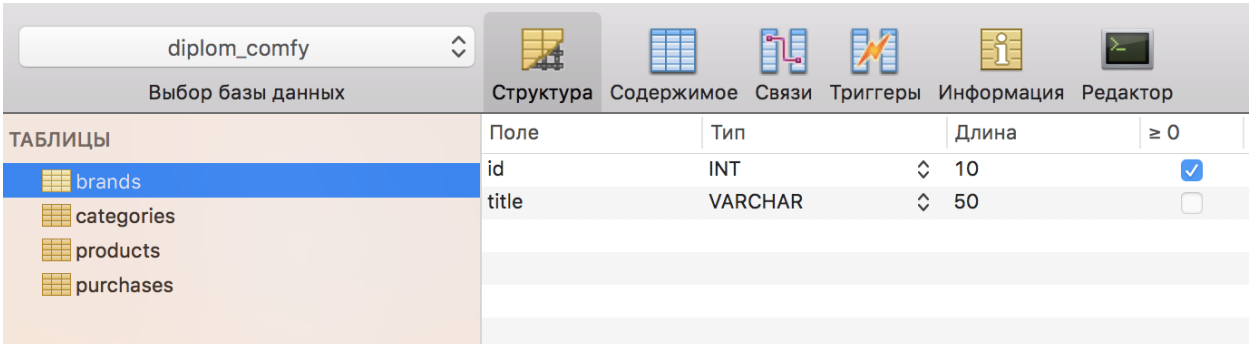

Рисунок 3.23 – Таблиця brands

Таблиця categories – категорії товарів (холодильники, телефони…). Таблиця наведена на рисунку 3.24 та має такі поля:

id – унікальний ідентифікатор,

title – назва категорії.

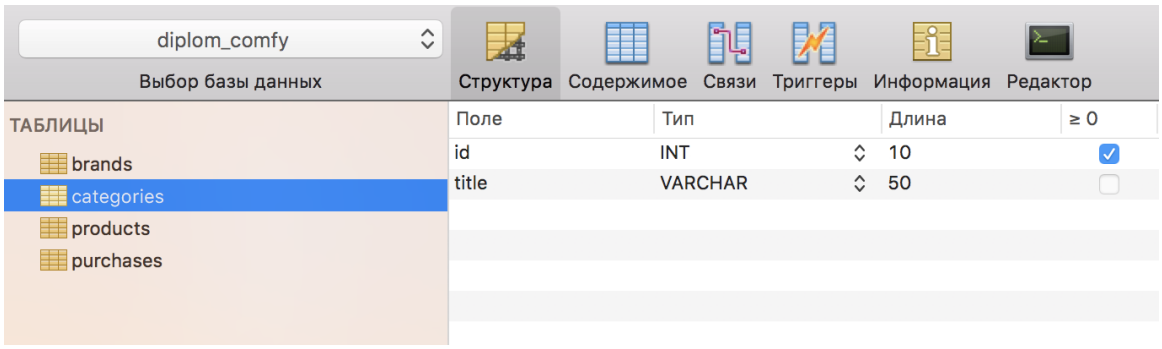

Рисунок 3.24 – Таблиця categories

Таблиця products – товари в магазині (iphone7, samsung, galaxy s9…). Таблиця наведена на рисунку 3.25 та має такі поля:

id – унікальний ідентифікатор,

brand_id – ідентифікатор бренда продукту (зв’язок з таблицею brands),

category_id – ідентифікатор категорії продукту (зв’язок з таблицею categories),

title – назва товару,

description – опис товару,

image – шлях до картинки.

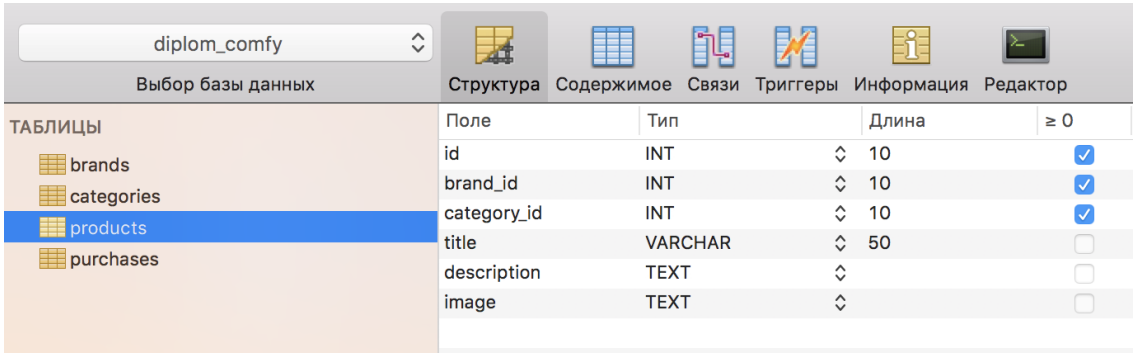

Рисунок 3.25 – Таблиця products

Таблиця purchases (англ. покупки) – журнал усіх покупок продуктів, що є в магазині. Таблиця наведена на рисунку 3.26 та має такі поля:

id – унікальний ідентифікатор,

product_id – ідентифікатор продукту (зв'язок з таблицею products),

price – ціна, за яку придбали товар,

date – дата покупки товару.

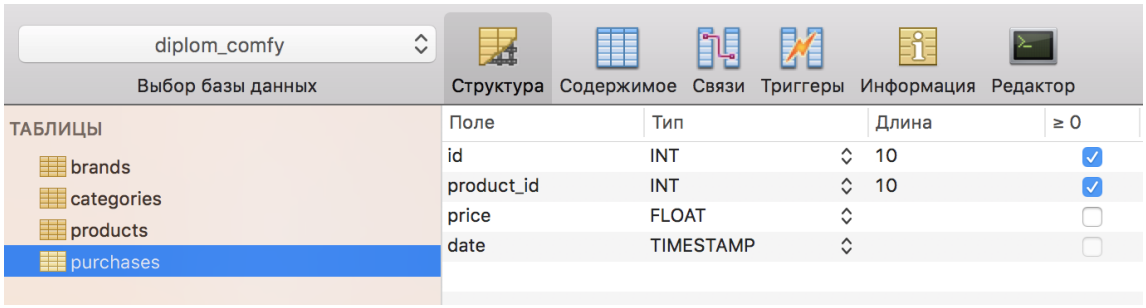

Рисунок 3.26 – Таблиця purchases

Для управління контентом веб-додатка та зберігання іншої інформації використовується база даних diplom. Підключення до бази даних відбувається у файлі handlers/db.php (рис. 3.27). Якщо у файлах проєкту потрібно використовувати базу даних, то ми одним рядком підключаємо даний файл у основний скрипт та використовуємо змінну $pdo для звернення до бази даних.

```php
<?php

$config = [
    'dbname'            => 'diplom',
    'host'              => '127.0.0.1',
    'login'             => 'root',
    'password'          => '',
    'dboption'          => array(
        \PDO::ATTR_ERRMODE => \PDO::ERRMODE_EXCEPTION,
        \PDO::ATTR_DEFAULT_FETCH_MODE => \PDO::FETCH_ASSOC,
        \PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES 'cp1251'"
    ),
];

$pdo = new PDO("mysql:host=" . $config['host'] . ";dbname=" . $config['dbname'] . ";charset=utf8", $config['login'], $config['password'], $config['dboption']);
$pdo->exec("set names utf8");
```

Рисунок 3.27 – Зміст файла db.php

Для того щоб звернутися до API магазину, потрібно підключити файл handlers/api.php, та викликати метод api_request($url, $method), передавши у нього актуальне посилання на API магазину та метод запиту (рис. 3.28).

```php
<?php

require_once 'handlers/db.php';
require_once 'handlers/api.php';

$stmt = $pdo->prepare('SELECT * FROM shops WHERE title = ?');
$stmt->execute([$_GET['shop']]);
$shop = $stmt->fetch();

$content = api_request($shop['path'] . '/product/' . $_GET['id'], 'GET', '', '');
```

Рисунок 3.28 – Приклад підключення файлів db.php і api.php
та використання їх можливостей

Для доступу до даних кожного магазину виконується запит зі спеціальними параметрами до API цього магазину. Запит відправляється, використовуючи надбудову PHP с URL (рис. 3.29).

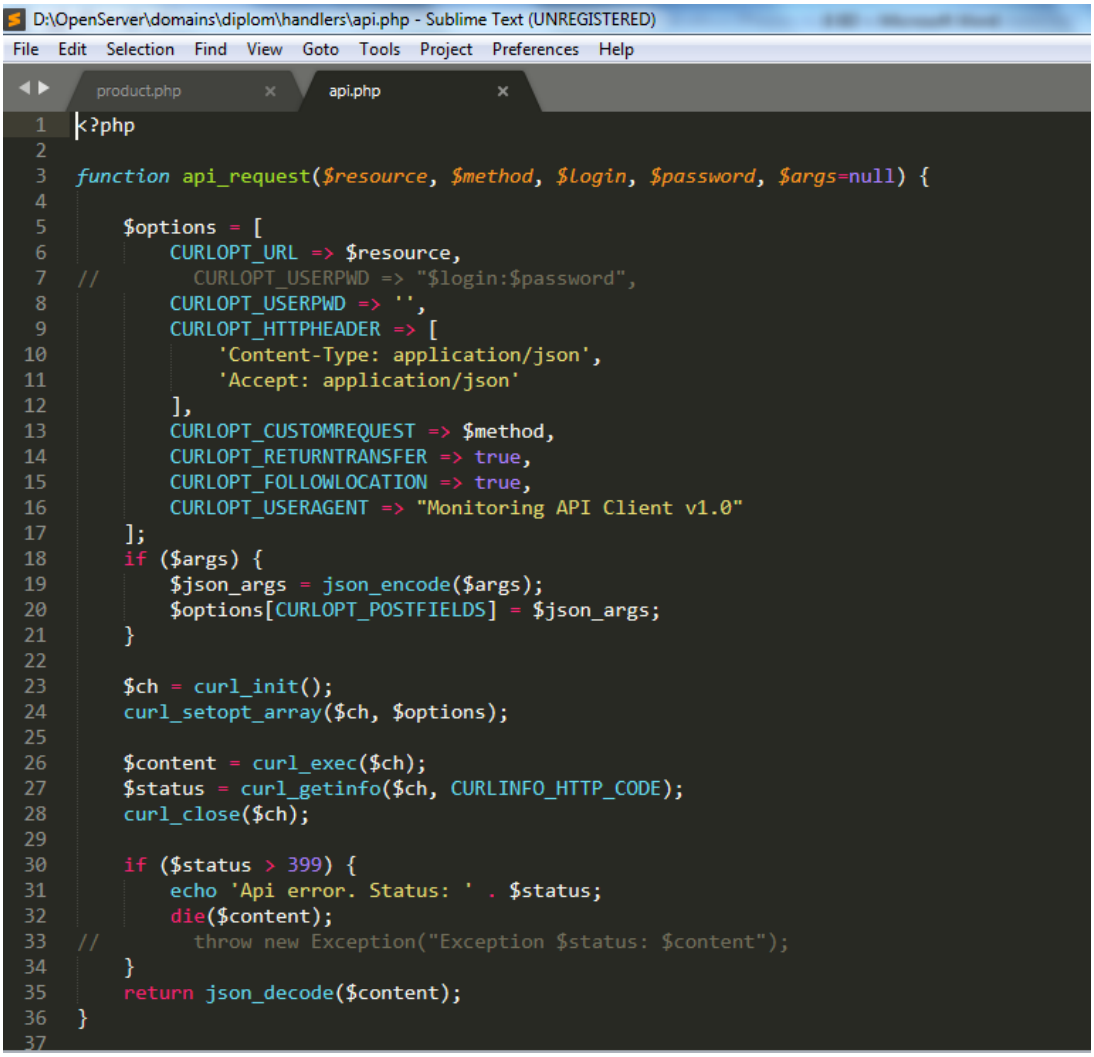

Рисунок 3.29 – Зміст файла api.php

3.3 Тестування веб-додатка

Під час тестування веб-додаток перевіряється на відповідність технічному завданню, перевіряються його технічні характеристики. Залежно від вихідного технічного завдання в процесі тестування можуть здійснюватися такі перевірки:

1) перегляд веб-додатка на різних моніторах. Перегляд сайта на моніторах різних розмірів і при різній роздільній здатності дозволяє оцінити, як буде виглядати дизайн сайту на комп'ютерах потенційних відвідувачів: чи з'явиться горизонтальна смуга прокручування на маленьких екранах, чи не зміщуються елементи при різному роздільній здатності тощо (рис. 3.30);

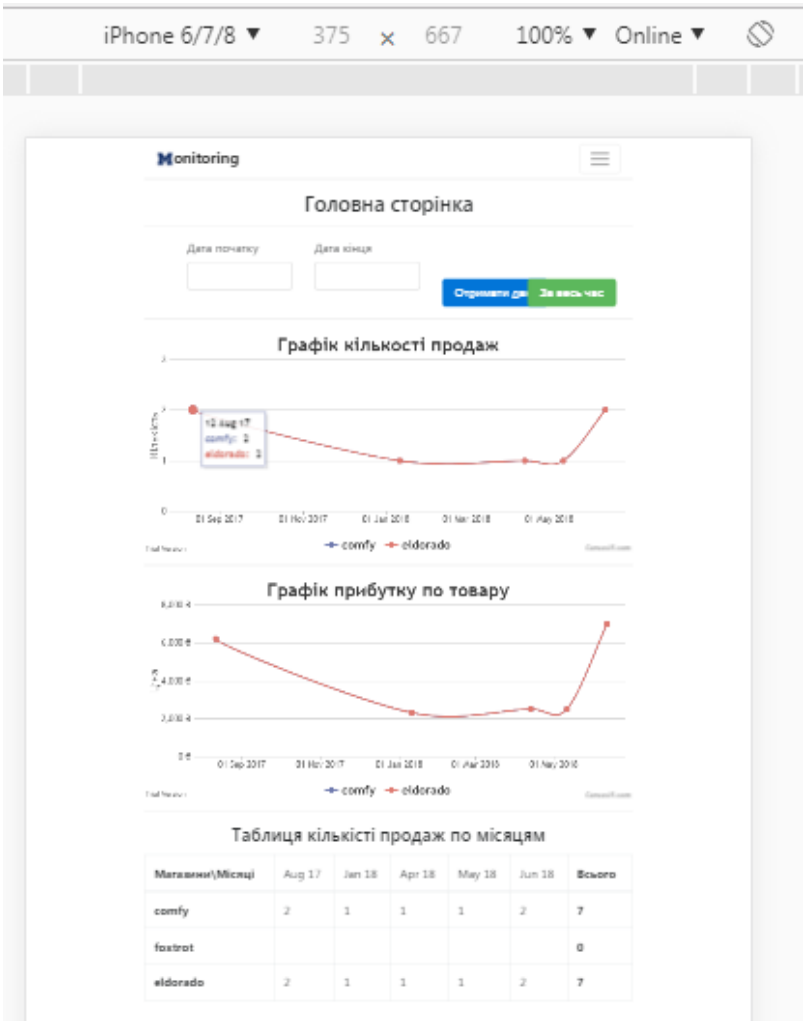

Рисунок 3.30 – Перегляд веб-додатка на різних моніторах

2) перегляд сайта в різних інтернет-браузерах і їхніх версіях дозволяє перевірити кросбраузерність веб-додатку. Було перевірено у браузерах: Opera, Chrome, Yandex, Firefox, Safari (рис. 3.31–3.32);

3) також було перевірено стабільну і безпомилкову роботу сайту на різних операційних системах, таких як Windows, MacOs, Linux Ubuntu 16.04. На рисунку 3.33 наведено перегляд сайта у браузері Chrome.

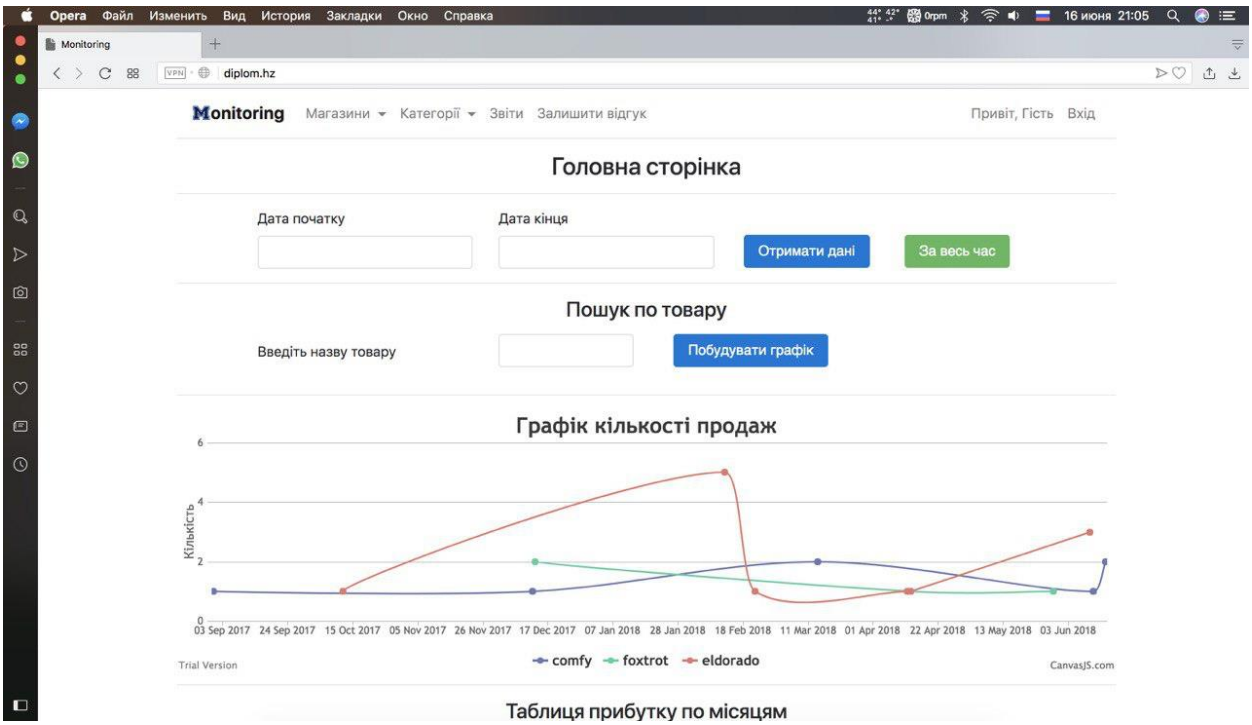

Рисунок 3.31 – Перегляд сайта в інтернет-браузері Opera

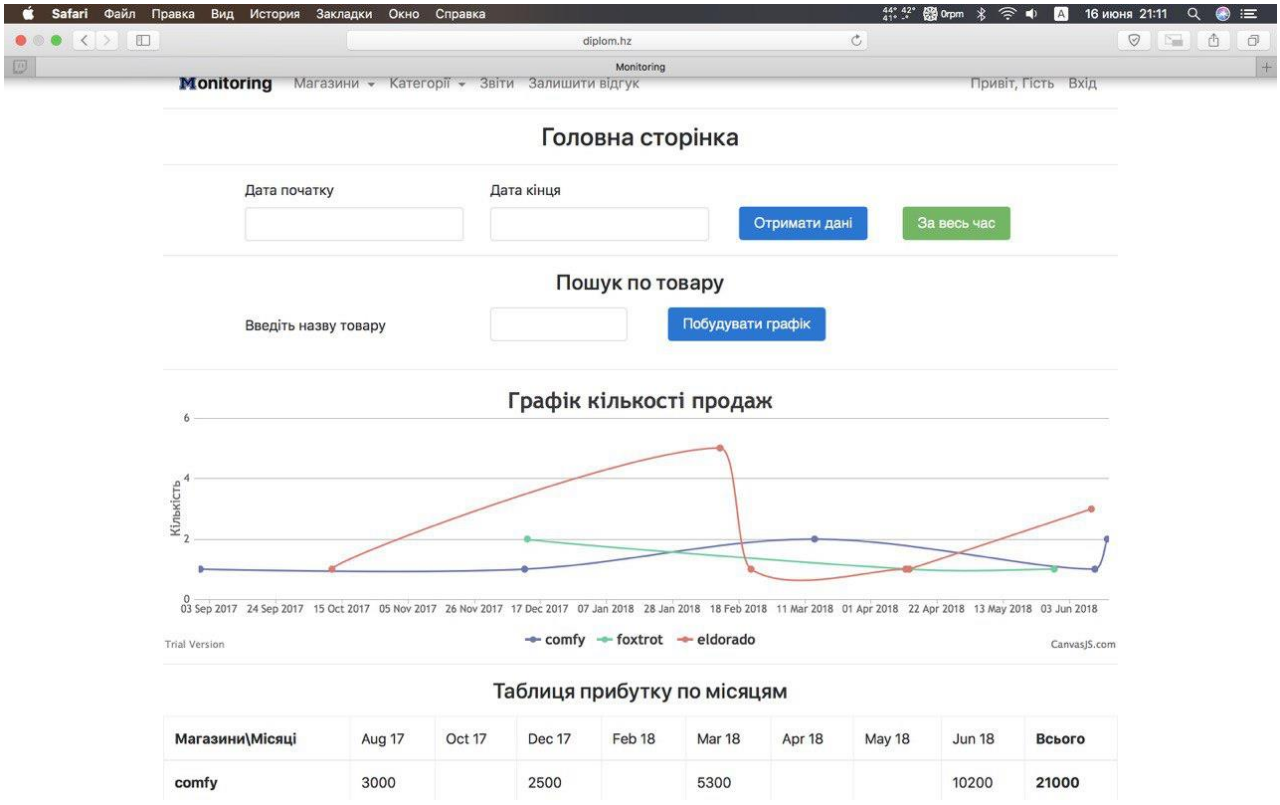

Рисунок 3.32 – Перегляд сайта в інтернет-браузері Safari

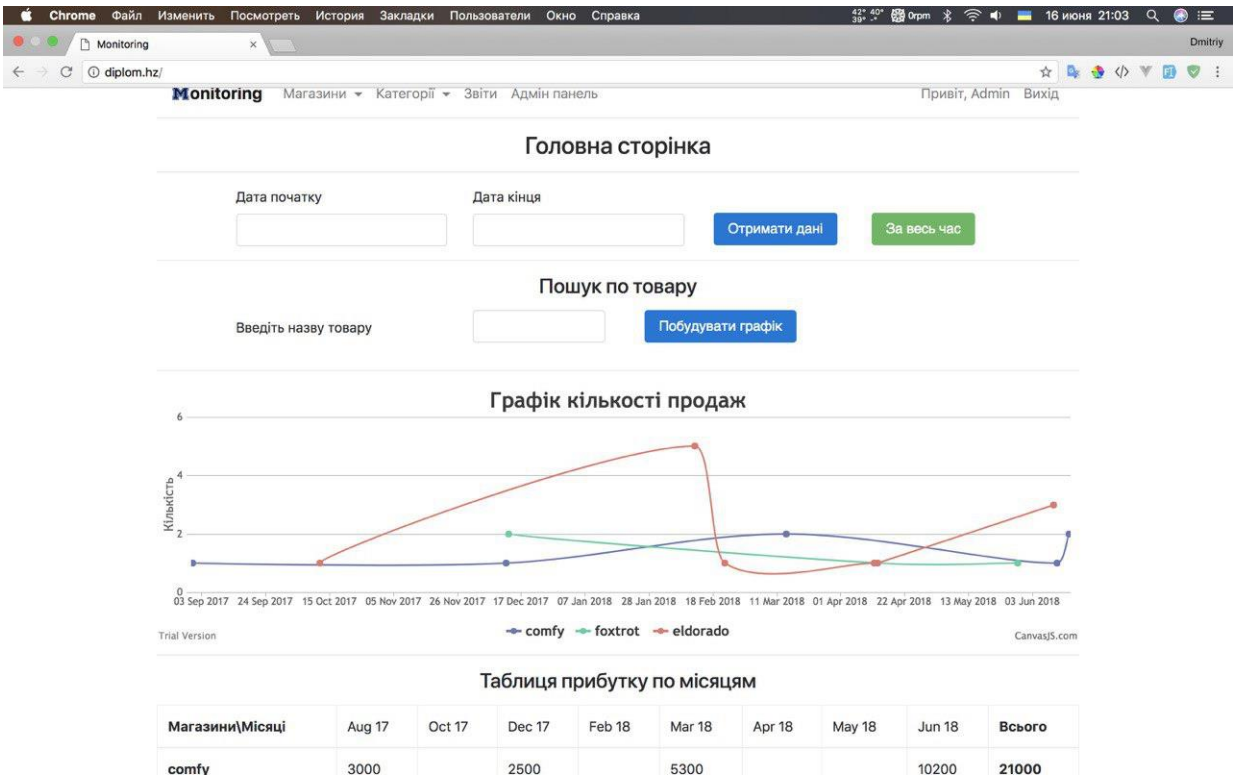

Рисунок 3.33 – Перегляд сайта в інтернет-браузері Chrome

3.4 Інструкція з використання веб-додатка

Інструкція адміністратора

Щоб перейти на панель адміністрування потрібно перейти на панель авторизації за посиланням «вхід» на «шапці» веб-додатка та виконати вхід до системи (рис. 3.1).

Панель адміністрування містить такі пункти меню, як назва веб-додатка налаштування магазинів, категорій та перегляд відгуків, привітання та посилання на сторінку для авторизації.

На головній сторінці адміністративної панелі є можливість обирати, які категорії будуть відображатися для користувачів (рис. 3.5).

При переході на сторінку «Категорії» відображаються всі категорії кожного магазину та можливість додати їх до обраних категорій. У обраних категоріях знаходяться ті категорії, які будуть відображатися у меню користувача (рис. 3.8).

При переході на сторінку «Відгуки» відображаються відгуки від користувачів та інформація про них (рис. 3.9).

Інструкція користувача

Стартова сторінка веб-додатка – це перша сторінка, яку бачить користувач перейшовши за посиланням http://diplom.hz/.

На даній сторінці користувач може перейти по таким пунктам меню як: «Магазини», «Категорії», «Звіти» та «Залишити відгук». Також для відвідувача веб-додатка відображені ті графіки, які обрав адміністратор для перегляду (рис. 3.4).

При переході на сторінку «Магазини» відображаються товари та інформація про них (рис. 3.10).

Щоб переглянути інформацію про кількість продажу та ціни конкретного товару в конкретному магазині треба натиснути на кнопку із іконкою корзини (рис. 3.11).

Щоб переглянути інформацію про кількість продажу та ціни конкретного товару у всіх магазинах треба натиснути на кнопку із зображенням кубку (рис. 3.12).

При переході на сторінку «Категорії» відображаються всі товари з магазинів, в яких є ця категорія (рис. 3.13).

При переході на сторінку «Звіти» користувач має можливість завантажити різні звіти собі на комп’ютер у форматі .csv (рис. 3.14).

При переході на сторінку «Залишити відгук» користувач має можливість залишити свій відгук (рис. 3.15).

ВИСНОВКИ

Досвідчені власники сайта або керівники, які за нього відповідають, знають про необхідність постійного моніторингу доступності сайта – відстеження того, як він працює. Моніторинг сервера потрібний для контролю стану операційної системи, СУБД, веб-сервера, інших компонентів програми. Моніторинг-додатки контролюють стан бізнес-показників самого додатка.

Задача моніторингу інфраструктури тісно пов’язана з моніторингом параметрів бізнес-операцій та вимогами бізнесу. Використання систем моніторингу дозволить чітко контролювати всі етапи бізнес-процесу, виявити та попередити відхилення від них, що значно підвищить ефективність управління бізнесом.

У результаті виконання випускної роботи було виконано:

1) дослідження такої предметної галузі як моніторинг обсягів купівлі-продажу для мережі магазинів;

2) обґрунтований вибір середовища розробки;

3) створена концептуальна модель бази даних;

4) створений веб-додаток моніторингу обсягів купівлі-продажу для мережі магазинів з простим і зрозумілим у використанні інтерфейсом, що дозволяє користувачу швидко розібратися в роботі додатка та легко знайти потрібну інформацію.

Даний комплекс основним призначенням має аналітичний аналіз різних категорій товарів в інших магазинах і їх порівняння з продукцією власної мережі магазинів.

Веб-сервіс реалізований в повному обсязі та відповідно до усіх заданих вимог і норм. Усі поставлені завдання виконані автором.

Основні результати роботи оприлюднені та обговорені на міжнародній науково-технічній конференції студентів та молодих вчених «Інформатика, математика, автоматика» (ІМА – 2020) (Суми, 2020 р.) [13].

СПИСОК ЛІТЕРАТУРИ

1. Типи веб-сторінок – http://veb-page.blogspot.com/.
2. Класифікація веб-сторінок – http://www.shag.com.ua/klasifikaciya-veb-storinok.html.
3. Системи моніторингу веб-сайтів – http://elartu.tntu.edu.ua/bitstream/123456789/13439/2/VseukrStud_2012v1_Savula_V-Systemy_monitorynhu_veb_saitiv_84.pdf
4. Ярцев В. П. Організація баз даних та знань: навчальний посібник / В. П. Ярцев. – К.: ДУТ, 2018. – 214 с.
5. Навчальні матеріали з інформатики. Бази даних – https://www.ua5.org/database/.
6. Харів Н. О. Бази даних та інформаційні системи: навчальний посібник / Н. О. Харів. Рівне: НУВГП, 2018. – 127 с.
7. Формування моделі предметної області – https://stud.com.ua/59748/informatika/formuvannya_modeli_predmetnoyi_oblasti.
8. Шаталова О. К. Інтегрування інформаційних баз даних для оптимізації роботи підприємств та установ / О. К. Шаталова, А. В. Шостак, Т. В. Шабельник // Матеріали II Всеукраїнської науково-практичної інтернет-конференції студентів, аспірантів та молодих вчених за тематикою «Сучасні комп’ютерні системи та мережі в управлінні». 30 листопада 2019 р., м. Херсон, Україна. – Херсон, 2019. – С. 277–280. – http://kntu.net.ua/ukr/content/download/63883/377713/file/Матеріали конференції CICT 2019.pdf
9. Електронна організація баз даних – https://pidruchniki.com/74243/informatika/elektronna_organizatsiya_danih.
10. Шикула О. М. Дизайн Web-сторінок: HTML+CSS: навчальний посібник / О. М. Шикула. – К.: ІПДО, 2018. – 60 с.
11. Vital Things to Consider When Choosing a Database for Your App – https://yalantis.com/blog/how-to-choose-a-database/.

12. Collins M. J. Pro HTML5 with CSS, JavaScript, and Multimedia : Complete Website Development and Best Practices / M. J. Collins. – 1st ed. 2017. – Berkeley, CA : Apress, 2017. – XXXII, 560 p.
13. Шовкопляс Н. Р. Інформаційна система моніторингу обсягів купівлі продажу для мережі магазинів / Н. Р. Шовкопляс, О. А. Шовкопляс // Матеріали та програма міжнародної науково-технічної конференції студентів та молодих вчених «Інформатика, математика, автоматика» (ІМА – 2020), 20–24 квітня 2020 р., м. Суми, Україна. – Суми, 2020. – С. 59. – http://elitconference.sumdu.edu.ua/proceedings/

ДОДАТОК А

Код програмного додатка

Файли для відображення сторінок веб-додатка збирають усю інформацію про сторінку, зберігають у буфері та через метод ob_get_contents() видають користувачу.

Файли для відображення основних сторінок веб-додатка:

index.php – головна сторінка

```
<?php

require_once 'handlers/db.php';
require_once 'handlers/api.php';

$stmt = $pdo->prepare('SELECT * FROM shops where enabled = 1');
$stmt->execute();
$shops = $stmt->fetchAll();

$stmt = $pdo->prepare('SELECT * FROM categories where enabled = 1');
$stmt->execute();
$categories = $stmt->fetchAll();

$stmt = $pdo->prepare('SELECT * FROM main_page where enabled = 1');
$stmt->execute();
$parts = $stmt->fetchAll();

$api_params_str = '';
if (isset($_GET['dates']) && $_GET['dates'] === 'true') {
    $api_params_str .= '/?dates=' . $_GET['dates'] . '&from=' .
$_GET['from'] . '&to=' . $_GET['to'];
}

$top_products = false;
$price_categories = false;
$count_categories = false;
$count_purchases = [];
$price_purchases = [];
$count_month_purchases = [];
$price_month_purchases = [];
$count_names = false;

$price_names = false;

foreach ($parts as $part) {
    switch ($part['title']) {
        case 'top_products':
            $top_data = [];
            foreach ($shops as $shop) {
                $content = api_request($shop['path'] .
'/product/by/purchases' . $api_params_str, 'GET', '', '');
```

```
                foreach ($content->data as $item) {
                    $top_data[] = $item;
                }
            }

            if (count($top_data) > 0) {
                foreach ($top_data as $key => $value) {
                    $volume[$key] = $value->amount;
                }
                array_multisort($volume, SORT_DESC, $top_data);
                $top_products = array_slice($top_data, 0, 3);

                foreach ($top_products as $key => $value) {
                    $price = 0;
                    foreach ($value->product->purchases as $purchase)
{
                        $price += $purchase->price;
                    }
                    if ($price != 0) {
                        $value->avg_price = round($price /
count($value->product->purchases));
                    } else {
                        $value->avg_price = '-';
                    }
                }
            }
            break;
        case 'count_purchases':
            foreach ($shops as $shop) {
                $content = api_request($shop['path'] .
'/purchase/all/count' . $api_params_str, 'GET', '', '');
                $result = [
                    'name' => $shop['title'],
                    'type' => 'spline',
                    'showInLegend' => true,
                    'dataPoints' => $content->data,
                ];
                $count_purchases[] = $result;
            }
            break;
        case 'price_purchases':
            foreach ($shops as $shop) {
                $content = api_request($shop['path'] .
'/purchase/all/price' . $api_params_str, 'GET', '', '');
                $result = [
                    'name' => $shop['title'],
                    'type' => 'spline',
                    'showInLegend' => true,
                    'yValueFormatString' => '#,###.## ₴',
                    'dataPoints' => $content->data,
                ];
                $price_purchases[] = $result;
            }
            break;
        case 'count_categories':
            $count_categories = true;
            break;
```

```
        case 'price_categories':
            $price_categories = true;
            break;
        case 'count_month_purchases':
            $dates = [];
            foreach ($shops as $shop) {
                $content = api_request($shop['path'] .
'/purchase/all/count/month' . $api_params_str, 'GET', '', '');
                $sum = 0;
                foreach ($content->data as $item) {
                    $sum += $item->count;
                    $dates[] = $item->date;
                }
                $result = [
                    'name' => $shop['title'],
                    'data' => $content->data,
                    'sum' => $sum,
                ];
                $count_month_purchases['values'][] = $result;
            }
            $dates = array_unique($dates);
            array_multisort($dates);

            $dates_result = [];

            foreach ($dates as $date) {
                $time = new DateTime($date);
                $dates_result[] = [
                    'value' => $date,
                    'string' => date_format($time, 'M y'),
                ];
            }
            $count_month_purchases['dates'] = $dates_result;
            break;
        case 'price_month_purchases':
            $dates = [];
            foreach ($shops as $shop) {
                $content = api_request($shop['path'] .
'/purchase/all/price/month' . $api_params_str, 'GET', '', '');
                $sum = 0;
                foreach ($content->data as $item) {
                    $sum += $item->price;
                    $dates[] = $item->date;
                }
                $result = [
                    'name' => $shop['title'],
                    'data' => $content->data,
                    'sum' => $sum,
                ];
                $price_month_purchases['values'][] = $result;
            }
            $dates = array_unique($dates);
            array_multisort($dates);

            $dates_result = [];

            foreach ($dates as $date) {
```

```
                $time = new DateTime($date);
                $dates_result[] = [
                    'value' => $date,
                    'string' => date_format($time, 'M y'),
                ];
            }
            $price_month_purchases['dates'] = $dates_result;
            break;
        case 'count_names':
            $count_names = true;
            break;
        case 'price_names':
            $price_names = true;
            break;
    }
}
$data = [
    'menu_items' => include 'data/menu.php',
    'categories' => $categories,
    'shops' => $shops,
    'parts' => [
        'top_products' => $top_products,
        'count_purchases' => $count_purchases,
        'price_purchases' => $price_purchases,
        'count_categories' => $count_categories,
        'price_categories' => $price_categories,
        'count_month_purchases' => $count_month_purchases,
        'price_month_purchases' => $price_month_purchases,
        'count_names' => $count_names,
        'price_names' => $price_names,
    ],
    'time_from' => isset($_GET['from']) ? $_GET['from'] : '',
    'time_to' => isset($_GET['to']) ? $_GET['to'] : '',
];

ob_start();
extract($data);

require 'templates/header.php';
require 'templates/main.php';
require 'templates/footer.php';

ob_get_contents();
```

login.php

```
<?php

ob_start();

$data = [
    'menu_items' => include 'data/menu.php',
];

extract($data);

require 'templates/header.php';
```

```
require 'templates/login.php';
require 'templates/footer.php';

ob_get_contents();
```

logout.php

```
<?php

session_start();

unset($_SESSION['token']);

header("Location: http://".$_SERVER['SERVER_NAME']);
```

shop.php

```
<?php

require_once 'handlers/db.php';
require_once 'handlers/api.php';
$stmt = $pdo->prepare('SELECT * FROM shops WHERE id = ?');
$stmt->execute([$_GET['id']]);
$shop = $stmt->fetch();

$content = api_request($shop['path'] . '/all/products', 'GET', '',
'');

foreach($content->data as $key => $value) {
//    $value->avg_price = end($value->purchases)->price;
    $price = 0;
    foreach($value->purchases as $purchase) {
        $price += $purchase->price;
    }
    if ($price != 0) {
        $value->avg_price = $price / count($value->purchases);
    } else {
        $value->avg_price = '-';
    }
}
$data = [
    'menu_items' => include 'data/menu.php',
    'shop' => $shop,
    'content' => array_chunk($content->data, 3),
];

ob_start();

extract($data);

require 'templates/header.php';
require 'templates/shop.php';
require 'templates/footer.php';

ob_get_contents();
```

category.php

<?php

```
require_once 'handlers/db.php';
require_once 'handlers/api.php';

$stmt = $pdo->prepare('SELECT * from categories where id = ?');
$stmt->execute([$_GET['id']]);
$category = $stmt->fetch();

$stmt = $pdo->prepare('SELECT * from shops where enabled = 1');
$stmt->execute();
$shops = $stmt->fetchAll();
$filtered_shops = [];

foreach ($shops as $key => $value) {
    $content = api_request($value['path'] .
'/product/by/category?title=' . $category['title'], 'GET', '', '');

    if ($content->data) {

        foreach ($content->data as $key2 => $value2) {
            $price = 0;
            foreach($value2->purchases as $purchase) {
                $price += $purchase->price;
            }
            if ($price != 0) {
                $value2->avg_price = $price / count($value2-
>purchases);
            } else {
                $value2->avg_price = '-';
            }
        }

        $shops[$key]['products'] = array_chunk($content->data, 3);
        $filtered_shops[$key] = $shops[$key];
    }
}

$data = [
    'menu_items' => include 'data/menu.php',
    'shops' => $filtered_shops,
    'category' => $category,
];

ob_start();

extract($data);

require 'templates/header.php';
require 'templates/category.php';
require 'templates/footer.php';

ob_get_contents();
```

feedback.php

```
     <?php

require_once 'handlers/db.php';
require_once 'handlers/api.php';

$data = [
    'menu_items' => include 'data/menu.php',
];

ob_start();

extract($data);

require 'templates/header.php';
require 'templates/feedback.php';
require 'templates/footer.php';

ob_get_contents();
```

feedback-success.php

```
     <?php

ob_start();

require 'templates/feedback-success.php';
require 'templates/footer.php';

ob_get_contents();
```

product.php

```
     <?php

require_once 'handlers/db.php';
require_once 'handlers/api.php';

$stmt = $pdo->prepare('SELECT * FROM shops WHERE title = ?');
$stmt->execute([$_GET['shop']]);
$shop = $stmt->fetch();

$content = api_request($shop['path'] . '/product/' . $_GET['id'],
'GET', '', '');

$product = $content->data[0];
$price = 0;
$amounts = [];

foreach($product->purchases as $purchase) {
    $price += $purchase->price;
    $date = new DateTime($purchase->date);
    $format_date = $date->format('Y-m-d');
    if (array_key_exists($format_date, $amounts)) {
        $amounts[$format_date] ++;
    } else {
        $amounts[$format_date] = 1;
    }
}
```

```
$new_amounts = [];
foreach ($amounts as $key => $value){
    array_push($new_amounts, [
        'date' => $key,
        'count' => $value,
    ]);
}
if ($price != 0) {
    $product->avg_price = round($price / count($product->purchases));
    $product->act_price = end($product->purchases)->price;
} else {
    $product->avg_price = '-';
    $product->act_price = '-';
}
$data = [
    'menu_items' => include 'data/menu.php',
    'product' => $content->data[0],
    'shop_title' => ucfirst($shop['title']),
    'amounts' => $new_amounts,
];
ob_start();

extract($data);

require 'templates/header.php';
require 'templates/product.php';
require 'templates/footer.php';

ob_get_contents();
```

product-all.php

```
<?php
require_once 'handlers/db.php';
require_once 'handlers/api.php';

$stmt = $pdo->prepare('SELECT * FROM shops where enabled = 1');
$stmt->execute();
$shops = $stmt->fetchAll();

$chart_data = [];
$carousel_data = [];
foreach ($shops as $shop) {
    $content = api_request($shop['path'] . '/product/by/name/' .
$_GET['name'], 'GET', '', '');

    if (!isset($content->data->purchases) || !isset($content->data-
>product)) continue;

    $chart_data[] = [
        'name' => ucfirst($shop['title']),
        'type' => 'spline',
        'showInLegend' => true,
        'dataPoints' => $content->data->purchases,
    ];
    $sum_price = 0;
    $sum_count = 0;
```

```
    $avg_price = '-';
    foreach ($content->data->purchases as $item) {
        $sum_price += $item->price;
        $sum_count += $item->cnt;
    }

    if ($sum_count > 0) {
        $avg_price = round($sum_price / $sum_count);
    }
    $carousel_data[] = [
        'shop' => ucfirst($shop['title']),
        'id' => ucfirst($content->data->product->id),
        'title' => ucfirst($content->data->product->title),
        'description' => $content->data->product->description,
        'image' => $content->data->product->image,
        'brand' => ucfirst($content->data->product->brand->title),
        'category' => ucfirst($content->data->product->category-
>title),
        'count_all' => $sum_count,
        'avg_price' => $avg_price,
    ];
}
$data = [
    'menu_items' => include 'data/menu.php',
    'product_name' => ucfirst($_GET['name']),
    'carousel_data' => $carousel_data,
    'chart_data' => $chart_data,
];

ob_start();

extract($data);

require 'templates/header.php';
require 'templates/product-all.php';
require 'templates/footer.php';

ob_get_contents();
```

report.php

```
<?php
$data = [
    'menu_items' => include 'data/menu.php',
];

ob_start();

extract($data);

require 'templates/header.php';
require 'templates/report.php';
require 'templates/footer.php';

ob_get_contents();
```