МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний економічний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерної інженерії

КАПУСТИНСЬКИЙ Роман Ігорович

Сайт компанії "Євротерсервіс" на основі системи керування вмістом / The site of the company "Euroservice" based on the content management system

спеціальність: 6.050102 - Комп'ютерна інженерія
освітньо-професійна програма - Комп'ютерні системи та мережі

Випускна кваліфікаційна робота

Виконав студент групи КСМ-42/1
Капустинський Р.І.
_____

Науковий керівник:
Ігнатєв І.В.
_____

Тернопіль - 2019

РЕЗЮМЕ

Бакалаврська робота містить 78 сторінок пояснюючої записки, 25 рисунків, 8 таблиць, 3 додатки. Обсяг графічного матеріалу 2 аркуші формату А3.

Метою дипломного проекту є розробка сайту компанії «Євротерсервіс» на основі системи керування вмістом.

Даний дипломний проект присвячений розробки вебсторінки. В роботі проведений аналіз існуючих технологій розробки веб-сайтів, розроблено загальну структуру сторінки, розроблено базу даних, розроблено дизайн сайту.

Реалізовано сайт компанії «Євротерсервіс» на основі системи керування вмістом.

Ключові слова: САЙТ КОМПАНІЇ НА ОСНОВІ СИСТЕМИ КЕРУВАННЯ ВМІСТОМ, ВЕБ-РОЗРОБКА, ВЕБ-СТОРІНКА.

RESUME

The thesis project contains 78 pages of explanatory note, 25 figures, 8 tables, 3 annexes. The volume of graphic material is 2 sheets of A3 format.

The purpose of the diploma project is to develop the site of the company "Euroservice" on the basis of the content management system.

This diploma project is devoted to the development of a web page. In the work the analysis of existing technologies of web-site development was conducted, the overall structure of the page was developed, the database was developed, the website design was developed.

The site of the company "Euroservice" is implemented on the basis of content management system.

Keywords: SITE OF COMPANY BASED ON CONTENT SYSTEM CONTENT, WEB DEVELOPMENT, WEB-PAGE.

ЗМІСТ

ВСТУП

На сьогоднішній день, веб-сторінка являється найзручнішим та найефективнішим способом розповсюдження інформації. Саме тому, на мою думку, у будь-якого підприємства повинна бути своя веб-сторінка. У теперішній час, майбутньому клієнту зручніше та простіше зайти на сайт компанії, отримати інформацію про послуги які пропонують та у кінці залишити заявку на зворотній дзвінок. Якщо ж людина здійснює запит у пошуковій системі(наприклад Google), то підприємство взагалі знаходиться поза конкуренцією, якщо у нього немає веб-сторінки. Саме тому, на мою думку, сайт являється необхідністю, а не розкішшю.

Метою даної бакалаврської роботи є розробка веб-сторінки для компанії «Євротерсервіс».

Для цього необхідно розв’язати наступні задачі:

- проаналізувати технології розробки веб-сторінок;
- розробити структуру веб-сторінки;
- розробити базу даних;
- розробити дизайн сайту;
- створити веб-сторінку за допомогою програмного забезпечення.

1 ТЕХНОЛОГІЇ РОЗРОБКИ І ФУНКЦІОНУВАННЯ WEB-ТЕХНОЛОГІЙ

1.1 Програмні засоби створення web-сторінок

Веб-сторінка – це документ, до якого можна отримати доступ через інтернет або інші мережі використовуючи браузер. Це відбувається за допомогою введення посилання у полі пошуку. Вона може містити текст, зображення, посилання на інші сторінки та файли. Створення даних документів відбувається за допомогою спеціальних засобів, які називають мовою розмітки. Вона являє собою комбінацію слів та символів, які дають вказівки про те, як документ повинен відображатися. Найбільш широко відомою мовою розмітки сьогодні є мова гіпертекстової розмітки (HTML). Ця мова використовується браузерами для відображення веб-сайтів [1].

1.1.1 Мова розмітки гіпертексту

Що таке гіпертекст і що являє собою мова розмітки гіпертексту (абревіатура HTML)? Гіпертекст – це метод, за допомогою якого ви рухаєтесь по мережі інтернет клікаючи на спеціальний текст, який називається гіперпосиланням, який веде до іншої веб-сторінки. Мова розмітки гіпертексту – це засіб, який за допомогою спеціальних елементів коду, які називають тегами, повідомляє браузеру як відображати ті чи інші елементи, наприклад текст, зображення та інші види мультимедіа [2].

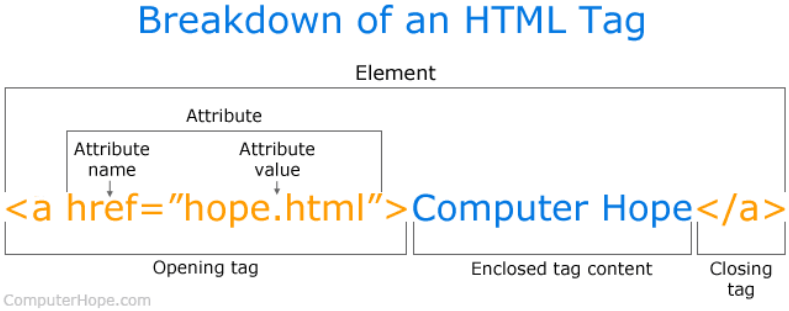

Рисунок 1.1 – Приклад html тегу

Як ми бачимо з рисунку 1.1, кожен тег відображається як літери або слова між знаками <(менше ніж) та >(більше ніж), наприклад <body>. Щоб завершити дію, яку він передбачує, необхідно його закрити, додавши знак “/” перед його іменем, наприклад </body>. Більшість тегів записуються такими парами, але це не являється абсолютним правилом, тому деякі із них не потребують закриття. Будь-яка веб-сторінка вміщує у собі наступні теги:

– <html ></html> – він дає браузеру зрозуміти, що це html документ;

– <head></head> – цей тег вміщує у собі метадані, такі як назва сторінки, назва кодування, яке використовується та багато іншого;

– <body></body> – охоплює весь зміст, який з’являється на сторінці.

З перших днів, HTML пережила неймовірну еволюцію. Всесвітній консорціум (абревіатура W3C) постійно публікує нові версії та оновлення. HTML версії 4, яку у ці дні зазвичай називають просто HTML, було опубліковано у 1999 році, та у 2014 році з’явилася нова версія названа п’ятою. HTML5 привнесла багато нових можливостей. Однією з найбільш очікуваних є вбудована підтримка аудіо та відео. Замість використання Flash-плеєра, ми можемо просто вставляти відео та аудіо файли на наші веб-сторінки за допомогою нових тегів <audio></audio> та <video></video> [3]. Ще однією особливістю є вбудована підтримка масштабованої векторної графіки (SVG) і MathML для математичних і наукових формул. HTML5 також представила кілька семантичних удосконалень. Нові спецальні теги інформують браузер про зміст контенту, що приносить користь як читачам, так і пошуковим системам. Найпопулярнішими такими тегами є: article, section, aside, header та footer.

Article(від англ. Article – стаття) – являє собою незалежний фрагмент веб-сторінки і, як правило, включає «шапку», основний зміст і «підвал», в яких розміщуються такі частини, як заголовок, ім’я автора, дата публікації і тд. Цей тег зазвичай застосовується для статей сайту, повідомлення блогу та форуму, коментарів.

Section(від англ. Section – розділ) – задає розділ документу, може використовуватися для блоку новин, контактної інформації, гловного тексту,

вкладок у діалоговому вікні та ін.

Aside(від англ. Aside – у стороні, відступ) – частина сторінки, яка має непряме відношення до вмісту документу і може бути розглянута окремо від нього. Зазвичай застосовується для бічних панелей, рекламних блоків, посилань на архів, міток та іншої інформації, яка відокремлена від основного вмісту сторінки.

Header(від англ. Header – верхній колонтитул, шапка) – задає шапку сайту або розділу веб-сторінки. Усередині зазвичай розміщується логотип, назва сайту, пошукова форма, навігація по сторінці та інше.

Footer(від. Англ. Footer – нижній колонтитул, підвал) – визначає «підвал» сайту або розділу веб-сторінки. У ньому може розміщуватися ім’я автору, дата документу, контактна і правова інформація [4].

Підводячи підсумок, можна сказати, що HTML являється головною мовою розмітки в інтернеті. Вона працює в кожному браузері і підтримується Всесвітнім Консорціумом (W3C). Незважаючи на те, що HTML є потужним інструментом, її недостатньо для створення професійного та повністю реагуючого веб-сайту. ЇЇ можна використовувати для додавання елементів та створення структури змісту сторінки. Для надання ексклюзивного вигляду веб-сторінці використовуються каскадні таблиці стилів(CSS), а для забезпечення інтерактивності – мова програмування JavaScript.

1.1.2 Каскадні таблиці стилів

Кожен тег мови HTML може бути модифікований завдяки атрибуту «style». З його допомогою можна змінювати колір тексту, стиль шрифтів, колір фону і тд. Але, використовуючи даний підхід, загальна структура html коду робиться більш складною для розуміння, через велику кількість надмірного коду. Тому, хорошою практикою являється опис зовнішнього вигляду елементів у окремому файлі з розширенням «*.css». Каскадні таблиці стилів (абревіатура CSS) є простою мовою дизайну, призначеною для спрощення

процесу створення веб-сторінок. CSS обробляє зовнішній вигляд веб-сторінки. Завдяки її використанню можна керувати кольором тексту, стилем шрифтів, інтервалом між абзацами, розміром та розміщенням стовпців, які фонові зображення або кольори використовуються, макетом, варіантами відображення для різних пристроїв і розмірами екрану, а також безліччю інших ефектів. У CSS можна виділити п’ять переваг.

1. CSS економить час – ви можете написати CSS файл один раз, а потім повторно використовувати його на декількох сторінках HTML. Ви можете визначити стиль для кожного елемента HTML і застосувати його до якомога більшої кількості веб-сторінок.

2. Сторінки завантажуються швидше. Якщо ви використовуєте CSS, вам не потрібно кожного разу писати атрибути тегів HTML. Просто напишіть одне правило CSS тегу і застосуйте його до всіх входжень цього тегу. Менше коду – швидше завантаження сторінки.

3. Простота в обслуговуванні. Щоб зробити глобальні зміни, просто внесіть зміни до файлу, і всі елементи на всіх веб-сторінках будуть автоматично оновлюватися.

4. Кращі стилі для HTML. CSS має набагато ширший масив атрибутів, ніж HTML, так що ви можете надати вашій сторінці набагато кращий вигляд, ніж у порівнянні з атрибутами HTML.

5. Сумісність з кількома пристроями. Таблиці стилів дозволяють оптимізувати вміст для більш ніж одного типу пристроїв. Використовуючи той самий документ HTML, можуть бути представлені різні версії веб-сайту для портативних пристроїв, таких як планшети та смартфони [5].

1.1.3 Мова програмування JavaScript

JavaScript – це мова сценаріїв або мова програмування, яка дозволяє реалізовувати складні речі на веб-сторінках, та перетворювати їх зі статичних у динамічні. Кожен раз, коли на сторінці щось відбувається, окрім простого

відображення статичного тексту або зображення, тобто своєчасне оновлення вмісту, інтерактивні карти, анімована 2D / 3D-графіка, прокрутка відео-музичних вікон і т.д, можна з впевненістю сказати, що за цим стоїть JavaScript. В основному вона використовується для розширення веб-сторінок та забезпечення більшого комфорту під час користування. Ще більш захоплюючою є функції, побудовані поверх ядра JavaScript. Так звані інтерфейси прикладного програмування (API) надають додаткові «суперсили» для використання в коді JavaScript. API – це готові блоки коду, які дозволяють розробнику реалізовувати програми, які інакше було б важко або неможливо реалізувати. Вони діляться на дві категорії – браузерні API та API третьої сторони [6].

Браузерні API вбудовані у веб-переглядач і можуть показувати дані з навколишнього комп'ютерного середовища або робити корисні складні речі. Прикладами браузерних API є: DOM, геолокація, Canvas і WebGL, аудіо та відео API.

DOM (Document Object Model) API дозволяє маніпулювати HTML і CSS, створювати, видаляти і змінювати HTML, динамічно застосовувати нові стилі до вашої сторінки, і т.д. Кожного разу, коли на сторінці з'являється спливаюче вікно або відображається новий вміст – це DOM у дії.

API геолокації отримує географічну інформацію. Так Google Maps може знайти ваше місце розташування та відобразити його на карті.

API Canvas і WebGL дозволяють створювати анімовану 2D і 3D графіку. Люди використовують їх для створення прекрасних анімацій на веб-сторінці та багатьох браузер них ігор, написаних на javascript.

Аудіо та відео API, такі як HTMLMediaElement і WebRTC, дозволяють робити дуже цікаві речі з мультимедіа, наприклад, відтворювати аудіо та відео прямо на веб-сторінці, або захоплювати відео з веб-камери і відображати їх на чужому комп'ютері.

API третьої сторони не вбудовані в браузер за замовчуванням і зазвичай їхній код та документацію знаходять на сторонніх сайтах у інтернеті.

Прикладом таких API є: API Твітеру, яке дозволяє вам відображати ваші останні твіти на своєму веб-сайті, та API карт гул і OpenStreetMap за допомогою яких можна вбудовувати карти у свою веб-сторінку, та створювати свої власні унікальні карти.

1.2 Алгоритм створення веб-сайту

Кожен веб-розробник або розробницька фірма мають свій особистий процес створення продукту, з різними етапами, фазами та компонентами. Але, існує загальна формула процесу створення продукту, яку зображену на рисунку нижче.

Рисунок 1.2 – Алгоритм створення веб-сайту

Існує три блоки: планування, розробка, удосконалення. Від початку і до кінця, кожен кусочок проведеної роботи, направленої на створення проекту, підпадає під один із цих етапів. Усередині них, може бути велика кількість кроків, але загалом алгоритм є таким: «Створити хороший продукт, який був досліджений і оцінений, та постійно удосконалювати його з плином часу». Далі, розберемо кожен етап детальніше.

1.2.1 Етап планування

Планування є найбільшим етапом процесу розробки з точки зору часу, зусиль і важливості. Воно передбачає велику кількість кроків, охоплює

величезну кількість завдань та багатьох людей різних професій. До планування входять:

– перша зустріч – початкова дискусія про бажання, витрати, терміни та роздуми про реалізацію;

– дослідження – проведення вправ, дій та утворень для розкриття діянь майбутнього користувача, його цілей. Вивчення уподобань вікової категорії для якої призначений сайт;

– технічне завдання – документ, у якому визначаються призначення, техніко-економічні та спеціальні вимоги до продукту, а також строки його створення. Укладається між двома сторонами – розробником та замовником;

– карта сайту – якщо сайт багатосторінковий, то створюється спеціальна карта, яка визначає усі можливі шляхи переходів між сторінками;

– SEO дослідження – аналіз ключових слів для пошукових систем, конкуренції та планування вмісту для контенту, що забезпечує рекламування;

– попередній дизайн – серія вправ, яка проводиться дизайнерами з клієнтами, для визначення їх уподобань та ідей щодо дизайну продукту.

1.2.2 Етап розробки

Етап розробки – фактичне виготовлення веб-сайту передбачає взяття всього, що було виявлено і сформульовано на етапі планування. Робота на даному етапі лягає на плечі дизайнерів користувача та користувальницького інтерфейсу, фронт-енд та бек-енд програмістів і архітекторів CMS. Наступні кроки є типовими для даного блоку:

– прототипування – створення каркасу сайту, на основі рисунків та ідей здобутих на етапі планування;

– дизайн інтерфейсу користувача – об’єднання шаблонів брендів (таких як колір, тип, зображення та ін.), що поєднується з прототипом, для більш виразнішого та живого вигляду сайту;

– розробка або налаштування CMS – система управління контентом має бути побудована та налаштована з урахуванням всіх вимог, які знадобляться клієнту для створення та управління контентом;

– бази даних – вибір, налаштування та створення структури бази даних;

– розробка сайту – створення веб-макету сайту та серверної частини за допомогою різних мов, таких як HTML, CSS, JavaScript, Ruby, PHP, Python та інших;

– перевірка якості забезпечення – тестування продукту на пошкоджені посилання, зламані сторінки, відсутність функціоналу тощо;

– тестування гнучкості – перевірка веб-сайту на різних пристроях та браузерах, для забезпечення ліпшого, оптимізованого досвіду для кожного користувача.

1.2.3 Етап удосконалення

Проект розробки веб-сайту не завершається під час його офіційного вводу у дію. Після цього, потрібно проводити його просування та аналіз тих факторів, які приводять користувачів на сайт та їх покращення. Цей етап називається удосконаленням та містить наступні кроки:

– технічне обслуговування сайту – виконання рутинних перевірок та оглядів для переконання, що вміст, функціональність та дизайнерські рішення працюють як потрібно;

– оптимізація сайту – удосконалення функціональності сайту та його оновлення і налаштування задля покращення загальних цілей сторінки;

– цифрова аналітика – огляд того, що користувачі роблять на сайті. За допомогою даних показників, можна визначити та запропонувати зміни та покращення;

– маркетинг – просування сайту за допомогою реклами, соціальних мереж, та розсилки листівок електронною поштою [7].

1.3 Взаємодія веб-серверів із середовищем інтернет

Поняття веб-сервер відноситься як і до апаратного так і програмного забезпечення. На апаратній стороні веб-сервер – це комп’ютер, який зберігає програмне забезпечення веб-сервера та файли компонентів веб-сайту (наприклад, документи HTML, зображення, таблиці стилів CSS і файли JavaScript). Він підключений до мережі Інтернет і підтримує обмін фізичними даними з іншими пристроями, які також знаходяться у мережі.

На стороні програмного забезпечення веб-сервер містить декілька частин, які контролюють, як користувачі Інтернету звертаються до розміщених файлів. HTTP-сервер – це програмне забезпечення, яке розуміє URL-адреси (веб-адреси) і HTTP (протокол, який використовує браузер для перегляду веб-сторінок). Він діє за трьома правилами.

1. Тільки клієнти можуть створити HTTP запит, і він повинен обов’язкового бути адресований серверу. Сервери можуть відповідати тільки на клієнтські HTTP запити.

2. Коли клієнт запитує файл через HTTP, він повинен надати посилання на нього.

3. Веб-сервер повинен надати відповідь на кожен HTTP запит, хоча б помилкою.

Програмне забезпечення веб-сервера відповідає за обробку та надсилання відповідей на вхідні запити. Він діє у два кроки.

1. Після отримання запиту HTTP-сервер спочатку перевіряє, чи запитувана URL-адреса відповідає існуючому файлу.

2. Якщо це так, веб-сервер передає вміст файлу браузеру. В іншому випадку, веб-сервер повертає повідомлення про помилку у браузер. Найбільш поширеною є помилка 404, яка вказує на те, що відповідного файлу не було знайдено на сервері [8].

1.4 Постановка задачі

На сьогоднішній день, людство має можливість виходу в Інтернет майже з любої точки нашої планети. З появою смартфонів, використання мережі стало ще швидшим та зручнішим. Стало можливим, буквально за пару секунд отримати доступ до будь-якого веб-ресурсу. Саме тому, на мою думку, у будь-якого підприємства повинна бути своя веб-сторінка. Вона являється найзручнішим та найефективнішим способом розповсюдження інформації.

У теперішній час, майбутньому клієнту буде зручніше та простіше зайти на сайт компанії, отримати інформацію про послуги які пропонують та у кінці залишити заявку на зворотній дзвінок. Якщо ж людина здійснює запит у пошуковій системі(наприклад Google), то підприємство взагалі знаходиться поза конкуренцією, якщо у нього немає веб-сторінки. Саме тому, на сьогоднішній день, сайт являється необхідністю, а не розкішшю.

Метою даної бакалаврської роботи є розробка веб-сторінки для компанії «Євротерсервіс». Запланованим є створення так званого «landing-page» - одно сторінкового веб-сайту, який має на меті зацікавити майбутнього клієнта, проінформувати його щодо послуг які надає підприємство та надати йому можливість зворотнього зв’зку із компанією. Мною було виділено п’ять пунктів, реалізація яких дозволить досягти найкращого результату.

1. Створення унікального, зрозумілого та приємного дизайну сторінки. Саме дизайн, підбір правильної колірної схеми, створення зручного у користуванні та спогляданні користувацького інтерфейсу являється запорукою утримання клієнта. Тому, саме розробці зовнішнього вигляду буде приділена велика увага. Головним інструментом для створення макету сторінки являється програмний продукт під назвою «Figma», фукнціонал та безоплатна модель розповсюдження якого являється чудовим вибором для досягнення поставлених цілей.
2. Адаптивність розробленого дизайну, що означає досягнення найбільш

привабливого вигляду сторінки на пристроях із різною роздільною здатністю. У наш час, близько 80% користувачів відвідують сайти з мобільних платформ. Саме тому, створення найкращих умов для усіх категорій користувачів, являється пріоритетною задачею для виконання.

3. Наявність форми зворотного зв’язку із компанією. Це являється ключовим компонентом для утримання клієнта, для якого простіше залишити свої контактні дані та зачекати дзвінка. Це являється хорошою практикою та призводить до збільшення кількості потенційних клієнтів.

4. Наявність системи управління контентом, яка надасть компанії змогу редагувати сторінку у разі необхідності без допомоги веб-розробників.

5. Оптимізація швидкості загрузки сайту за допомогою інструментів Google. Ідеальний час від початку загрузки сторінки до появи візуального контенту становить до 3-х секунд, на думку компанії Google. Високий час очікування часто призводить до передчасного виходу зі сторінки. Тому, оптимізація також являється важливим фактором при розробці сторінки.

2 ПРОГРАМНІ ЗАСОБИ РОЗРОБКИ ВЕБ-САЙТУ

2.1 Вибір системи управління контентом

На сьогоднішній день, більшість розроблюваних сайтів містять у своєму складі системи управління контентом. CMS (англ. «Content Management System») – це веб-додаток, що використовує базу даних (зазвичай MySQL) або інші методи для створення, редагування та зберігання вмісту HTML сторінки. Вміст створюється і редагується через браузер у спеціальному веб-додатку, та потім відображається глядачам на вашому сайті. Чому варто використовувати CMS? На це існує п’ять причин.

1. Можливість створення і редагування контенту на сайті без допомоги розробника. Зазвичай, ці системи зроблені для користувачів, які не мають досвіду у сфері програмування.
2. Наявність додаткових модулів та плагінів, як безкоштовних так і платних, які допоможуть розширити фукнціонал сторінки.
3. Можливість колаборації різних користувачів для роботи над одним і тим самим проектом. Все що необхідно – це створити для них облікові записи та надати посилання для входу у систему.
4. Додаткові можливості для оптимізації сайту у пошукових системах.
5. Наявність додаткового захисту від хакерів, що особливо важливо для сайтів електронної комерції.

Хитрість полягає в тому, щоб знайти CMS, що забезпечує функціональність, не жертвуючи простотою використання для себе або, залежно від ситуації, вашого клієнта. Хороша CMS дозволить вам витрачати більше часу на розробку інтерфейсу, а потім на реалізацію розширень або функціональних можливостей[9].

Існує чотири критерії для оптимального вибору CMS.

1. Простий, інтуїтивно зрозумілий інтерфейс для клієнта. Багато замовників не мають ніякого уявлення про html, css і подібні інструменти, тому

дружелюбний інтерфейс системи, з допомогою якої можна буде редагувати текст, зображення та інші елементи на сторінці є запорукою хорошого вибору.

2. Безкоштовний спосіб розповсюдження або наявність безкоштовної обмеженої у функціональності версії, для можливості тестування продукту без зайвих витрат.

3. Простота в інсталяції та налаштуванні.

4. Наявність захисту від несанкціонованого доступу.

У даний момент такі системи як WordPress, Drupal та Joomla займають більш ніж 85% на ринку CMS. Існують менші за популярністю проекти, які являються хорошою альтернативою, але ця трійка є «золотою серединою» та завдяки багатьом модулям і плагінам може змінювати свій функціонал як тільки завгодно. Розглянемо їх детальніше.

WordPress – це вибір більшості користувачів програмного забезпечення CMS. Її простота у використанні робить її ідеальним для людей, які не мають багато досвіду в галузі технологій для створення основних веб-сайтів. Як рішення з відкритим вихідним кодом та майже 50 000 плагінів, створених розробниками в усьому світі, вона має широкий потенціал налаштувань для задоволення більшості потреб бізнесу.

Drupal – це вибір підприємств, які хочуть створити більш надійний веб-досвід. Сайти, які вимагають організації складних даних, як, наприклад, ті, які потребують розміщення спільноти користувачів, часто звертаються до цієї CMS.

Joomla схожий на гібрид двох – гнучка і відносно проста в управлінні, в той же час дозволяє створення універсальних сайтів, наприклад, магазину електронної комерції або платформи соціальних мереж [10].

Завдяки цим CMS можна з нуля створити свій сайт та налаштувати його як вам потрібно, і постійно з плином часу добавляти новий та замінювати старий контент на сторінці. Вони прекрасно підходять для веб-сторінок з динамічним вмістом, тобто контентом, який постійно необхідно змінювати та доповнювати. Оскільки розроблюваний проект задумується як сайт-візитка, так

званий «landing page», на ньому не передбачена постійна зміна вмісту у великій кількості. Саме тому, щодо вибору необхідної системи керування вмістом, було прийнято рішення про використання продукту під назвою Textolite. Саме вона, на мою думку, являється прекрасним рішенням для управління сайту зі статичним контентом.

Всі зміни, які вносяться за допомогою цієї системи зберігаються прямо у html-файлі, що позбавляє необхідності мати сервер баз даних. Їй не потрібно ніякої інтеграції із сайтом, достатньо розмістити каталог у якому міститься сама система у корінь сайту.

Основною особливістю Textolite є візуальний редактор з можливістю редагувати вміст сайту без будь-яких форм введення. Для більш серйозних змін передбачений редактор вихідного коду з підсвічуванням синтаксису і нумерацією рядків.

Є також зручний файловий менеджер з функцією багатопотокового пакетного завантаження файлів на сервер. Приємний, простий інтерфейс редактора дозволяє людині без спеціальних навичок та знань без проблем редагувати увесь текст на сайті, а у розширеній платній версії дозволяє клонувати цілі блоки контенту, переміщувати їх та замінювати зображення на сторінці. Також, це продукт із відкритим вихідним кодом, що дає можливість розробляти на його основі своє програмне забезпечення. Textolite також дбає про захист від несанкціонованого доступу, представляючи блокування IP-адреси на визначений час після введення неправильного паролю декілька разів.

Механізм автоматичного блокування ботів на сторінці авторизації також присутній. В цілому, Textolite на мою думку являється найкращою системою керування вмістом статичних сайтів, і саме тому вона була обрана для використання у розроблюваному проекті [11].

2.2 Алгоритми роботи сайту

Як ми уже знаємо, веб-сторінки у мережі Інтернет працюють за протоколом HTTP. Веб-браузер виступає як клієнт, та надсилає запит серверу на отримання документу, коли здійснює клацання на гіперпосилання. Веб-сервер у свою ж чергу, надсилає у відповідь необхідний клієнтом документ, якщо той існує. Що ж відбувається після отримання? Розглянемо детальніше, яким чином браузер обробляє отриманий файл та відображає веб-сторінку.

Більшість сторінок у мережі Інтернет містять зображення, каскадні таблиці стилів та скріпти, написані мовою Javascript. Ці речі називаються ресурсами і для відображення веб-сторінки браузер також повинен їх отримати.

Отож, яким чином він знає, чи необхідні сторінці додаткові ресурси? Після того, як браузер отримав html файл, він проводить процес парсингу, який означає читання файлу з метою пошуку чогось у ньому. Якщо ресурси були знайдені у html файлі, то браузер також надсилає запит серверу на їх отримання. Коли відбулося завантаження якого-небудь файлу, також відбувається його парсинг [12].

Після завершення збирання усіх ресурсів наступає етап збірки сторінки і він полягає в об'єднанні інформації, що міститься в документі (оригінальному файлі HTML), та інформації, що міститься в ресурсах. Для побудови сторінки в браузері існують три кроки: створення DOM, CSSOM та дерева візуалізації.

DOM(англ. «Document Object Map») – це карта того, як речі відображаються на сторінці відповідно до HTML. DOM відображає те, що говорить HTML, розподіляючи елементи на сторінці відповідним способом.

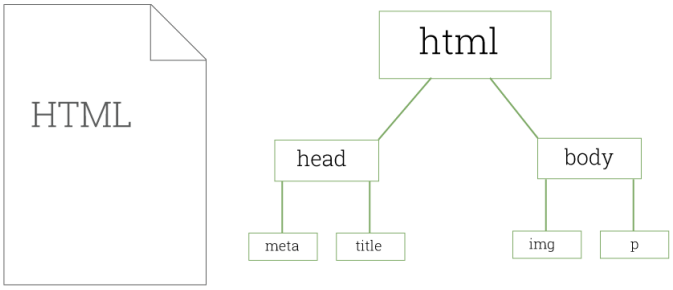

Рисунок 2.1 – Графічне представлення DOM

CSSOM(англ. «CSS Object Map») – це карта того, які стилі повинні застосовуватися до різних частин сторінки відповідно до CSS. CSSOM відображає, яким чином речі повинні бути представлені за допомогою стилів.

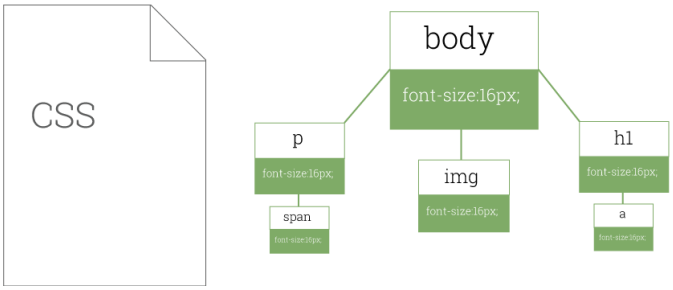

Рисунок 2.2 – Графічне представлення CSSOM

Дерево візуалізації поєднує DOM та CSSOM, створюючи повну карту того, як сторінка буде побудована та який вона матиме вигляд.

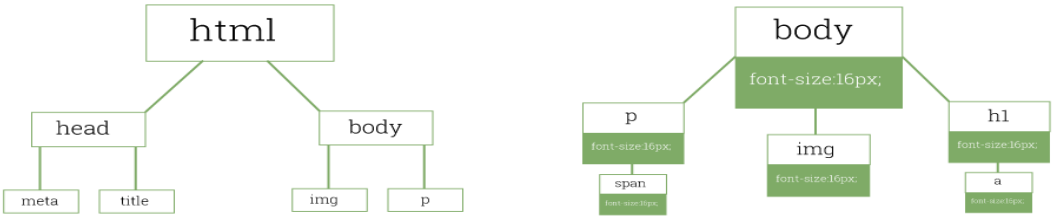

Рисунок 2.3 – Графічне представлення дерева візуалцізаціі

Після виконання кроків описаних вище, браузер врешті може відобразити щось на екрані користувача. Для цього, йому необхідно провести макетування та сам процес відображення.

На цьому етапі, браузер знає які елементи та стилі він має відображати (завдяки DOM та CSSOM) і яким чином вони пов’язані (дерево візуалізації). Єдине, чого він не знає – це розмір, у якому можна усе розмістити, і де буде знаходитися закінчення на екрані. Даний етап носить назву «макетування» і у ньому проводиться визначення ширини і висоти дисплею та обчислення розміру елементів на сторінці, оскільки деякі з них можуть відображатися у процентному відношенні від усього вмісту. У кінці, створюється макет сторінки та проводиться процес відображення у якому відбувається перетворення кожного вузла дерева відображення у фактичні пікселі на дисплеї.

2.3 Структура бази даних

База даних – це будь-який збір даних або інформація, яка спеціально організована для швидкого пошуку та пошуку за допомогою комп'ютера. Бази даних побудовані так, щоб полегшити зберігання, пошук, модифікацію і видалення даних у поєднанні з різними операціями обробки даних. Система управління базами даних (СУБД) витягує інформацію з бази даних у відповідь на запити [13].

База даних складається з декількох таблиць. Так само, як і таблиці Excel, таблиці баз даних складаються з стовпців і рядків. Кожен стовпець відповідає атрибуту, і кожен рядок відповідає одному запису. Кожна таблиця повинна мати унікальну назву в базі даних. Наприклад, розглянемо таблицю бази даних, яка містить імена та номери телефонів. Ви, напевно, встановите стовпці з назвою "Ім’я", "Прізвище" і "Номер телефону". Тоді ви просто почнете додавати рядки під стовпцями, які містять дані.

Важливим аспектом таблиці є те, що кожна з них повинна мати стовпець первинного ключа, для того щоб кожен рядок мав унікальне поле для його ідентифікації. Інформація в базі даних захищена обмеженнями. Обмеження

застосовують правила щодо даних для забезпечення загальної цілісності. Наприклад, унікальне обмеження гарантує, що первинний ключ не можна дублювати. Обмеження перевірки контролює тип даних, які можна ввести – наприклад, поле Ім'я може приймати звичайний текст, але поле номера телефону має містити лише цифри. Існують також деякі інші типи обмежень.

Однією з найпотужніших можливостей бази даних є можливість створювати відносини між таблицями за допомогою зовнішніх ключів. Наприклад, у вас може бути таблиця клієнтів і таблиця замовлень. Кожен клієнт може бути прив'язаний до замовлення в таблиці "Замовлення". Таблиця замовлень, у свою чергу, може бути пов'язана з таблицею продуктів.

Такий дизайн містить реляційну базу даних і спрощує розробку бази даних, щоб можна було організувати дані за категоріями, а не намагатися помістити всі дані в одну або лише кілька таблиць.

Для створення правильної структури бази даних перш за все необхідно обміркувати, для яких цілей вона буде використовуватися. Перше, про що потрібно подумати, це поля, які ви збираєтеся використовувати. Поля є категоріями інформації, яку зберігатиме ваша база даних. Наприклад, більшість шкіл видають заслуги студентам, які добре працювали.

Щоб створити базу даних по достоїнствах, вам слід включити такі поля, як ім'я студента, клас, тему, дату, причину заслуги, класного керівника тощо. Вибравши поля, які ви збираєтеся використовувати, можна починати вводити дані. Вся інформація для однієї особи або речі, тобто інформація для всіх полів, зібраних разом, називається записом. Таким чином, у прикладі бази даних достоїнств кожен студент, який мав заслугу, мав би запис у базі даних.

Більшість програм баз даних дозволяють, або, скоріше, вимагають, щоб ви надали кожному полі бази даних тип. Тип поля вказує, який тип інформації буде зберігатися в цьому полі. Загальні типи полів:

- цілі числа;
- десяткові числа;
- текст;

– дати;

– логічний (або так / ні).

Деякі більш удосконалені бази даних також дозволяють зберігати мультимедійні об'єкти, такі як зображення або звуки. Типи полів є базовим типом перевірки, оскільки база даних не дозволяє вводити, скажімо, текст у полі дати. Вони також полегшують сортування, так як база даних "знає" більше про вміст поля. Наприклад, якщо ви зберігали номери в текстовому полі, а потім сортували їх, число 11 у відсортованому списку йшло б раніше числа 2, оскільки символ 1 з'являється до 2, а номери будуть сортуватися в алфавітному порядку.

Для того, щоб створити логічну структуру в базі даних, має бути щось унікальне для кожного запису. Зазвичай це вміст одного конкретного поля, яке називається полем ключа. Наприклад, якщо у вас є база даних автомобілів, то ви можете використовувати реєстраційний номер як унікальний ключ, так як у кожного автомобіля він повинен бути неповторний. Іноді ви не можете визначити одне поле ключа у таблиці бази даних. Наприклад, у базі даних по достоїнствах не було б жодної справи, яка б була унікальною. Це не може бути дата, тому що в день може бути більше однієї заслуги. Це не може бути вчитель або учень, тому що це означало б, що вони можуть лише давати або отримувати одну заслугу. У таких випадках, ви можете створити те, що відомо як складний ключ – це комбінація полів, які є унікальними. Це може бути будь-яка кількість полів, але повинна бути мінімальна кількість, необхідна для створення унікального опису запису.

Наприклад, у базі даних по достоїнствах можна поєднати студента, дату та предмет. Останнім, що можна створити для своїх полів, є індекс. Після того, як ви почали використовувати базу даних, інформацію можна зберігати в таблиці в будь-якому порядку, можливо, в тому порядку, в якому ви ввели. Якщо у вашій базі є багато записів, сортування та пошук може тривати довго.

Індекс подібний до індексу в книзі – це додатковий біт, доданий до бази даних, щоб допомогти програмі управління швидко знайти записи. Вам не

рекомендується індексувати кожне поле у вашій базі даних, оскільки існують додаткові витрати при створенні записів, але ви повинні індексувати всі ключові поля та будь-які поля, які ви регулярно використовуєте для пошуку або сортування [14].

Для розроблюваного проекту було прийнято рішення про створення бази даних на базі движка Mysql. MySQL – це найбільш використовувана та найбільш відома реляційна база даних. При проектуванні бази даних було дотримано певну послідовність кроків.

Крок 1. Визначено предметну область для якої створюється база даних. Від цього залежить специфіка створення таблиць. У нашому випадку нею є сфера надання послуг.

Крок 2. Визначення таблиць. Для проекту з даною предметною областю необхідні таблиці клієнтів, замовлень та послуг які надаються.

Крок 3. Визначення атрибутів таблиць, їхніх ключів.

Крок 4. Визначення додаткових операцій, таких як тригери, процедури, функції.

Для кожної таблиці існує первинний ключ із назвою id, що спрощує побудову запитів та дозволяє пов’язати таблиці. У таблиці послуг(services), окрім первинного ключа є ще три атрибути – це назва сервісу, ціна та опис (див. рисунок 2.4). У таблиці клієнтів(clients) – ім’я, прізвище, електронна пошта, адреса і телефон. На рисунку 2.5 зображено схему таблиці. Сутність замовлень(orders) має атрибут client_id, який є зовнішнім ключем та відноситься до первинного ключа id у таблиці clients. Таким чином, неможливо просто створити замовлення, воно повинно бути пов’язане із певним клієнтом. Також, у неї є ще атрибути для цін – services_price, additional_price та total_price, які означають ціну за послуги, додаткові витрати(наприклад на матеріали) та повну ціну замовлення яка вираховується із двох попередніх. Також, у таблиці є атрибут description, який дозволяє записати опис замовлення, тобто додаткові важливі речі, які його стосуються. (див. рисунок 2.6).

Останньою є таблиця orders_services, яка пов’язує між собою зв’язком багато до багатьох таблиці послуг та замовлень. Таким чином, у ній зберігаються первинні ключі замовлення та послуги, тобто до одного замовлення можна прикріпити декілька замовлених послуг. Здійснюється ця прив’язка за допомогою зовнішніх ключів order_id та service_id, які відносяться до первинних ключів таблиць orders та services відповідно. Оскільки, ціна послуги вказується за годину, то було добавлено атрибут hours, який вказує на кількість годин, витрачених для надання послуги.Усі таблиці створені на движку InnoDB, який дозволяє здійснювати прив’язку за допомогою зовнішніх ключів, та кодуванням utf8_general_ci, яке дозволяє без перешкод здійснювати ввід та зберігання даних не тільки на латиниці а й кирилиці.

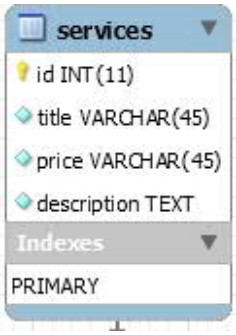

Рисунок 2.4 – Таблиця послуг (services)

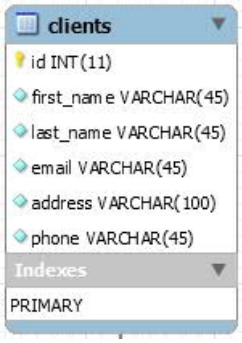

Рисунок 2.5 – Таблиця клієнтів(clients)

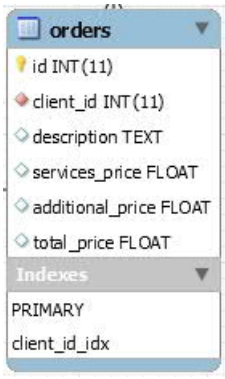

Рисунок 2.6 – Таблиця замовлень(orders)

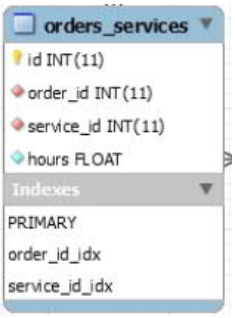

Рисунок 2.5 – Таблиця orders_services

На рисунку 2.6 зображена EER діаграма яка відображає таблиці, їхні атрибути, ключі, індекси та найголовніше – відношення між таблицями.

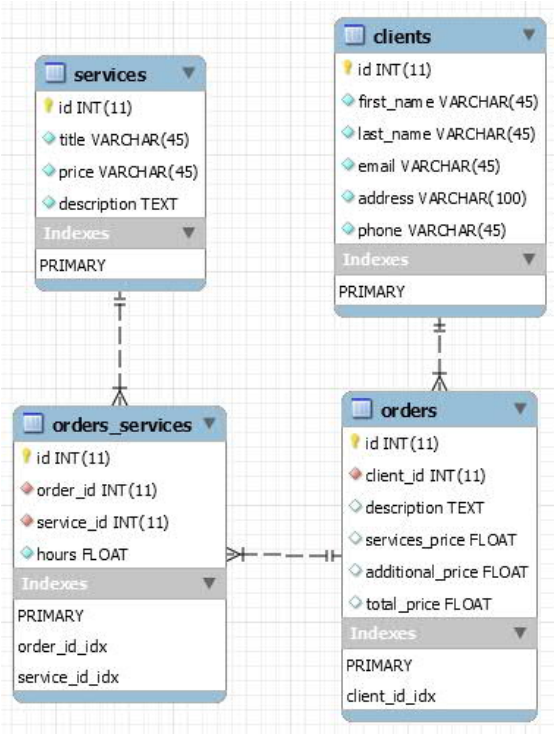

Рисунок 2.6 – Структура бази даних

На даному рисунку чітко видно назви таблиць, зв’язки між ними, їхні атрибути, первинні та зовнішні ключі. Дана структура, на мою думку, являється оптимальною для такої предметної області. За бажанням, її можна доповняти та розширювати, наприклад добавити окрему таблицю телефонних номерів та адрес для можливості збереження декількох телефонних номерів та декількох адрес для одного клієнта.

Для того, щоб автоматично вираховувати ціну за послуги та загальну ціну замовлення були добавлені тригери. Тригер – це збережена процедура, яку неможливо безпосередньо викликати. Вона виконується при певних діях, таких як внесення даних(команда INSERT), редагування запису(команда UPDATE) та видалення запису(команда DELETE) [15].

Для реалізації автоматичного обрахування цін загалом необхідно 4 тригери, три з яких відносяться до таблиці order_services та один до таблиці orders. Для початку, було створено тригер, який виконується після внесення даних у таблицю orders_services. Він прикріпляється до події під назвою AFTER INSERT. Алгоритм його роботи складається із 5 кроків.

Крок 1. Береться ключ послуги внесений у таблицю та знаходиться відповідна йому послуга у таблиці services.

Крок 2. Далі, із даної таблиці береться ціна за годину для послуги.

Крок 3. Ціна множиться на кількість годин вказаних у таблиці orders_services.

Крок 4. Береться ключ замовлення внесений у таблицю та знаходиться відповідне йому замовлення у таблиці orders.

Крок 5. До поля services_price у таблиці orders додається розрахована ціна.

Таким чином, при додаванні інформації у таблицю order_services автоматично обновляється поле services_price із таблиці orders. Описується дана логіка на мові SQL наступним чином:

```
CREATE DEFINER=`root`@`localhost` TRIGGER
```

```
`euroterservice`.`orders_services_AFTER_INSERT_1` AFTER INSERT ON
`orders_services` FOR EACH ROW
BEGIN
     UPDATE orders SET orders.services_price =
orders.services_price +
     (SELECT (services.price * NEW.hours)
     FROM services
     WHERE services.id = NEW.service_id)
     WHERE orders.id = NEW.order_id;
END
```

У даному прикладі для доступу до даних які були введені у запиті використовується оператор NEW. Також, було створено тригер для здійснення оберненої операції, тобто віднімання ціни послуги при видаленні рядка із таблиці. Він виконується при настанні події, яка носить назву AFTER_DELETE. Код тригера представлено нижче.

```
CREATE DEFINER=`root`@`localhost` TRIGGER
`euroterservice`.`orders_services_AFTER_DELETE` AFTER DELETE ON
`orders_services` FOR EACH ROW
BEGIN
     UPDATE orders SET orders.services_price =
orders.services_price –
     (SELECT (services.price * OLD.hours)
     FROM services
     WHERE services.id = OLD.service_id)
     WHERE orders.id = OLD.order_id;
END
```

У ньому було використано оператор OLD який дозволяє звернутися до інформації, яка знаходиться на даний момент у рядку таблиці, який намагаються видалити.

Третій тригер необхідний при оновленні даних у таблиці. Він викликається при настанні події AFTER UPDATE. Особливістю даного тригера є використання обох операторів NEW та OLD а також умовної конструкції. Це зумовлено тим, що при оновленні може відбуватися багато змін, і необхідно врахувати усі можливі варіанти розвитку подій. Код тригера:

```
CREATE DEFINER=`root`@`localhost` TRIGGER
`euroterservice`.`orders_services_AFTER_UPDATE` AFTER UPDATE ON
`orders_services` FOR EACH ROW
BEGIN IF (NEW.order_id <> OLD.order_id) THEN
UPDATE orders SET orders.services_price = orders.services_price
     - SELECT(services.price * OLD.hours) FROM services WHERE
     services.id = OLD.service_id) WHERE orders.id = OLD.order_id;
     UPDATE orders SET orders.services_price =
orders.services_price
     + (SELECT(services.price * NEW.hours) FROM services WHERE
     services.id = NEW.service_id) WHERE orders.id = NEW.order_id;
ELSE UPDATE orders SET orders.services_price =orders.services_price
(SELECT (services.price * NEW.hours) FROM services WHERE
services.id = OLD.service_id)+(SELECT (services.price * NEW.hours)
     FROM services WHERE services.id = NEW.service_id);END IF;END
```

Для повного автоматичного обрахування цін необхідне додавання ще одного тригера, який буде обраховувати загальну ціну для замовлення, тобто поле total_price при зміні даних у таблиці orders. Він запуститься при настанні події BEFORE UPDATE:

```
BEFORE UPDATE.
CREATE DEFINER=`root`@`localhost` TRIGGER
`euroterservice`.`orders_BEFORE_UPDATE` BEFORE UPDATE ON `orders`
FOR EACH ROW
BEGIN
     SET NEW.total_price = NEW.services_price +
NEW.additional_price;
END
```

У результаті проектування мною було створено базу даних для підприємства Євротерсервіс. Вона складається із 4 сутностей – клієнти, послуги, замовлення та черга замовлень. Встановлені відношення між таблицями дозволяють зберігати її цілісність та структуру, а розроблені тригери автоматично розраховують ціни для замовлень. На мою думку, дана розробка являється оптимальною для цього підприємства. При його зростанні, дану базу можна із легкістю розширювати та удосконалювати.

3 ПРАКТИЧНА ЧАСТИНА РОЗРОБКИ ВЕБ-САЙТУ

3.1 Створення сторінки

3.1.1 Розробка дизайну

Цільова сторінка(landing page) насамперед спрямована на залучення нових відвідувачів та перетворення їх на клієнтів. Така дія називається конверсією. Збільшення їх кількості напряму залежить від вигляду вашого сайту, швидкості його завантаження та ряду інших факторів. Дизайн цільової сторінки - це перше враження, яке складеться про ваш продукт або послугу, і згідно з ConversionXL перші враження пов'язані з дизайном на 94%. Компанія UXmag зазначила, що у нас є лише 50 мілісекунд (0,05 сек), перш ніж користувач зробить перші судження про продукти [16].

Тед Барнс, креативний директор компанії 42connect, підкреслює два міркування, які треба мати на увазі при створенні цільової сторінки.

1. Складність. Чи розумію я мету цієї сторінки, і що я повинен робити менше ніж за секунду?

2. Знайомство. Користувачі очікують, що сторінка виглядатиме як сторінка, яку вони вже знають з будь-якого іншого веб-сайту (наприклад, сторінки продукту виглядають як сторінки продукту, форми контактів виглядають як форми контактів). Сила у простоті, не потрібно перевертати усе з ніг на голову, намагаючись бути оригінальними. У більшості випадків ви повинні дотримуватися того, що працює для інших. Дизайн, який враховує досвід користувача, змусить людей залишатися довше, створюючи більш високі шанси на конверсію.

Для розробки дизайну проекту, було вибрано програмне забезпечення під назвою Figma. Це онлайн сервіс для розробки інтерфейсів і прототипування з можливістю організації спільної роботи у режимі реального часу. Має наявність передплаченої моделі розповсюдження, проте перший проект можна розробляти абсолютно безкоштовно.

При розробці дизайну у першу чергу необхідно визначити загальну структуру проекту. Розроблювана цільова сторінка складається із 7 блоків.

1. Header – так звана “шапка” сайту, у якій зазвичай розміщують логотип та меню навігації.
2. Головний блок – містить у собі заголовок h1, який являється унікальною цільовою пропозицією, мініатюру та заклик до дії у вигляді кнопки.
3. Послуги – у ньому описуються усі послуги, які надає компанія.
4. Про нас – містить інформацію, яка призначена для зацікавлення користувача у співпраці.
5. Відгуки – містить відгуки від клієнтів.
6. Контакти – відображає контактну інформацію компанії та містить форму зворотнього зв’язку.
7. Footer – розміщується у кінці сторінки та містить у собі інформацію про компанію, копірайт, терміни використання та політику конфіденційності.

Також може вміщувати посилання на сторінку компанії у соціальних мережах. Ще одним важливим кроком при розробці дизайну є підбір кольорової схеми. На основі макетів із візитками компанії, було підібрано головний колір – голубий відтінок із шістнадцятковим кодом 00A6FF. Далі, за допомогою колірного круга, методом так званої “тріади” було визначено два гармонічні кольори, які чудово доповнювали основний, а саме – червоний колір із кодом FF0028 та жовтий із кодом FFD200(див. рисунок 3.2). Дана колірна палітра надає широкі можливості при розробці дизайну та дозволяє фокусувати увагу користувача на спеціально визначених місцях.

У результаті наполегливої роботи, за визначеною структурою було виготовлено дизайн сторінки. Перший блок під назвою “Header” містить у собі логотип компанії та меню. Дане меню реалізує навігацію по секціях. Вигляд даного блоку зображено на рисунку 3.1.

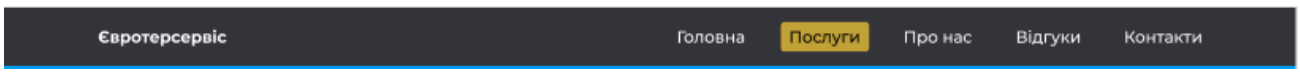

Рисунок 3.1 – Вигляд блоку Header

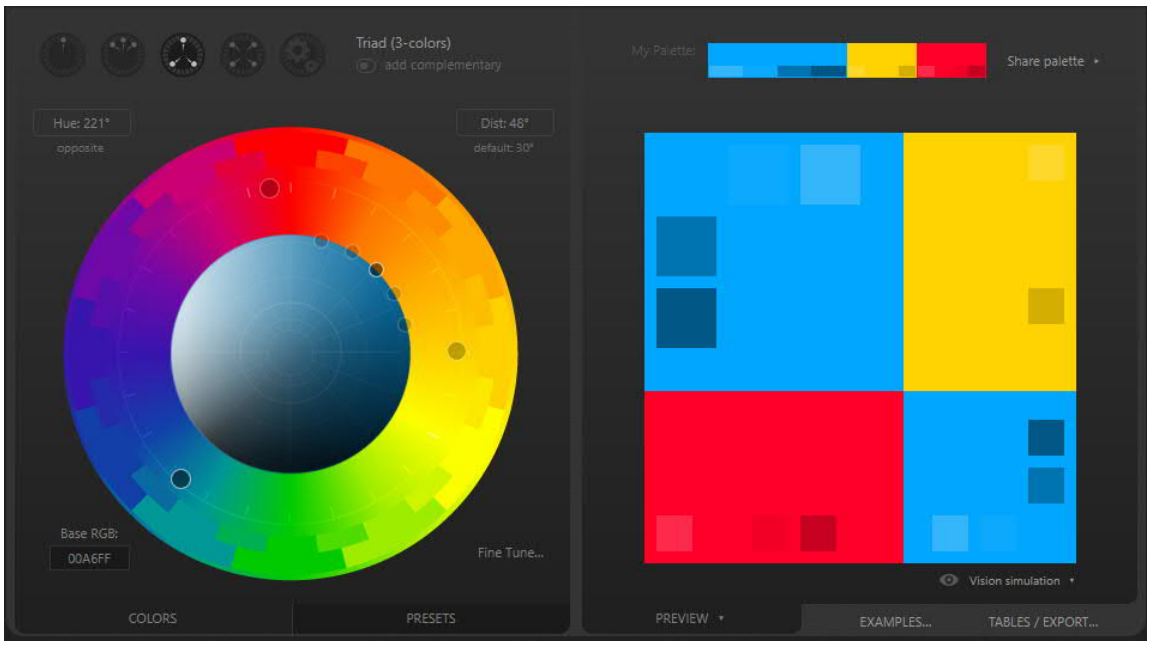

Рисунок 3.2 – Визначення кольорів за колірним колом

Наступним було розробленого головну секцію, вигляд якої зображено на рисунку 3.3. У ній розміщені 4 елементи.

1. Заголовок – коротко описує діяльність компанії.
2. Підзаголовок – повідомляє користувача інформацією про місце діяльності компанії.
3. Кнопка – заклик до дії.
4. Ілюстрація – призначена для заповнення лишнього простору та забезпечення візуальної гармонії.

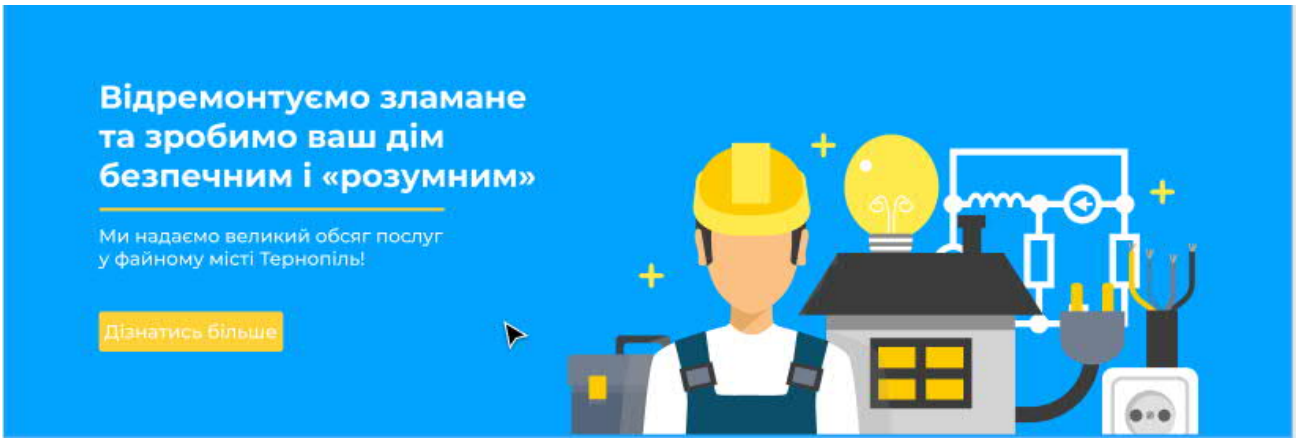

Рисунок 3.3 – Головна секція веб-сторінки

Секція “послуги” надає користувачу інформацію про послуги, які надає компанія. Представлена секція у вигляді змістовних іконок та детального опису. Вигляд блоку зображено на рисунку 3.4.

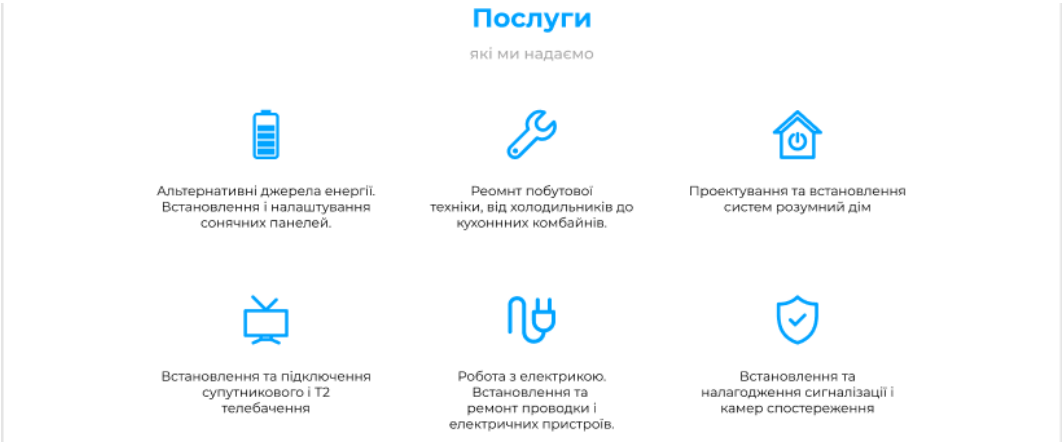

Рисунок 3.4 – Блок “Послуги”

Секція “Про нас” чітко висловлює інформаціє про те, чому користувачу слід обрати саме цю компанію. Дизайн блоку зображений нижче(див. рисунок 3.5).

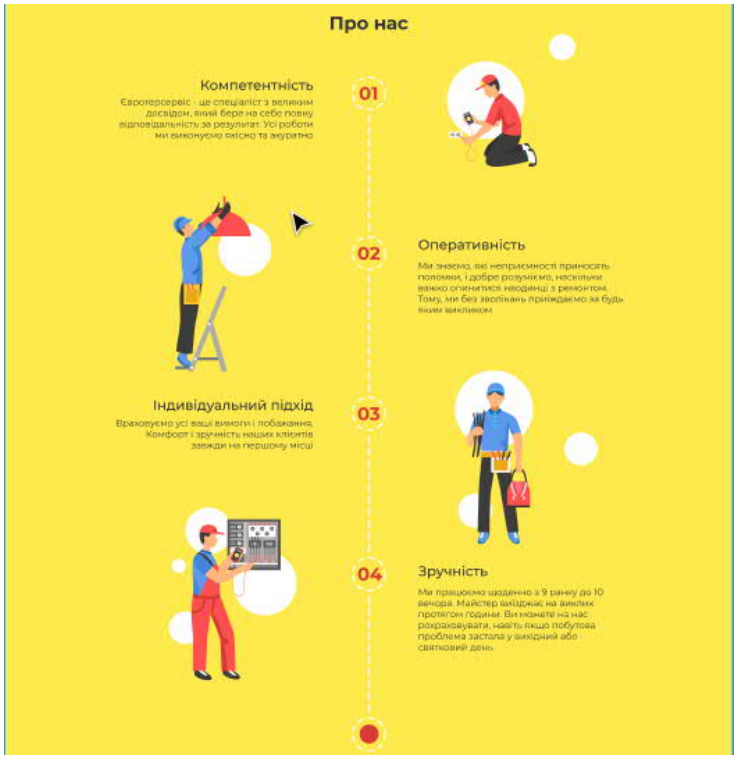

Рисунок 3.5 – Блок “Про нас”

Відгуки допомагать переконати майбутнього клієнта звернутися за допомогою до компанії. Зовнішній вигляд блоку відгуків зображений на рисунку 3.6.

Рисунок 3.6 – Блок “Відгуки”

Найважливішим компонентом поряд із головною секцією виступає контактна форма(див. рисунок 3.7). При нажатті на кнопку, компанії надішлеться лист із вказаними даними. Це забезпечує зворотній зв'язок.

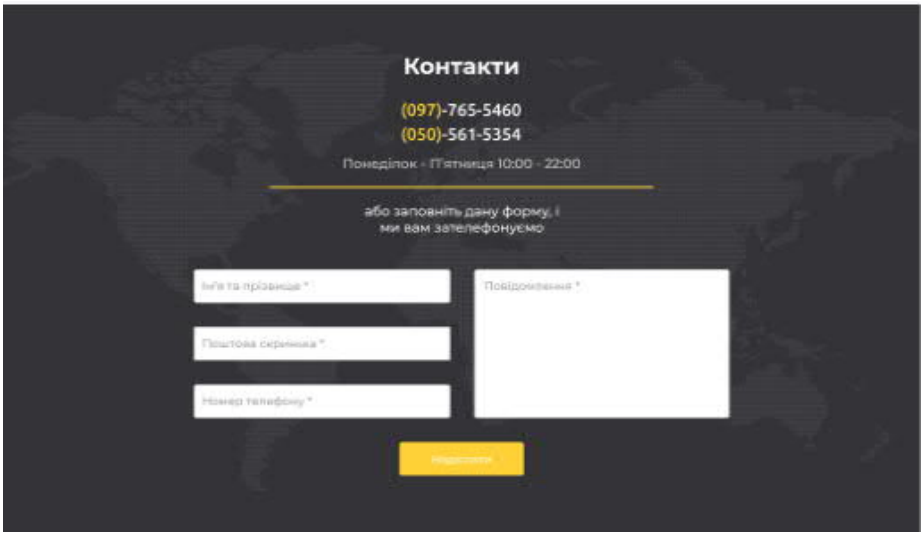

Рисунок 3.7 – Секція “Контакти”

Останнім блоком виступає “Footer”, у якому розміщено інформацію про компанію, копірайт та посилання на соціальні мережі. (див. рисунок 3.8)

Рисунок 3.8 – Блок “Footer”

У результаті було розроблено макет сайту для компанії Євротерсервіс. При його розробці було дотримано принципи побудови цільових сторінок для високих конверсій, а саме наявність:

- унікальної ціннісної пропозції;
- заклику до дії;
- опису переваг роботи із даною компанією;
- форми зворотнього зв'язку.

3.1.2 Верстка сторінки

Для пришвидшення верстки та забезпечення адаптивності розроблюваного сайту було використано інструмент під назвою Bootstrap. Bootstrap – це вільний набір інструментів для створення сайтів і веб-додатків. Включає в себе HTML- і CSS-шаблони оформлення для типографіки, веб-форм, кнопок, міток, блоків навігації та інших компонентів веб-інтерфейсу, включаючи JavaScript-розширення [17]. Найголовнішою перевагою використання цього фреймворку є гнучка сітка, яка дозволяє із легкістю створювати адаптивний дизайн. Приклад HTML коду із використанням bootstrap наведено нижче. Повний HTML код сторінки міститься у додатку А.

```
<section id="main" class="main"> <div class="container">
            <div class="row"> <div class="hero col-lg-5">
                    <h1 class="hero__title">
                        Відремонтуємо зламане та зробимо ваш дім
безпечним і &laquo;розумним&raquo;
                    </h1> <div class="hero__vector"></div>
                    <p class="hero__subtitle">
                        Ми надаємо великий обсяг послуг у файному
місті Тернопіль!
                    </p>
                    <a href="#" class="hero__button button">
                        Детальніше
                    </a> </div>
</div></div> </section>
```

За допомогою додавання тегів div із класами “container”, “row” та “col” здійснюється розмітка макету. Клас “container” вказує на базовий контейнер, ширина якого змінюється при зміні ширини екрану. “Col” вказує на кількість колонок, які займає контент при певному розширенні екрану. У класу “col” існує багато опцій, усі вони наведені у таблиці 3.1.

Таблиця 3.1 – Сіткова система bootstrap

	Дуже малий <576px	Малий ≥576px	Середній ≥768px	Великий ≥992px	Дуже великий ≥1200px
Максимальна ширина контейнера	Немає (auto)	540px	720px	960px	1140px
Префікс класу	.col-	.col-sm-	.col-md-	.col-lg-	.col-xl-
Кількість колонок	12				
Ширина відступів	30px (15px з кожної сторони колонки)				
Вкладення	Так				
Визначення порядку стовбців	Так				

У результаті було розроблено сторінку з підтримкою адаптації під різну ширину екрану. Це позитивно відзначиться на конверсії сайту, оскільки на теперішній день близько 80% користувачів відвідують сайти із мобільних платформ.

3.1 3 Написання скриптів

Для того, щоб зробити наш сайт інтерактивним та збільшити можливості взаємодії користувача із елементами сторінки було прийнято рішення додати на сторінку js код, який налаштовує навігацію. Nav-scroll-spy.js – це невеликий, але корисний Jquery плагін для забезпечення плавної прокрутки під час навігації та захоплення і відображення активного на даний час елементу в меню. Його підключення здійснюється шляхом додавання скриптів на сторінку

через тег <script>. Також, для коректної роботи необхідно підключити бібліотеку jquery.

```
<script src="https://code.jquery.com/jquery-3.3.1.min.js"
        integrity="sha384-
tsQFqpEReu7ZLhBV2VZlAu7zcOV+rXbYlF2cqB8txI/8aZajjp4Bqd+V6D5IgvKT"
        crossorigin="anonymous"></script>
<script src="jq.nav-scroll-spy.js"></script>
```

Далі, необхідно додати спеціальні класи до html розмітки. За допомогою класів nav-item та spy-item здійснюється зв’язок між секціями та пунктами меню.

```
<ul class="nav-group">
  <li class="nav-item active">Item 1</li>
  <li class="nav-item">Item 2</li>
  <li class="nav-item">Item 3</li>
  ...
</ul>
<div class="container">
  <div class="spy-item">Section 1</div>
  <div class="spy-item">Section 2</div>
  <div class="spy-item">Section 3</div>
  ...
</ul>
```

Залишається лише написати скрипт та додати у нього виклик спеціальної функції:

```
<script>
        $(document).jqNavScrollSpy();
        $(document).jqNavScrollSpy({
            speed: 300 // in ms
        });
</script>
```

Таким чином, було налаштовано слідування меню за положенням користувача на сторінці та можливість плавного переходу до різних секцій сайту, що повинно привабити більше користувачів та збільшити кількість можливих конверсій.

3.1.4 Розробка валідації та захисту форми зворотнього зв’язку

Написання валідації та захисту для форм являється однією із першорядних задач, оскільки без цього існує можливість атаки спам-ботів. Це відповідно призводить до зниження конверсій через неможливість пошуку відповідей від реальних клієнтів.

Насамперед, для валідації полів форми слід використовувати відповідні семантичні теги мови html. Так, наприклад тег <input type=”email”> не дозволить користувачу відправити запит, якщо у даному полі не було знайдено ключового символу “@”. Існує багато способів фільтрації вмісту форми, мною було виділено чотири з них.

1. Позначення полів атрибутом “required”, який не дозволяє відправку форми без заповнення відповідних полів із цим тегом.
2. Використання підходящих типів тегу input: text, email, password, url, number, date і тд.
3. Використання placeholder для надання користувачу інформації про те, що ми очікуємо отримати у даному полі.
4. Патерни для input, які дозволяють гнучко налаштувати фільтрацію вхідних дані [19].

Нажаль, валідація через використання html не являється потенційним захистом для просунутих зловмисників. Необхідно також здійснювати обов’язкову валідацію на стороні сервера. У цьому допоможе серверна мова програмування PHP зі своїми готовими пресетами фільтрів та можливістю співставлення даних із регулярними виразами. Приклад валідації форми наведений нижче.

```
public function isEmailValid($email)
    {
        if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
            return 'Email is not valid';
        }
        return '';
    }
```

```
    public function isPhoneValid($phone)
    {
        if (!preg_match('/^[+][3][8][0][0-9]{2}[-][0-9]{2}[-][0-
9]{2}[-][0-9]{3}+$/', $phone)) {
            return 'Phone is not valid';
        }
        return '';
    }
```

Щодо захисту від спам-ботів, мною було додано генерацію спеціального ключа у сесії користувача, для забезпечення відправки форми тільки із вікна браузера. Принцип генерації та перевірки ключа описаний нижче.

```
<?php
$token = bin2hex(random_bytes(32));
$_SESSION['token'] = $token;
?>
<form action="comment.php" method="post">
    <input type="hidden" name="token" value="<?php echo $token;
?>">
</form>
<?php
// comment.php
if ($_SESSION['token'] !== $_POST['token']) {
    // It's SPAM
}
?>
```

3.2 SEO оптимізація

Оптимізація пошукової системи стосується процесу редагування елементів веб сторінки для підвищення ймовірності відображення сайту у результатах пошуку, коли хтось вводить відповідний запит.

При побудові веб сторінки , для її оптимізації у пошукових системах необхідно дотримуватися чотирьох порад.

1. Визначення ключових слів. Необхідно скласти список термінів, які мають відношення до вашої сторінки, і спробувати з’ясувати, які комбінації цих

термінів хтось може ввести в Google із наміром знайти те, що ви можете запропонувати.

2. Стратегічно розставляти ключові слова на вашій сторінці. Найбільшу вигоду можна отримати шляхом написання ключових слів у тегах title, h1, meta description та ін.

3. Забезпечення посилань на вашу сторінку з інших сайтів. Це являється найшвидшим способом підняття рейтингу у пошукових системах.

4. Оптимізація швидкості завантаження сторінки. Google підтвердив, що швидкість сторінки має важливу роль у визначенні рейтингу [20].

Для просування веб-сайту було написано ряд ключових слів та “meta description” для назви сайту.

```
<meta name="description" content="Євротерсервіс - найбільша у
Тернополі компанія-постачальник технічних послуг">
<meta name= "keywords" content="Ремонт побутової техніки,
Сигналізація, Відеоспостереження, Розумний дім, технічні послуги у
Тернополі,  робота з електрикою, встановлення телебачення
Тернопіль, супутникове телебачення Тернопіль, Т2 Тернопіль,
сонячні панелі у Тернополі">
```

Очікуваним результатом є підняття сторінки у рейтингу пошукових двигунів. Для забезпечення більших конверсій необхідно додавати більше ключових слів та добре міркувати над їх змістом.

3.3 Тестування та верифікація

Для тестування розробленого проекту було прийнято рішення про його розміщення на безкоштовному хостинг провайдері під назвою “000webhost”. Це дозволить здійснювати перевірку сторінки на швидкість загрузки, валідацію коду, перевірку працездатності системи керування вмістом та інше.

Спершу, провалідуємо код за допомогою посилання

“https://validator.w3.org/”. Даний валідатор чітко вказує на всі помилки, допущені при написанні html коду. Результат його перевірки зображений на рисунку 3.9.

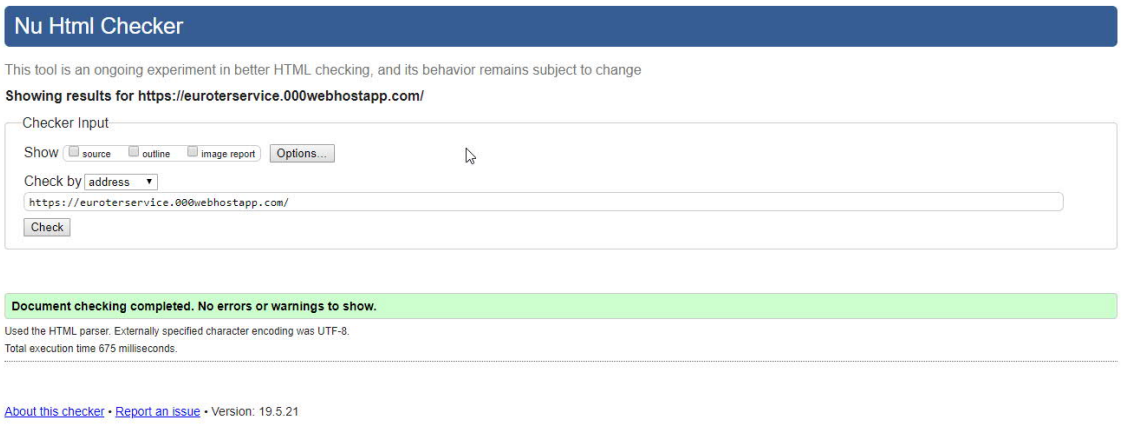

Рисунок 3.9 – Перевірка html коду валідатором

Виходячи із результатів перевірки, помилок при написанні html коду допущено не було.

Наступним кроком є аналіз швидкості завантаження за допомогою сервісу “Google PageSpeed Insights”. Він аналізує вміст веб-сторінки та пропонує рішення, які дозволяють пришвидшити її завантаженея. Оскільки швидкість напряму впливає на рейтинг сторінки у пошуковому двигуні Google, даною перевіркою не потрібно нехтувати.

У результаті, проект набрав 95 та 99 балів за перевірку мобільної версії та десктопної версії сторінки. Усі дані наведені на рисунках 3.10 та 3.11.

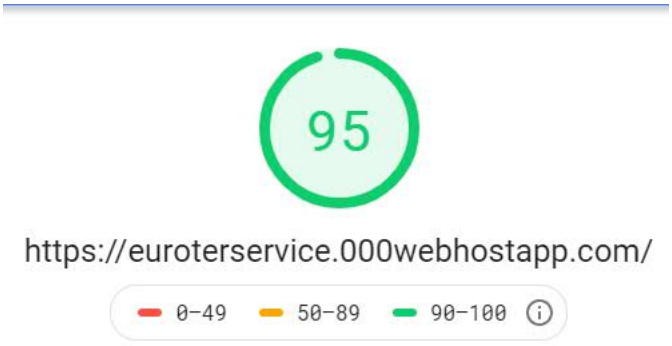

Рисунок 3.10 – Оптимізація завантаження мобільної версії сторінки

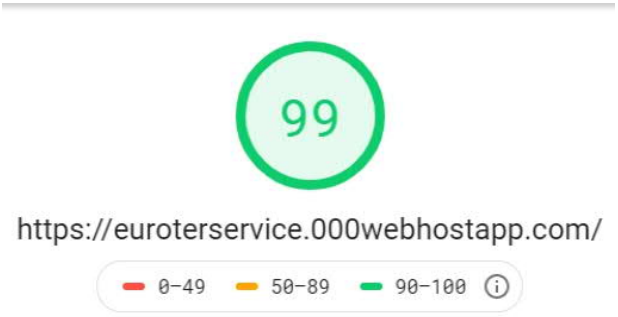

Рисунок 3.11 – Оптимізація завантаження комп’ютерної версії сторінки

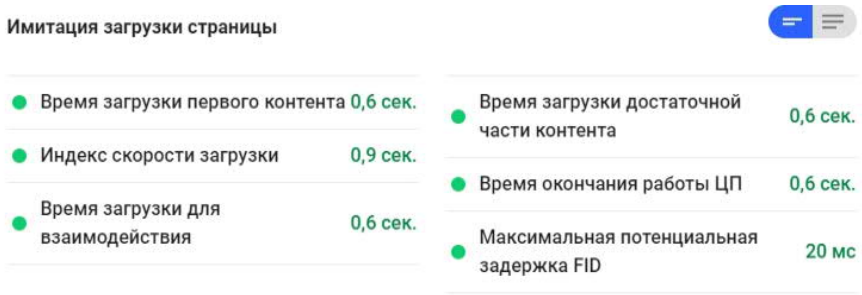

Рисунок 3.12 – Статистика Google PageSpeed Isights

Також, важливо перевірити відображення сайту на малих розширеннях екрану. Симулюємо вхід із різних пристрою за допомогою інструментів розробника браузера Google Chrome. Результати тестування зображені на рисунку 3.13 та 3.14.

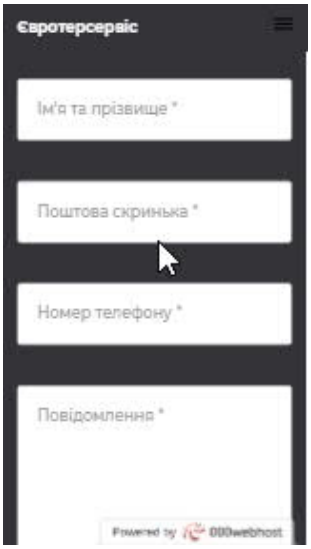

Рисунок 3.13 – Відображення сторінки на смартфоні

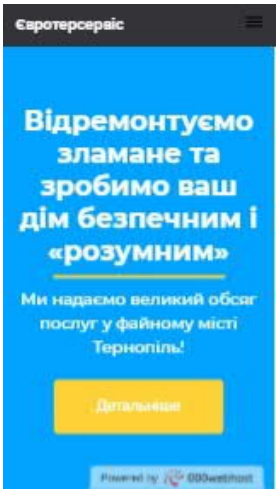

Рисунок 3.14 – Відображення сторінки на смартфоні

Завершальним кроком є перевірка роботи системи керування вмістом. Для цього необхідно перейти в адміністративну панель “textolite”, та спробувати внести зміни. Результат тестування зображено на рисунках 3.15 та 3.16.

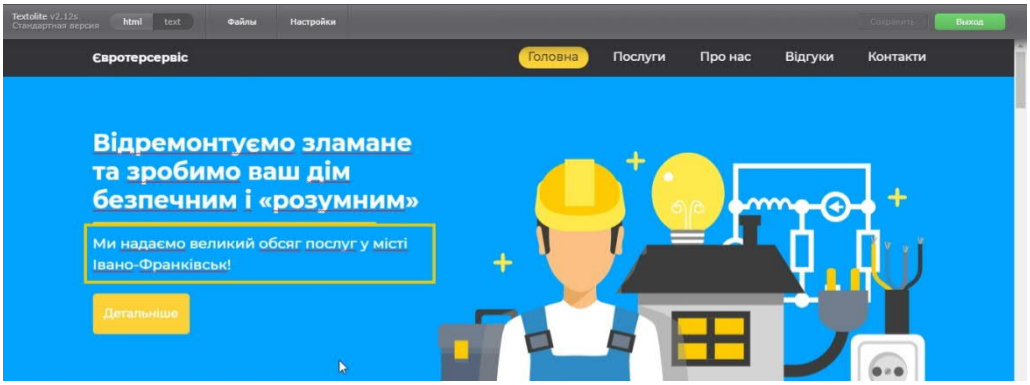

Рисунок 3.15 – Тестування системи керування вмістом

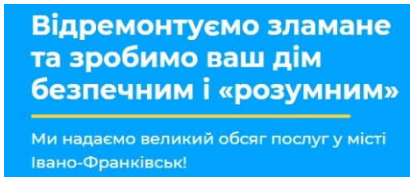

Рисунок 3.16 – Результат тестування системи керування вмістом

Зміни успішно були внесені та збрежені. Сторінка функціонує правильно, та оптимізована для швидкого завантаження.

4 ТЕХНІКО-ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ РОЗРОБКИ ПРОГРАМНОГО ЗАСОБУ

В цьому розділі дипломного проекту проводиться економічне обґрунтування доцільності розробки веб-сайту компанії «Євротерсервіс». Зокрема, здійснюється розрахунок витрат на розробку програмного забезпечення, експлуатаційних витрат, ціни споживання програмного забезпечення. В заключній частині визначаються показники економічної ефективності нового програмного продукту, обґрунтовуються відповідні висновки.

Розроблене програмне забезпечення призначене для виходу компанії на новий рівень, що призводить до притоку нових клієнтів.

4.1 Розрахунок витрат на розробку програмного забезпечення

Витрати на розробку і впровадження програмних засобів (К) обраховуються за формулою (4.1).

$$K = K_1 + K_2 \ , \tag{4.1}$$

де K_1 – витрати на розробку програмних засобів, грн;

K_2 – витрати на відлагодження і дослідну експлуатацію програми рішення задачі на комп’ютері, грн.

Витрати на розробку програмних засобів включають:

- витрати на оплату праці розробників ($B_{ОП}$);
- витрати на відрахування у спеціальні державні фонди ($B_Ф$);

– витрати на покупні вироби ($П_В$);

– витрати на придбання спецобладнання для проведення експериментальних робіт ($О_Б$);

– накладні витрати ($Н_В$);

– інші витрати ($І_В$).

4.1.1 Розрахунок витрат на оплату праці

Витрати на оплату праці включають заробітну плату (ЗП) всіх категорій працівників, безпосередньо зайнятих на всіх етапах проектування. Розмір ЗП обчислюється на основі трудомісткості відповідних робіт у людино-днях та середньої ЗП відповідних категорій працівників.

У розробці програмного забезпечення задіяні наступні спеціалісти - розробники, а саме – керівник проекту, студент-дипломник, консультант техніко-економічного розділу.

Таблиця 4.1 – Вихідні дані для розрахунку витрат на оплату праці

Посада виконавців	Місячний оклад, грн
Керівник ВКР, викладач	5950
Консультант техніко-економічного розділу, доцент	7293
Студент	1400

Витрати на оплату праці розробників проекту визначаються за формулою (4.2).

$$B_{ОП} = \sum_{i=1}^{N} \sum_{j=1}^{M} n_{ij} \cdot t_{ij} \cdot C_{ij}, \quad (4.2)$$

де n_{ij} – чисельність розробників *i*-ої спеціальності *j*-го тарифного розряду, осіб;

t_{ij} – затрачений час на розробку проекту співробітником *i*-ої спеціальності *j*-го тарифного розряду, год;

C_{ij} – годинна ставка працівника *i*-ої спеціальності *j*-го тарифного розряду, грн.

Середньо годинна ставка працівника може бути розраховується за формулою (4.3).

$$C_{ij} = \frac{C_{ij}^0(1+h)}{РЧ_i}, \quad (4.3)$$

де C_{ij}^0 – основна місячна заробітна плата розробника *i*-ої спеціальності *j*-го тарифного розряду, грн;

h – коефіцієнт, що визначає розмір додаткової заробітної плати (при умові наявності доплат);

$РЧ_i$ – місячний фонд робочого часу працівника *i*-ої спеціальності *j*-го тарифного розряду, год (приймаємо 168 год).

Результати розрахунку записують до таблиці 4.2.

Таблиця 4.2 – Розрахунок витрат на оплату праці

Посада виконавців	Час розробки, год	Погодинна заробітна плата, грн/год	Витрати на розробку, грн
Керівник ВКР, доцент	16	35,4	566,4
Консультант техніко-економічного розділу, доцент	2	63,8	127,6
Студент	120	8,3	996
Разом	1690		

4.1.2 Відрахування на соціальні заходи

Величну відрахувань у спеціальні державні фонди визначають у відсотковому співвідношенні від суми основної та додаткової заробітних плат. Згідно діючого нормативного законодавства сума відрахувань у спеціальні державні фонди складає 20,5 % від суми заробітної плати. Розрахунки проведено у формулі (4.4).

$$B_{\Phi} = \frac{20,5}{100} \cdot 1690 = 346,45 \text{ (грн).} \qquad (4.4)$$

4.1.3 Розрахунок витрат на матеріали та комплектуючі

У таблиці 4.3 наведений перелік купованих виробів і розраховані витрати на них.

Таблиця 4.3 – Розрахунок витрат на матеріали та комплектуючі

Найменування купованих виробів	Одиниця виміру	Ціна, грн	Кількість куповани х виробів	Сума, грн	Транспортні витрати (10% від суми)	Загальна сума, грн
Папір (формат А4)	уп	90,0	1	90,00	9,0	99,0
Ручка кулькова	шт	15,0	2	30,00	3,00	33,0
Олівець простий	шт	5,0	2	10,00	1,0	11,0
Зошит, 18 арк	шт	5,50	1	5,50	0,55	6,50
Тонер для принтера	уп	50	1	50	5,0	55,0
Разом						204,05

4.1.4 Витрати на використання комп’ютерної техніки

Витрати на використання комп’ютерної техніки включають витрати на амортизацію комп’ютерної техніки, витрати на користування програмним забезпеченням, витрати на електроенергію, що споживається комп’ютером. За даними обчислювального центру ТНЕУ для комп’ютера типу IBM PC/ATX вартість години роботи становить 5,2 грн. Середній щоденний час роботи на комп’ютері – 2 години. Розрахунок витрат на використання комп’ютерної техніки приведений в таблиці 4.4.

Таблиця 4.4 – Розрахунок витрат на використання комп’ютерної техніки

Назва етапів робіт, при виконанні яких використовується комп’ютер	Час використання комп’ютера, год.	Витрати на використання комп’ютера, грн.
Проведення досліджень та оформлення їх результатів	75	390
Оформлення техніко-економічного розділу	10	36
Оформлення ВКР	15	54
Разом	100	360

4.1.5 Накладні витрати

Накладні витрати проектних організацій включають три групи видатків: витрати на управління, загальногосподарські витрати, невиробничі витрати. Вони розраховуються за встановленими відсотками до витрат на оплату праці. Середньостатистичний відсоток накладних витрат приймемо 150% від заробітної плати. Розрахунки проведені у формулі (4.5).

$$H_B = 1{,}5 \cdot 1690 = 2536\,(\text{грн}). \qquad (4.5)$$

4.1.6 Інші витрати

Інші витрати є витратами, які не враховані в попередніх статтях. Вони становлять 10% від заробітної плати. Розрахунки проведено у формулі (4.6).

$$I_B = 1690 \cdot 0{,}1 = 169 \text{ (грн).} \quad (4.6)$$

Витрати на розробку програмного забезпечення обраховуються за формулою (4.7). Проведені розрахунки зображено у формулі (4.8).

$$K_1 = B_{ОП} + B_Ф + H_B + I_B \quad (4.7)$$

$$K_1 = 1690 + 346{,}45 + 204{,}5 + 2535 + 169 = 4944{,}95 \text{ (грн).} \quad (4.8)$$

Витрати на відлагодження і дослідну експлуатацію програмного продукту визначаємо за формулою (4.9).

$$K_2 = S_{м.г.} \cdot t_{від}, \quad (4.9)$$

де $S_{м.г.}$ – вартість однієї машино-години роботи ПК, грн/год.

$t_{від}$ – комп'ютерний час, витрачений на відлагодження і дослідну експлуатацію створеного програмного продукту, год.

Загальна кількість днів роботи на комп'ютері дорівнює 40 днів. Середній щоденний час роботи на комп'ютері – 2 години. Вартість години роботи комп'ютера дорівнює 5,2 грн. Обрахунок витрав на від лагодження зображено у формулі (4.10).

$$K_2 = 5{,}2 \cdot 100 = 520 \text{ (грн).} \quad (4.10)$$

На основі отриманих даних складаємо кошторис витрат на розробку

програмного забезпечення (таблиця 2.5).

Таблиця 4.5 – Кошторис витрат на розробку програмного забезпечення

Найменування витрат	Сума витрат, грн
Витрати на оплату праці	1690
Відрахування у спеціальні державні фонди	346,45
Витрати на куповані вироби	204,5
Накладні витрати	2535
Інші витрати	169
Витрати на відлагодження і дослідну експлуатацію програмного продукту	520
Разом	5464,95

4.2 Визначення експлуатаційних витрат

Для оцінки економічної ефективності розроблюваного програмного продукту слід порівняти його з аналогом, тобто існуючим програмним забезпеченням ідентичного функціонального призначення.

Експлуатаційні одноразові витрати по програмному забезпеченню і аналогу включають вартість підготовки даних і вартість роботи комп'ютера (за час дії програми). Розрахунки проводяться за формулою (4.11).

$$E_{П} = E_{1П} + E_{2П}, \tag{4.11}$$

де $E_{П}$ – одноразові експлуатаційні витрати на ПЗ (аналог), грн.;

$E_{1П}$ – вартість підготовки даних для експлуатації ПЗ (аналогу), грн;

$E_{2П}$ – вартість роботи комп'ютера для розробки програмного забезпечення (аналогу), грн.

Річні експлуатаційні витрати $B_{ЕП}$ визначаються за формулою (4.12).

$$B_{ЕП} = E_{П} \cdot N_{П}, \tag{4.12}$$

де $N_{П}$ – періодичність експлуатації ПЗ (аналогу), раз/рік.

Вартість підготовки даних для роботи на комп'ютері визначається за формулою (4.13).

$$E_{1П} = \sum_{i=1}^{N} n_i \cdot t_i \cdot c_i, \tag{4.13}$$

де i – категорії працівників, які приймають участь у підготовці даних (i=1,2,...n);

n_i – кількість працівників i-ої категорії, осіб;

t_i – трудомісткість роботи співробітників i-ої категорії по підготовці даних, год.;

c_i – середнього годинна ставка працівника і-ої категорії з врахуванням додаткової заробітної плати, що знаходиться із співвідношення, зображеного на формулі (4.14).

$$c_i = \frac{c_i^0 (1+b)}{m}, \tag{4.14}$$

де c_i^0 – основна місячна заробітна плата працівника і-ої категорії, грн;

b – коефіцієнт, який враховує додаткову заробітну плату;

m – кількість робочих годин у місяці, год.

Для роботи з даними як для поточного програмного забезпечення так і аналогу потрібен один працівник, основна місячна заробітна плата якого складає: c^0 = 1400 грн. Середньо годинна ставка розрахована у формулі (4.15).

$$c_1 = \frac{1400}{22*8} = 12,5 \text{ (грн/год)} \quad (4.15)$$

Трудомісткість підготовки даних для програмного забезпечення складає 1 год., для аналога 1,5 год.

Таблиця 4.6 – Розрахунок витрат на підготовку даних та реалізацію програмного забезпечення на комп'ютері

Час роботи співробітників, год	Середньогодинна заробітна плата, грн/год	Витрати, грн
Проектне рішення		
1	12,5	12,5
Аналог		
1,5	12,5	18,75

Витрати на експлуатацію комп'ютера визначаються за формулою (4.16).

$$E_{2П} = t \cdot S_{МГ}, \quad (4.16)$$

де t – витрати машинного часу для реалізації програмного продукту, год;

$S_{МГ}$ – вартість однієї години роботи комп'ютера, грн/год.

$$E_{2П} = 1 \cdot 5,2 = 5,2 \text{ (грн)}; \; E_{2А} = 1,5 \cdot 5,2 = 7,8 \text{ (грн)}.$$

$$E_{П} = 12,5 + 5,2 = 17,7 \text{ (грн)}; \; E_{А} = 18,75 + 7,8 = 26,55 \text{ (грн)}.$$

$$B_{ВП} = 17,7 \cdot 252 = 4460,4 \text{ (грн)}; \; B_{ВА} = 26,55 * 252 = 6690,6 \text{ (грн)}.$$

4.3 Розрахунок ціни споживання програмного продукту

Ціна споживання – це витрати на придбання і експлуатацію програмного продукту за весь строк його служби. Їх розрахунок проведено у формулі (4.17).

$$Ц_{С(П)} = Ц_П + B_{(E)NPV}, \quad (4.17)$$

де $Ц_П$ – ціна придбання програмного продукту, грн., яка розраховується за формулою (4.18).

$$Ц_П = К(1 + \frac{П_р}{100}) + К_0 + К_к, \quad (4.18)$$

де $К$ – кошторисна вартість;

$П_р$ – рентабельність;

$К_0$ – витрати на прив'язку та освоєння програного забезпечення на конкретному об'єкті, грн;

$К_к$ – витрати на доукомплектування технічних засобів на об'єкті, грн.

$$Ц_Д = 5464,95 \cdot (1 + 0,3) = 7104,4 \text{ (грн)}.$$

Вартість витрат на експлуатацію програмного забезпечення (за весь час його експлуатації) розраховуються за формулою (4.19) Вартість витрат на експлуатацію програмного забезпечення для розробки та аналога зображені на рівняннях (4.20)-(4.21).

$$B_{(E)NPV} = \sum_{t=0}^{T} \frac{B_{вп}}{(1+R)^t}, \quad (4.19)$$

де $B_{вп}$ – річні експлуатаційні витрати, грн;

T – термін служби програмного забезпечення, років;

R – річна ставка проценту банку.

$$B_{(E)NPV} = \sum_{t=1}^{5} \frac{4460,4}{(1+0,08)^t} = 17845,3 (\text{грн}). \quad (4.20)$$

$$B_{(E)NPV} = \sum_{t=1}^{5} \frac{6690,6}{(1+0,08)^t} = 26767,9 \ (\text{грн}). \quad (4.21)$$

Тоді ціна споживання програмного забезпечення дорівнюватиме обрахункам, які зображені у формулі (4.22).

$$Ц_{СП} = 71404,4 + 17845,3 = 24949,7 \ \underline{(\text{грн}).} \quad (4.22)$$

Аналогічно визначається ціна споживання для аналогу. Обрахунки наведені у формулі (4.23).

$$Ц_{СА} = 6000,0 + 26767,9 = 32767,9 \ (\text{грн}). \quad (4.23)$$

4.4 Визначення показників економічної ефективності

Економічний ефект в сфері розробки програмного продукту обраховується за формулою (4.24). Проведені розрахунки зображені у формулі (4.25).

$$E_{ПР} = Ц_{СП} - Ц_{СА} \quad (4.24)$$

$$E_{ПР} = 7104,4 - 6000,0 = 1404,4 \text{ (грн).} \quad (4.25)$$

Річний економічний ефект в сфері експлуатації обраховується за формулою (4.26). Розрахований економічний ефект зображений у формулі (4.27).

$$E_{КС} = B_{ЕА} - B_{ЕП} \quad (4.26)$$

$$E_{КС} = 6690,6 - 4460,4 = 2230,2 \text{ (грн).} \quad (4.27)$$

Додатковий економічний ефект у сфері експлуатації обраховується за формулою (4.28). Проведені обрахунки із підставленими значеннями зображені у формулі (4.29).

$$\text{V}E_{екс} = \sum_{t=1}^{T} E_{екс}(1+R)^{T-t} \quad (4.28)$$

$$\text{V}E_{екс} = \sum_{t=1}^{5} 2230,2 \cdot (1+0,08)^{5-t} = 13068,85 \text{ (грн).} \quad (4.29)$$

Сумарний ефект зображений у формулі (4.30).

$$E = E_{ПР} + \text{V}E_{екс} = 1104,4 + 13068,85 = 14173,25 \text{ (грн).} \quad (4.30)$$

Таблиця 4.7 – Показники економічної ефективності програмного забезпечення

Найменування	Одиниці вимірювання	Значення показників	
		Базовий варіант	Новий варіант
1	2	3	4
Капітальні вкладення	грн.	–	5464,95
Ціна придбання	грн.	6000,0	7104,4

Продовження таблиці 4.7.

1	2	3	4
Річні експлуатаційні витрати	грн.	17845,3	26767,9
Ціна споживання	грн.	24949,7	32767,9
Економічний ефект в сфері проектування	грн.	–	1104,4
Економічний ефект в сфері експлуатації	грн.	–	2230,2
Додатковий ефект в сфері експлуатації	грн.	–	13068,85
Сумарний ефект	грн.	14173,25	

Отже, В даному розділі проведено розрахунок витрат на розробку програмного забезпечення. Здійснено порівняння з існуючим аналогом, і цим показано, що вказане програмне забезпечення має переваги в порівнянні з аналогами, зокрема: надійність, простота використання, гнучкість, зручність.

ВИСНОВКИ

1. Проведено аналіз технологій розробки веб-сторінок, що дозволило виділити основні інструменти для роботи.

2. Проведено аналіз систем керування вмістом, та визначено оптимальний вибір – Textolite cms.

3. Розроблено структуру веб-сторінки, виділено основні блоки сторінки та описано їх роботу.

4. Створено адаптивний дизайн сторінки, який створює приємне відображення на усіх видах розширень.

5. Розроблено верстку сайту за допомогою фремворку bootstrap, що дозволило із легкістю реалізувати адаптивність.

6. Досліджено способи протистояння спам-ботам, та реалізовано захист контактної форми.

7. Досліджено оптимізацію пошукових систем, та реалізовано підняття рейтингу SEO.

8. Проведено тестування та верифікацію розробленої сторінки.

Згідно проведеного економічного обґрунтування зазначене програмне забезпечення є конкурентноздатним. Крім того, отримано економічний ефект у розмірі 14173,25 грн. і тому розробка і впровадження цього програмного забезпечення є економічно доцільними.

Результати бакалаврської роботи доповідалися на інтернет-конференції «Науково-практична конференція інтелектуальні комп'ютерні системи та мережі» (додаток Б).

Розроблена веб-сторінка має практичне значення, що підтверджено довідкою про використання (додаток В).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. How Web Pages Work: веб-сайт. URL: https://tinyurl.com/y67nzh2x (дата звернення: 12.02.19)
2. Лоусон Б., Шарп Р. Изучаем HTML 5. Санкт-Петербург, 2011. 253 с.
3. What is HTML? The Basics of Hypertext Markup Language Explained: веб-сайт. URL: https://tinyurl.com/y2z7aocw (дата звернення 12.02.19).
4. Руководства по веб-технологиям: веб-сайт. URL: https://webref.ru (дата звернення 12.02.19)
5. Макфарланд Д. Большая книга CSS3. Санкт-Петербург, 2014. 608 с.
6. Lindley C. JavaScript Enlightenment. Boston, 2012. 166 с.
7. The Process of Making a Website: веб-сайт. URL: https://tinyurl.com/y5qffton (дата звернення 12.02.19).
8. How do Web Servers work? Веб-сайт. URL: https://tinyurl.com/y4m2jp3u (дата звернення 12.02.19).
9. Barker D. Web Content Management. Boston, 2016. 350 с.
10. How to choose the right CMS for your website - Open Source CMS Comparison: веб-сайт. URL: https://tinyurl.com/y2unsaln (дата звернення 26.02.19).
11. Textolite. Система управления статическим сайтом: веб-сайт. URL: https://textolite.ru/ (дата звернення 26.02.19).
12. How a webpage is loaded and displayed: веб-сайт. URL: https://varvy.com/pagespeed/display.html (дата звернення 26.02.19).
13. What is database?: веб-сайт. URL: https://www.lifewire.com/what-is-a-database-1019737 (дата звернення 26.02.19).
14. Дейт К. Введение в системы баз данных. Москва, 2005. 1316 с.
15. Триггеры в MySQL / Хабр: веб-сайт. URL: https://tinyurl.com/y58swh4b (дата звернення 15.05.19).
16. 7 Tips for Designing Landing Pages that Convert: веб-сайт. URL:

https://tinyurl.com/yxzmn3x7 (дата звернення 15.05.19).

17. Bootstrap - The popular HTML, CSS, and JS library in the world: веб-сайт. URL: https://getbootstrap.com/ (дата звернення 15.05.19).

18. Navigation scrollspy & smooth scroll plugin - nav-scroll-spy.js: веб-сайт. URL: https://tinyurl.com/y4t7mkda (дата звернення 15.05.19).

19. HTML5 Form Validation Examples: веб-сайт. URL: https://tinyurl.com/y4l49vzz (дата звернення 15.05.19).

20. How to Build SEO-Friendly Landing Pages: веб-сайт. URL: https://tinyurl.com/y26bqnar (дата звернення 15.05.19).

21. 5 Reasons To Use A Content Management System: веб-сайт. URL: https://tinyurl.com/y3wskx6g (дата звернення 15.05.19)

22. 10 Ways to Make Your Website Load Faster: веб-сайт. URL: https://tinyurl.com/y38klypr (дата звернення 15.05.19)

23. How to Measure Your Website Speed and Load Times: веб-сайт. URL: https://tinyurl.com/y4bglbwa (дата звернення 15.05.19)

24. Paletton – The Color Scheme Designer: веб-сайт. URL: https://tinyurl.com/y3n8g2c9 (дата звернення 15.05.19)

25. Relational Databases. Databases. ICT: веб-сайт. URL: https://tinyurl.com/y5djg5vg (дата звернення 15.05.19)

26. 6 Phases Of The Website Design: веб-сайт. URL: https://tinyurl.com/y69zvm2h (дата звернення 15.05.19)

27. Critical rendering path – An in-depth guide: веб-сайт. URL: https://tinyurl.com/nn3vz78 (дата звернення 15.05.19)

28. The Process of Designing and Building a Website: веб-сайт. URL: https://tinyurl.com/y5qffton (дата звернення 15.05.19)

29. Adding Meta Yags Yo Your Website: веб-сайт. URL: https://tinyurl.com/yynqwyyl (дата звернення 15.05.19)

30. Top 8 Tips for Designing an Effective Landing Page - Prototypr: веб-сайт. URL: https://tinyurl.com/yyz75q6a (дата звернення 15.05.19)

31. 2019`s Best Free Web Hosting: веб-сайт. URL: https://www.000webhost.com/ (дата звернення 15.05.19)

32. 11 Tips For Designing High-Converting Landing Pages: веб-сайт. URL: https://tinyurl.com/y489ebyq (дата звернення 15.05.19)

33. Методичні вказівки до оформлення курсових проектів, звітів про проходження практики, випускних кваліфікаційних робіт для студентів спеціальності "Комп'ютерна інженерія" / І.В.Гураль, Л.О.Дубчак / Під ред. О.М.Березького. - Тернопіль: ТНЕУ, 2019. - 33с.

34. Методичні рекомендації до виконання дипломного проекту з освітньо-кваліфікаційного рівня "Бакалавр" напряму підготовки 6.050102 "Комп'ютерна інженерія" фахового спрямування "Комп'ютерні системи та мережі" / О.М.Березький, Л.О.Дубчак, Г.М.Мельник, Ю.М.Батько, С.В.Івасьєв / Під ред. О.М.Березького. - Тернопіль ТНЕУ, 2016, - 60с.

35. Методичні вказівки до написання техніко-економічного розділу дипломних проектів освітньо-кваліфікаційного рівня «бакалавр» підготовки 6.050102 комп’ютерна інженерія/ І.Р. Паздрій – Тернопіль: ТАНГ, 2014.– 37 с.