Київський національний торговельно-економічний університет

Кафедра комп'ютерних наук та інформаційних систем

ВИПУСКНИЙ КВАЛІФІКАЦІЙНИЙ ПРОЕКТ

на тему:

«Проектування та створення web-орієнтованої інформаційної системи для закладу громадського харчування»

студента 4 курсу, 9 групи,
спеціальності
122 «Комп'ютерні науки» _____ *підпис студента* Іштукіна Владислава Валерійовича

Науковий керівник
кандидат фізико-математичних наук, доцент _____ *підпис керівника* Самойленко Ганна Тимофіївна

Гарант освітньої програми
кандидат технічних наук, доцент _____ *підпис керівника* Демідов Павло Георгійович

Київ 2020

Київський національний торговельно-економічний університет

Факультет інформаційних технологій
Кафедра комп'ютерних наук та інформаційних систем
Спеціальність 122 «Комп'ютерні науки»

Затверджую
Зав. кафедри _____Пурський О.І.
«20» грудня 2019р.

Завдання
на випускний кваліфікаційний проект студенту

Іштукіну Владиславу Валерійовичу
(прізвище, ім'я, по батькові)

1. Тема випускного кваліфікаційного проекту.
«Проектування та створення web-орієнтованої інформаційної системи для закладу громадського харчування»
Затверджена наказом ректора від *«04» грудня 2019 р. № 4111*

2. Строк здачі студентом закінченого проекту 29 червня 2020 року

3. Цільова установка та вихідні дані до проекту

Мета проекту: *обґрунтування та розробка веб-сайту, з урахуванням сучасних світових тенденцій побудови організаційних та функціональних інформаційних структур підприємств.*

Об'єкт дослідження: процес проектування веб-сайту закладу громадського харчування.

Предмет дослідження: засоби створення веб-сайту закладу громадського харчування.

4. Перелік графічного матеріалу_____
_____
_____
_____

5. Консультанти по проекту із зазначенням розділів, за якими здійснюється консультування:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Самойленко Г.Т.	15.12.2019 р.	15.12.2019 р.
2	Самойленко Г.Т.	15.12.2019 р.	15.12.2019 р.
3	Самойленко Г.Т.	15.12.2019 р.	15.12.2019 р.

6. Зміст випускного кваліфікаційного проекту (перелік питань за кожним розділом)

ВСТУП

РОЗДІЛ 1. ПРОЕКТУВАННЯ WEB-ОРІЄНТОВАНОЇ СИСТЕМИ ЗАКЛАДУ ГРОМАДСЬКОГО ХАРЧУВАННЯ

1.1. Функціональні вимоги до web-орієнтованої системи

1.2. Огляд існуючих засобів проектування

1.3. Огляд API-технологій

РОЗДІЛ 2. ОРГАНІЗАЦІЯ РОЗРОБКИ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Моделювання інформаційного забезпечення

2.2. Висновки до розділу

РОЗДІЛ 3. РОЗРОБКА ВЕБ-ОРІЄНТОВАНОЇ СИСТЕМИ ЗАКЛАДУ ГРОМАДСЬКОГО ХАРЧУВАННЯ

3.1. Налаштування серверу

3.2. Розробка та верстка сайту

3.3. Робота з API технологіями

3.4. Розробка мобільної версії сайту

ВИСНОВКИ ТА РЕЗУЛЬТАТИ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

7. Календарний план виконання проекту

№ Пор.	Назва етапів випускного кваліфікаційного проекту	Строк виконання етапів проект	
1	2	3	4
1	*Вибір теми випускного кваліфікаційного проекту*	*01.10.2019*	*01.10.2019*
2	*Розробка та затвердження завдання на випускний кваліфікаційний проект*	*15.12.2019*	*15.12.2019*
3	*Вступ*	*03.02.2020*	
4	*РОЗДІЛ 1. Проектування веб-орієнтованої системи закладу громадського харчування.*	*28.02.2020*	
5	*РОЗДІЛ 2. Організація розробки інформаційної системи.*	*06.04.2020*	
6	*РОЗДІЛ 3. Розробка веб-орієнтованої системи закладу громадського харчування.*	*12.05.2020*	
7	*Висновки*	*15.05.2020*	
8	*Здача випускного кваліфікаційного проекту на кафедру науковому керівнику*	*20.05.2020*	
9	*Попередній захист випускного кваліфікаційного проекту*	*03.06.2020*	
11	*Виправлення зауважень, зовнішнє рецензування випускного кваліфікаційного проекту*	*09.06.2020*	
12	*Представлення готової зшитої випускного кваліфікаційного проекту на кафедру*	*12.06.2020*	
13	*Публічний захист випускного кваліфікаційного проекту*	*За розкладом роботи ЕК*	

8. Дата видачі завдання *«15» грудня 2019 р.*

9. Керівник випускного кваліфікаційного проекту
Самойленко Г.Т.
(прізвище, ініціали, підпис)

10. Гарант освітньої програми Демідов П.Г.
(прізвище, ініціали, підпис)

11. Завдання прийняв до виконання студент-дипломник
Іштукін В.В._____
(прізвище, ініціали, підпис)

12. Відгук керівника випускного кваліфікаційного проекту

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

Керівник випускного кваліфікаційного проекту

_____ _____

(підпис, дата)

13. Висновок про випускний кваліфікаційний проект

Випускний кваліфікаційний проект студента _____

(прізвище, ініціали)

може бути допущена до захисту в екзаменаційній комісії.

Гарант освітньої програми _____Демідов П.Г.

(підпис, прізвище, ініціали)

Завідувач кафедри _____Пурський О.І.

(підпис, прізвище, ініціали)

«_____»_____2020 р.

АНОТАЦІЯ

У випускному кваліфікаційному проекті здійснено проектування та створенняweb-орієнтованої інформаційної системи для закладу громадського харчування з метою подальшого збору коштів та постійного інвестування проекту розробки продукту харчування. В ході виконання роботи було розглянуто питання основних понять проектування макетів та скетчів сайту, проведено аналіз процесу розробки повноцінного web-ресурсу, підходи до впровадження новітніх APIтехнологій з метою візуалізацій процесу збору коштів, їх переваги та недоліки, проведено аналіз різних інструментів для створення необхідного web-ресурсу та обраний найбільш оптимальний для виконання роботи. В практичній частині було створено web-орієнтовану інформаційну систему для закладу громадського харчування.

Ключові слова: проектування web-ресурсу, web-орієнтована інформаційна система, сайт, збір коштів.

ABSTRACT

In the final qualification project, the design and creation of the web-oriented information system for social catering establishment in order to raise funds and constant investing of the creation a new food product project . During the work the issues of basic concepts of layout design and site sketch were considered, the analysis of the process of developing a full-fledged web-resource, approaches to the introduction of the latest API technologies to visualize the fundraising process, making the analysis of various tools for creating appropriate web-resource. In the practical part, a web-oriented information system for a catering establishment was created.

Keywords: web-resource design, web-oriented information system, site, fundraising.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

API— Application Programming Interface

JSON— Текстовий формат обміну даними

SOAP— Simple Object Access Protocol

REST — Representational State Transfer

HTTP— Hyper Text Transfer Protocol

HTTPS— Hyper Text Transfer Protocol Secure

HTML — Hyper Text Markup Language.

CSS — Cascading Style Sheets

JavaScript— динамічна, об'єктно-орієнтована прототипна мова програмування.

PHP— скриптова мова програмування, для генерації HTML-сторінок на стороні веб-сервера.

JQuery— Бібліотека JavaScript, що базується на взаємодії JavaScript з HTML.

XML — eXtensible Markup Language.

URL — Uniform Resource Locator.

CMS — Content Management System.

UTF-8— Unicode Transformation Format, 8-bit

FTP —File Transfer Protocol

ІС — Інформацйна система

ЗМІСТ

ВСТУП

Актуальність дослідження полягає в тому, що зростання можливостей бізнесу в мережі Інтернет та розвиток глобальної інформаційної інфраструктури створюють принципово нову ситуацію в економіці. Тому надзвичайно важливо створювати якісні, прості в застосуванні та головне унікальні інформаційні системи, котрі можуть допомагати малому та великому бізнесу у впровадженні web-технологій до їх підприємств.

У даний час Інтернет - це один з найрозвиненіших і найефективніших платформ для ведення бізнесу. При ефективному використанні усіх можливостей рівень доходу як і рівень комфорту клієнтів значно зростає, що змушує відмітити, що компаніям необхідне власне представництво в мережі Інтернет.

Першочерговим завданням для підприємства сьогодні є вихід на інтернет-ринок, що дозволяє залучити нових клієнтів та не втратити вже існуючих постійних клієнтів, охопити більший відсоток ринку, за рахунок моніторингу популярності товарів, цін, рекламних компаній конкурентів та фінансових уподобань інвесторів. Інтенсивний розвиток можливостей глобальної мережі і використання її як інструменту ведення бізнесу висунуло нові вимоги як до створення web-ресурсів, так і до просування web-сайтів у глобальному пошуку.

Мета проекту: обґрунтування та розробка web-сайту, з урахуванням сучасних світових тенденцій побудови організаційних та функціональних інформаційних структур підприємств.

Об'єкт дослідження: процес проектування web-сайту закладу громадського харчування.

Предмет дослідження: засоби створення web-сайту закладу громадського харчування.

Методи дослідження: аналіз методів створення web-ресурсів, використання принципів інформаційного моделювання, застосування системного підходу та структурно-функціонального методу.

Основні завдання дослідження:

1. Аналіз роботи та використання API-технологій.
2. Огляд інструментальних засобів для використання технологій взаємодії модулів.
3. Розробка макетів створюваного сайту.
4. Розробка web-сайту з використанням API-технологій.

Наукова новизна дослідження:

1. Проведено аналіз та дана характеристика API-технології як інструмента при створенні web-ресурсу.
2. Проведено аналіз web-орієнтованих систем закладів громадського харчування.
3. Розроблено web-ресурс з поєднанням різних технологій для найкращого представлення та візуалізації даних для звичайного користувача.

Практична значущість проекту:

1. Наведено результати практичного використання API-технологій та їх застосування при створенні web-сторінки для популяризації та збору коштів для компанії "LLC InbicoEurope".
2. Дано рекомендації по застосуванню різних технологій при створенні одно сторінкового web-ресурсу для представлення продукту.

РОЗДІЛ 1. ПРОЕКТУВАННЯ WEB-ОРІЄНТОВАНОЇ СИСТЕМИ ЗАКЛАДУ ГРОМАДСЬКОГО ХАРЧУВАННЯ

1.1. Функціональні вимоги до web-орієнтованої системи

Web являє собою результат застосування можливостей доступу до інформації, що може бути розподілена по всьому світу для створення мультимедіа інформаційно-пошукових та глобальних гіпертекстових систем. Можливості доступу до територіально-розподіленої інформації забезпечує для Web всесвітня мережа Internet. Web-система в основному розвивається як сховище різнопланової, неузгодженої і часто слобостуктурованої інформації і значно відрізняється від баз даних, де наявна інформація гарно структурована та взаємозалежна.

Web являє собою мережу вузлів, що містять гіпермедіа-документи й зв'язки, що дозволяють із одного документа посилатися на інші, розміщені як на тому ж вузлі, так і на інші.

Із самого моменту свого народження Web була визначена як технологія-посередник для зв'язування різних типів інформаційних ресурсів. При цьому HTML-сторінки відігравали роль цементу всієї цієї інформаційної конструкції [1]. Це давало можливість швидко нарощувати інформаційну ємність за рахунок конвертації інформаційних масивів у формат Web або їхнього підключення серверам Web через програми-шлюзи.

Сама технологія була побудована за схемою «клієнт-сервер», неорієнтованої на постійне з'єднання.

Технологія клієнт-сервер є реалізацією розподіленої обробки даних. У системі архітектури клієнт-сервер обробка даних розділена між комп'ютером-клієнтом і комп'ютером-сервером, зв'язок між якими відбувається по мережі. Це поділ процесів обробки даних засновано на групуванні функцій. Як правило, комп'ютер-сервер баз даних виділяється для виконання операцій з базами даних, а комп'ютер-клієнт виконує прикладні програми.

Основна функція комп'ютера-клієнта складається у виконанні додатка (інтерфейсу з користувачем і логіки подання) і здійсненні зв'язку із сервером, коли цього вимагає додаток. Комп'ютер-клієнт може бути як простою машиною типу персонального комп'ютера (ПК) із процесором 286 і операційною системою DOS, так і потужною робочою станцією із багатозадачною і багатокористувацькою операційною системою.

Таким чином, вибір комп'ютера, операційної системи, оперативної й дискової пам'яті, іншого устаткування визначається вимогами додатка. Як програма-клієнт звичайно виступає браузер.

Як слідує вже із самого терміну, головна функція комп'ютера-сервера полягає в обслуговуванні потреб клієнта. Одна з важливих вимог до сервера - операційна система, у середовищі якої розміщений сервер, повинна бути багатозадачною. Сервером, як правило, виступає програма-сервер протоколу обміну гіпертекстовою інформацією HTTP, що відповідає на запити клієнтів [2].

Браузер надає користувачеві можливість указувати зовнішні програми-інтерпретатори для різних типів документів.

Чудесною знахідкою, що дозволила відкрити великій кількості людей доступ до Інтернет, була концепція гіпертексту, запропонована Теодором Хольмом Нельсоном. Саме Нельсон вважається батьком ідеї гіпертексту в тім виді, у якому він зараз існує.

Гіпертекст - це звичайний текст, що містить посилання як на власні фрагменти, так і на інші тексти. Розповідаючи про те, що послужило прообразом для цього винаходу, Нельсон згадує уривок з одного нарису Ванневара Буша, написаного в 1945 році: «Робота людської думки побудована на принципі асоціацій. Аналізуючи яке-небудь поняття або елемент, вона неодмінно прагне поставити йому у відповідність який-небудь інший знайомий образ, що підказує асоціацією думок, і ця відповідність установлюється завдяки важковловимій павутині зв'язків, сформованих клітками людського мозку». Спроектував цю ідею про роботу мозку однієї людини на комп'ютерну мережу,

що охоплює увесь світ, Нельсон посіяв насіння явища, що згодом переросло в «Всесвітню Павутину».

Збереження ж і власне обробка інформації в Інтернеті відбувається за допомогою так званих "web-орієнтованих" інформаційних систем, які можуть і використовуватися в локальній мережі. Web-орієнтовані ІС побудовані з використанням web-додатків(Web-Application) – це допоміжні програмні засоби, що призначені для автоматизованого виконання будь-яких дій як на web-серверах, так і на стороні користувача[3]. При цьому в якості користувацьких інтерфейсів web-додатки використовують web-браузери. До числа засобів створення web-додатків відносяться клієнтські і серверні технології.

Процес створення web-орієнтованих інформаційних систем нічим не відрізняється від написання програм. Стадії визначення вимог, аналізу, реалізації, проектування, тестування цей процес теж проходить. При роботі веб - сайту, не важливо з використанням яких технологій він написаний, користувач так само які у звичайній програмі, отримує інформацію, працює з вікнами і меню, користується гіперпосиланнями, зберігає дані на сервері. Але є одна різниця, яка полягає в тому, що саме програмне забезпечення працює не на комп'ютері користувача, а на віддаленому сервері мережі, а доступ до даних можна отримати із любої точки світу де є кабельні мережі чи телефонний зв'язок. З однієї сторони це зручно, а з іншої з'являються вимоги до програмного забезпечення, яке створюється[4].

Особливості web-орієнтованих інформаційних систем:

1) надійність;
2) багатокористувацька робота;
3) проблема швидкодії;
4) незалежність від операційної системи клієнта;
5) знаходження на одному місці;
6) для користувача не потрібна ніяка програма;
7) користувач не являється адміністратором;

8) в ролі адміністратора розробник системи;

9) від користувача нічого не потрібно;

10) малий розмір;

11) переносимість;

12) простота;

13) архітектура web-додатків не видна для користувача.

Надійність таких систем полягає у тому, що вони повинні працювати без збоїв, а не просто працювати. Неможна навантажувати систему занадто великим потоком користувачів і оскільки всі користувачі працюють віддалено в різних містах або навіть в різних континентах, коли до системи є доступ через Інтернет, необхідно забезпечувати можливість перегляду web-ресурсу з будь-якої точки світу в будь-який момент часу[5].

Тут головною проблемою виступає перенавантаження сервера. Вона залежить він ряду факторів таких як: пропускна здатність системи, кількість користувачів, якість Інтернет зв'язку серверу та web-ресурсу, обмеження базового аналогового фізичного середовища, доступна потужність обробки компонентів системи та поведінка кінцевого користувача.

В успішних web-проектах на сайт приходять десятки користувачів на секунду. Тому варто замислитися над оптимізацією швидкості виконання web-ресурсу.

Проблема швидкодії вирішується просто в локальних мережах. Тут ті, хто розробляють програмне забезпечення не хвилюються, яку кількість даних передає і приймає їх програма. Коли пропускна спроможність лінії замала для невеликих систем, з яким працюють декілька десятків користувачів, тоді прокладаються оптоволоконні кабелі та встановлюється високошвидкісна мережа. Однак, коли в локальній мережі з цим ще можна примиритись так як об'єми даних, що передаються компенсуються швидкодією мережі, яка збільшується, то створеному по такому принципу web-додатку працювати буде неможливо[6]. Хоча швидкість модемів постійно зростає, і багато користувачів працюють з виділеною лінією Інтернет, проте тут виникає питання оплати

трафіка, при якому кожний зайвий мегабайт даних буде оплачуватися з кишені користувача.

Незалежність від операційної системи клієнта (кросплатформеність) означає, що web-орієнтована ІС повинна працювати на всіх операційних системах.

Поведінка web-орієнтованої ІС, як і будь-якого об'єкта в цьому світі, залежить від зовнішніх і внутрішніх умов.

До внутрішніх умов функціонування можна віднести такі фактори:

- налаштування сервера;
- тип сервера;
- тип бази даних;
- зміст перемінних середовища;
- зміст інформації на жорсткому диску сервера;
- зміст самої бази даних.

Також особливістю web-орієнтованих ІС є знаходження всієї програмної логіки на сервері в порівнянні з простим ПЗ, де логіка знаходиться на комп'ютері кожного користувача. Так як є одна тільки логіка програмного забезпечення її набагато простіше розповсюджувати серед користувачів. Про старий спосіб розповсюдження програми можна забути. По суті проблеми розповсюдження web-орієнтованої ІС не існує, адже доступ до неї можна одержати в любий момент в любому місці, коли вона доступна через всесвітню мережу Інтернет[7].

Користувачу не потрібна ніяка програма. Все що йому потрібно, це просто запустити браузер і набрати в адресному рядку URL. В наші дні браузер є стандартною програмою, яку користувач отримує при встановленні ОС автоматично. Шукати браузер йому не потрібно, адже він уже є встановлений на його комп'ютері, і по суті, це все, що йому потрібно для його роботи[8].

Користувач не є адміністратором даного web-ресурсу. Як правило, коли користувач встановлює на своїй машині додаток, то йому приходиться брати на себе роль адміністратора цього додатку. Йому потрібно встановлювати його,

запускати, налаштовувати, вирішувати виникаючі проблеми. У випадку з web-додатком, так як воно знаходиться на web-сервері, користувачу не потрібно турбуватися про за це. А це – в свою чергу є ознакою хорошого web-додатку. У випадку з простим додатком про таке і мріяти не доводиться.

В ролі адміністратора виступає розробник ІС. Так, це ще один вантаж на плечі програміста. Проте якщо порівнювати вартість створення web-додатку з вартістю утримання команди спеціалістів, які займаються встановленням, підтримкою простих додатків на машинах користувачів, то зразу можна побачити, що це буде дешевше, не говорячи уже про ефективність. З економічної точки зору, набагато вигідніше утримувати невелику команду програмістів, які працюють в одному місці над одним додатком[9].

Web-орієнтовані інформаційні системи не потребують нічого від користувача. Web-додатки, з яких побудована інформаційна система не висувають ніяких вимог до апаратної платформи.

Такої проблеми як підтримки різних версій в минулому тепер не існує. Як тільки виходить нова версія web-додатку, то всі без виключень користувачі отримають її негайно. Так існує тільки одна копія додатку (додатків) на білому світі, то всі старі версії негайно зникають, а користувач навіть не помічає, що у нього нова версія програми [9]. Це також означає, що розробникам не потрібно турбуватися про підтримку старих версій програм і про підтримку зворотної сумісності.

Як показує практика, web-рішення все частіше інтегруються в інформаційну інфраструктуру підприємства, стають його невід'ємною частиною. Принципи швидкого доступу до Інтернету, які добре зарекомендували себе в Інтернеті прекрасно працюють. Швидка публікація інформації на внутрішньому сайті компанії і здобуття інформації з внутрішньої бази даних, доступ до всіх ресурсів за допомогою звичайного web-браузера, легке нарощування можливостей — все це робить web-ресурси чудовим інструментом для роботи з інформацією[9].

Web-системи – гнучкі, адже кожна сторінка, яка передається клієнту, динамічно створюється на сервері у відповідності до конкретного запиту. Що передати, і як це оформити вирішує web-ресурс. Web-системи генерують користувацьких інтерфейс «в польоті». Керівникам IT-служб треба чітко уявляти собі, для чого використовуватиметься система, яку вони розробляють, які проблеми вона покликана вирішувати, а також обґрунтувати її необхідність. Лише при такому підході можна сказати, що web-ресурс буде дійсно робочим інструментом, а не пам'ятником втраченим інвестиціям[10].

1.2. Огляд існуючих засобів проектування

Під час створення web-ресурсу, перш за все необхідно визначитись з інструментами для його розробки. Тому для наявної проблеми можна виділити наступні засоби проектування:

1. **Створення шляхом написання коду вручну**. Цей спосіб є найскладнішим, адже для повного проектування необхідні глибокі знання таких мов як HTML, CSS, JavaScript, та в нашому випадку хоча б ще мову PHP та мінімальної кількості бібліотек до цих мов (до прикладу бібліотека JavaScript - JQuery) [11]. Проте звісно цей спосіб потребує багато часу на розробку та налаштування всіх необхідних моментів.

2.**Створення web-ресурсу за допомогою онлайн конструкторів**. Цей спосіб, на відміну від попереднього, є напевно найпростішим способом створення сайту, зі сторонніми інструментами для редагування та візуально гарного розміщення інформації на сторінці. І хоча цей спосіб є надзвичайно легким в процесі дизайну сайту, проте код сторінки створюється механічно і він є просто нечитабельним для розробників. За необхідності щось змінити вручну та додати якийсь функціонал, будь-який розробник зіштовхнеться з проблемою незрозумілості коду. Тому даний спосіб є найзручнішим для створення дизайну, проте найгіршим для подальшого редагування коду [12].

3.**Наступний спосіб проектування сайту це використання CMS (системи керування вмістом)**. Вона являє собою інформаційну систему або ж

комп’ютерну програму для забезпечення та організації спільного процесу створення, редагування та керування контентом. Цей спосіб є чудово збалансованим варіантом проектування web-ресурсу, котрий поєднує в собі як і зручність та відносну простоту створення та адміністрування сайту, так і можливість ручного редагування коду [13]. Також CMS відомі тим, що це найпростіший спосіб створення онлайн магазинів та сайтів з можливістю авторизації. Системи керування вмістом дозволяють створювати багатосторінкові web-ресурси, робити сторінки взаємозалежними та персоналізованими [14].

4.**Використання технології фреймворк**. Фреймворк - це програмний продукт, який є основою для створення сайтів, але він не має готових рішень для побудови сайтів, не має рішень для виконання певних функцій. Це більш низький рівень ніж CMS. Розробники на фреймворках створюють і інтерфейсну частину, і базу даних, і алгоритми та програмні рішення проблемно орієнтованої частини і скоріше не сайту, а web-додатку[15]. Фреймворк – це структура програмної системи, що полегшує розробку і об'єднання різних компонентів великого програмного проекту. На відміну від бібліотек, які об'єднують набір підпрограм близької функціональності, фреймворк містить в собі велику кількість різних за призначенням бібліотек. Вживається також слово каркас, а деякі автори використовують його в якості основного. Можна також говорити про каркасний підхід як про підхід до побудови, де будь-яка конфігурація будується з двох частин: перша, постійна частина - каркас, незалежний від конфігурації до конфігурації і несе в собі гнізда, в яких розміщується друга, змінна частина - змінні модулі (або точки розширення).

5.**Змішаний спосіб проектування сайту**. Частіше за все під час створення web-ресурсів доводиться поєднувати можливості декількох варіантів розробки сайту для досягнення бажаного результату [16].

1.3. Огляд API-технологій

API або ж прикладний програмний інтерфейс являє собою ніщо інше як набір визначень підпрограм, протоколів взаємодії та засобів для створення програмного забезпечення. Спрощено – це набір чітко визначених методів для взаємодії різних компонентів. API надає розробнику засоби для швидкої розробки програмного забезпечення. API може бути для веб-базованих систем, операційних систем, баз даних, апаратного забезпечення, програмних бібліотек. Розглянемо детальніше API, орієнтоване на роботу з веб-базованими системами [17].

При використанні прикладного програмного інтерфейсу в контексті веб-розробки, як правило, API позначається набором повідомлень-запитів HTTP та структурою повідомлень<>відповідей. Повідомлення можуть мати різний формат, як правило це XML або JSON [18]. Доступ відбувається до з однієї або декількох загальнодоступних кінцевих точок (endpoints).

Кінцеві точки є важливими аспектами взаємодії з веб-інтерфейсами на стороні сервера, оскільки вони вказують, де знаходяться ресурси, доступ до яких може отримати стороння програма. Зазвичай доступ здійснюється через URL, до якого надсилаються HTTP-запити, і звідки очікується відповідь. Кінцеві точки повинні бути статичними, інакше правильне функціонування програмного забезпечення, яке взаємодіє з нею, не може бути гарантоване. Якщо місце розташування ресурсу змінюється (і разом з ним кінцева точка), то раніше написане програмне забезпечення буде перервано, оскільки потрібний ресурс більше не може бути знайдено в одному місці. Оскільки постачальники API все ще хочуть оновлювати свої веб-API, багато хто з них запровадили систему версій в URL, яка вказує на кінцеву точку, до прикладу: кінцева точка для функцій позначення в Web API має такий URL: "*https: //api.google.com/b1/tag/". "/ B1 /*" частина URL визначає доступ до першої версії веб-API [19]. Якщо потрібно оновити цей інструмент до останньої версії, це можна зробити, зберігаючи при цьому підтримку стороннього програмного забезпечення, яке використовує першу версію.

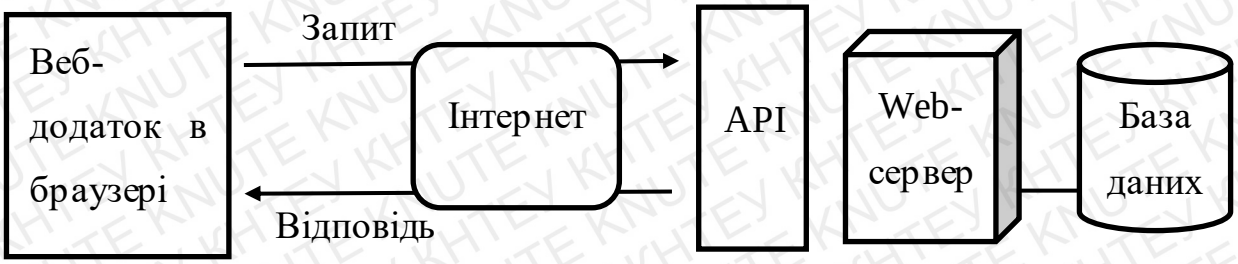

Рис. 1.1. Схема роботи API технологій

Веб-інтерфейси Web 2.0 часто використовують взаємодії на основі таких технологій як REST та SOAP. У той час як прикладний програмний інтерфейс у Web історично був практично синонімом для веб-служби, останнім часом тенденція змінилась (так званий Web 2.0) на відхід від Simple Object Access Protocol (SOAP) на основі веб-сервісів і сервіс-орієнтованої архітектури до більш прямих передач репрезентативного стану (REST) стилів веб-ресурсів та ресурсно-орієнтованої архітектури (ROA). RESTful Веб-інтерфейси зазвичай базуються на основі методів HTTP для доступу до ресурсів за допомогою URL-кодованих параметрів та використання JSON або XML для передачі даних [20].

При використанні деяких Web API, що мають певні обмеження для використання або потребують ідентифікації програмного забезпечення, що викликається, необхідно вказувати API ключ. API ключ– це код, який передається комп'ютерними програмами, викликаючи прикладний програмний інтерфейс (API) на веб-сайті, для ідентифікації викликаючої програми, її розробника або ж її користувача. API ключ використовуються для відстеження та керування використанням API, наприклад, для запобігання зловмисному використанню або зловживання API (як це визначено, можливо, умовами надання послуг). API ключ часто виступає і як унікальний ідентифікатор, так і секретний маркер для аутентифікації, і, як правило, має набір прав доступу до пов'язаного з ним API. API ключі можуть базуватися на універсально унікальному ідентифікаторі (UUID) щоб забезпечити унікальність кожного користувача[21].

Отже, проаналізувавши наявні технології розробки web-орієнтовних інформаційних систем для створення web-ресурсу для збору коштів шляхом краудфандингу та попереднього продажу виготовленої продукції, було обрано використання способу розробки сайтів як ручне написання коду, для повного контролю над процесом створення ресурсу. Для реалізації проекту були обрані мови HTML CSS PHP JavaScript та бібліотека розширення jQuery, а також було вирішено застосовувати в розробці технології взаємодії модулів API.

РОЗДІЛ 2. ОРГАНІЗАЦІЯ РОЗРОБКИ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Моделювання інформаційного забезпечення

2.1.1. Розстановка акцентів та пріоритетів процесу розробки

Перед тим як створювати web-орієнтовану систему закладу громадського харчування, необхідно створити макет системи та перевірити його на адекватність. Після цього необхідно перевірити її відповідність заданим вимогам замовника.

Оскільки сторінка створюється для того, щоб інформувати потенційних клієнтів та перенаправляти їх до іншої сторінки, де відбувається вкладення інвестицій, необхідно визначити, що за платформа для цього використовується. Замовником була обрана платформа з назвою Kickstarter. Kickstarter — це сайт фінансування творчих проектів за схемою краудфандингу, тобто громадське фінансування, де співпраця людей, що добровільно об'єднують свої гроші чи інші ресурси разом, заради підтримки зусиль інших людей або організацій. Kickstarter фінансує різноманітні проекти, у 13-ти категоріях: мистецтво, комікс, танець, дизайн, мода, фільми і відео, їжа, відеоігри, музика, фотографія, видавництво, технологія, театр.

За 3 роки існування більш ніж 1 800 000 людей своїми пожертвами повністю успішно профінансували понад 20 000 проектів, зібравши більше $200 000 000. Тому саме через свою надійність та перевіреність часом, була обрана саме ця платформа для краудфандингу.

Розробку стратегії створення необхідно розпочати з вирішення декількох питань, серед яких основним повинно бути визначення:

1) середовища розробки web-ресурсу;
2) аудиторії, котру повинен приваблювати сайт та зацікавлювати в інвестуванні потенційних клієнтів;
3) дизайну та скетчу сторінки;
   а) розробка макету сайту для десктопної версії;

b) розробка макету для мобільної версії сайту.

Лише після виконання вищевказаних пунктів можна буде просуватись до наступних.

4) підготовка та налаштування серверу;
5) отримання всіх необхідних прав та ключів доступу;
6) типу і структури створюваного сайту.

Перш за все необхідно визначитись з програмним забезпеченням, в середовищі якого буде розроблятись web-ресурс. Під час вибору належного програмного забезпечення потрібне проведення аналізу ефективності кожної з позицій. Тільки в цьому випадку можна очікувати результативності та ефективності роботи самого процесу розробки та виправданості бюджету. З особистого досвіду та гарної обізнаності в програмі "Sublimetext 3" вибір було зроблено в його користь.

Дана програма є кодовим редактором з розширеними можливостями, автозаповненням, підключенням бібліотек та завантаженням коректора будь-якої необхідної для розробки мови програмування.

При визначенні цільової аудиторії сайту, слід врахувати, що загальні запити з'являються в системах пошуку часто, при цьому покупцями стають в основному ті користувачі, які приходять за запитами вужчого і більш конкретного характеру, так як саме цей користувач уже готовий до певного роду інформації, тому головною метою дизайну сайту є виникнення бажання у клієнта залишитись на сторінці та ознайомитись з наведеною інформацією, що в повинно призвести до інвестування або просто подальшого розповсюдження сайту серед власного оточення.

Відповідно до типів цільових аудиторій сформувалися і стратегії, орієнтовані на той дизайн та макет сторінки, який був би новим, простим у використанні та привабливим для користувача.

2.1.2. Розробка макету десктоп-версії web-ресурсу

Наступним етапом буде створення дизайну та макету сайту. Так як створюваний web-ресурс необхідно створювати як односторінковий сайт, з цього випливає ряд важливих моментів, які необхідно враховувати до створення макету.

Оскільки процес створення макету буде поділений на створення десктопної версії сайту та мобільної, необхідно визначити ці поняття.

Десктопна версія – це та версія сайту, яку використовують користувачі, відкриваючи сторінки за допомогою ноутбуків або персональних комп'ютерів [9].

Мобільна ж версія завантажується, коли користувач переходить на сторінку з смартфону, її пропорції та вимоги до створення макету значно відрізняються від попередньої версії, адже розширення екрану стає в декілька разів менше, тому зазвичай ці дві версії розробляються різними, проте в одному дизайні та стилістиці [10].

Перш за все, перший екран або ж верхівка сторінки повинна коротко але змістовно інформувати відвідувача про головне та давати можливість зручно пересуватись сайтом, полегшуючи його взаємодію та зацікавлюючи в переході на іншу сторінку зі збором інвестицій.

Аналізуючи отримані матеріали для розміщення, а саме: текстова інформація, зображення, гіф-анімація та відеоматеріали, було прийнято рішення, компонувати їх одне з іншим для досягнення різноманіття можливостей під час перегляду сторінки та альтернатив під час вибору способу перегляду матеріалів.

Необхідно створити мінімалістичний перший екран, котрий буде відображувати поточний прогрес в процесі збору коштів та пропонувати переглянути наведену нижче на сторінці інформацію.

Оскільки в розпорядженні наявні двоє готових та повністю завершених відеоматеріалів, було прийнято рішення використати один з них на першому

екрані. Він повинен одразу приковувати до себе погляди відвідувачів, тому було запропоновано зробити його більше половини ширини екрана.

В іншій, вільній частині екрану було вирішено розмістити інформацію про кількість вже зібраних коштів та загальної необхідної для завершення інвестування суми, а також кнопки навігації, кожна з яких виконує гіперпосилання на свою інформацію. Перша з них з назвою «Button 1»повинна при натисканні робити плавний перехід від першого екрану до матеріалів, котрі розміщуються нижче. Це буде реалізовано за допомогою посилань всередині одного документу, а точніше якорю, котрий буде встановлений на певному місці в документі. Головною ціллю існування якорів в документі є те, що вони дозволяють атрибутам робити гіперпосилання не на сторонній документ або сторонню web-сторінку, а саме на окремо виділені місця в тому ж документі, дозволяючи робити навігацію користувачу по одній сторінці.

Друга ж кнопка «Button 2» буде містити гіперпосилання на сайт KickStarter.com, на конкретну сторінку даного проекту, з метою залучити клієнта до інвестування. Вона буде реалізована з допомогою того ж методу, що і попередня, проте посилатись буде вже на сторонню сторінку.

Найголовнішою частиною першого екрану було вирішено зробити саме кругову діаграму, що відображає поточний прогрес в процесі збору коштів. Її реалізація буде вимагати підключення сторонніх модулів, створення запитів та отримання відповідей від іншого серверу та використання API-технологій, що будуть брати дані з сторінки проекту на сайті KickStarter.com та перетворювати ці дані в кругову діаграму. Зовнішнє оформлення кругової діаграми буде реалізовано за допомогою CSS та її взаємодії з функціями мов JavaScript та PHP.

Наведений на Рис. 2.1. приклад гарно поєднує визначені вимоги до першого екрану, тому він буде використовуватись як макет першого екрану.

Рис. 2.1. Макет першого екрану сторінки

Просуваючись далі, необхідно розробити загальний макет сайту. Головним пріоритетом для сторінки залишається утримання уваги користувача та заохочення його до подальшого перегляду. Тому найбільш оптимальним варіантом буде комбінувати між собою відео, текстові, фото матеріали, а також гіф-анімацію. Основна ідея буде відтворена з сторінки збору коштів на KickStarter.com, де розміщена головна інформація, що стосується всього розроблюваного проекту.

Матеріали будуть викладені послідовно не суміщаючи в одному ряді інформацію різного типу, адже враховуючи невеликий об'єм для розміщення даних, необхідно повністю розкрити суть проекту та залучити користувача стати клієнтом.

На Рис.2.2. зображено планування послідовного розміщення матеріалів. Різноплановість інформаційних матеріалів допоможе втримати увагу користувача та не набриднути йому, тим самим дозволить йому переглядати сторінку та читати й далі та дізнаватись все більше про мету даного проекту.

Рис. 2.2. Макет другого екрану сторінки

З метою обмежити можливість користувача перейти одразу в кінець сторінки було прийняте рішення прибрати полосу прокрутки. Таким чином користувач зможе сприймати інформацію послідовно та краще її засвоювати. Для допомоги в навігації без полоси прокрутки допоміжним рушієм буде «Кнопка 2», котра ініціює перший перехід до нижченаведеної інформації.

Після розміщення всієї інформації, останніми елементами сторінки будуть саме зображення команди розробників, та футер з посиланнями на сторінки в соціальних мережах.

Рис. 2.3. Скетч зображення команди розробників проекту

Футер – у web-програмуванні це нижній колонтитул web-сторінки, розділ, що розташований під основним текстом або останнім елементом. Зазвичай він використовується в якості місця для посилань на сайти партнери або на інші важливі для web-ресурсу сторінки.

Але в односторінкових сайтах зазвичай футер або містить інформацію про розробників, або просто контакти такі як телефон та e-mail, або ж посилання на сторінки в соціальних мережах чи на сторінку завантаження мобільного додатку.

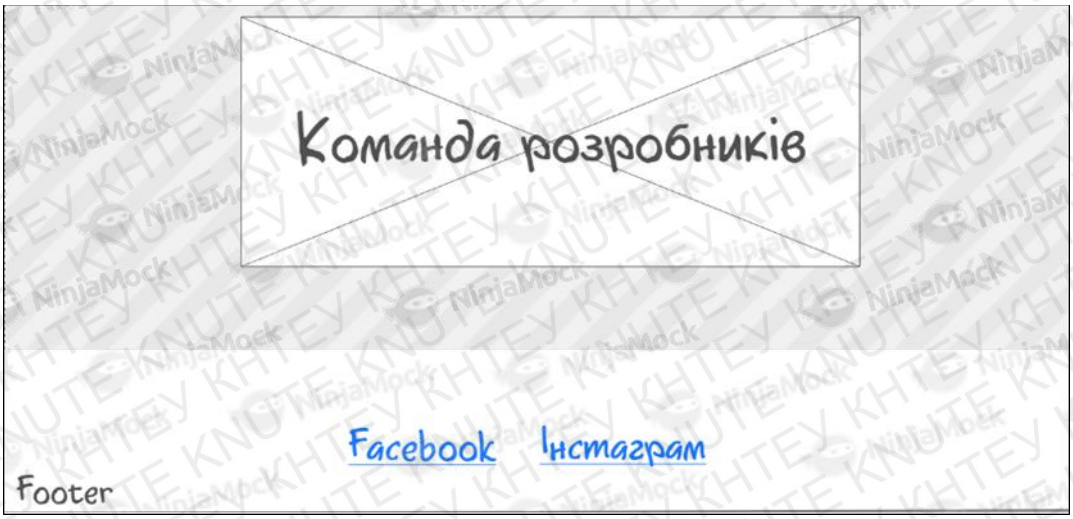

Рис. 2.4. Макет футеру web-сторінки

Оскільки даний проект має створені та функціонуючі сторінки в таких соціальних мережах як Instagram та Facebook, в футері буде розміщено саме посилання на ці сторінки.

Отже, створивши повний макет сайту для десктопної версії, можна переходити до створення макету мобільної версії.

2.1.3. Розробка макету мобільної версії web-ресурсу

Створивши та маючи вже готовий та затверджений макет сторінки, можна переходити до створення макету мобільної версії.

Виходячи з розташування елементів на всьому макеті окрім першого екрану, можна сміливо сказати, що вони розташовані належним чином і для мобільної версії. Елементи сторінки не заважають один одному та знаходяться в чіткій послідовності, утворюючи та розповідаючи цільну історію. Тому ці елементи було вирішено залишити в такому ж вигляді, змінивши лише їх розміри та адаптувавши їх під розширення смартфонів.

Стосовно першого екрану, то тут необхідно провести великі зміни. Оскільки на попередньому макеті дві третини екрану займав саме відео матеріал, то в мобільній версії це недопустимо. Головною відмінністю десктопної версії від мобільної є те, що в першій екран має горизонтальне положення, а в мобільній воно змінюється на вертикальне.

На першому екрані мобільної версії необхідно залишити як кнопки навігації так і кругову діаграму, що відображає поточний прогрес збору інвестицій. Зважаючи на це, відео буде перенесено до другого екрану та буде розміщене одразу після першого екрану. В той же час для мобільної версії необхідно змінити посилання «Button 2» для переходу не до текстової інформації, а до нового місцезнаходження першого відеоматеріалу.

На першому екрані, після всіх змін будуть залишені такі елементи як: логотип, кругова діаграма, текстові поля, що інформують про збір коштів, та дві кнопки навігації, що зображено на Рис. 2.5.

Рис. 2.5. Макет першого екрану в мобільній версії

Однією з головних проблем реалізації даного макету буде питання адаптивності елементів сторінки. Необхідно впровадити відносну систему розмірів елементів, на заміну абсолютній. Причиною цьому є те, що кожен смартфон має своє розширення, та при абсолютних показниках елементів сторінка буде виглядати по-різному на різних девайсах. Оскільки цього необхідно уникати, при розробці web-ресурсу необхідно буде створювати всі елементи у відносній системі виміру.

Головною відмінністю даного макету від попереднього буде відсутність полів протягом всієї сторінки, котрі були присутні в десктопній версії. Це допоможе пропорційно збільшити всі елементи сторінки для читабельності не змінюючи розташування елементів.

Тому після взаємодії користувача з другою кнопкою, буде спрацьовувати перехід до місця наведеного на Рис. 2.6., котре буде розміщуватись одразу після завершення першого екрану.

Рис. 2.6. Нове місцезнаходження якорю переходу

Розташування інших не згаданих елементів буде залишене в такому ж порядку та будуть збережені всі пропорції з першого макету.

2.2. Висновки до розділу

Отже, в результаті проведеної роботи було створено два макета розроблюваного web-ресурсу: макет десктопної версії та макет мобільної версії сайту. За отриманими макетами будуть розроблені відповідно дві версії сайту та поєднані в одну остаточну, що буде постійно перевіряти розширення екрану користувача, тим самим розуміючи, яку з двох версій потрібно створювати в даний момент часу. Поєднання двох макетів в одну версію допоможе скоротити витрати ресурсів та часу на створення web-ресурсу.

Тому створені раніше макети є невід'ємною частиною розробки web-орієнтованої інформаційної системи для закладу громадського харчування.

РОЗДІЛ 3. РОЗРОБКА ВЕБ-ОРІЄНТОВАНОЇ СИСТЕМИ ЗАКЛАДУ ГРОМАДСЬКОГО ХАРЧУВАННЯ

3.1. Налаштування серверу

Проведені в попередніх розділах дії дозволили визначити нам спосіб створення сучасного та конкурентоспроможного web-сайту для закладу громадського харчування.

Першим, під час розробки web-ресурсу, необхідно визначити та налаштувати апаратну частину. Це можу бути як безкоштовний хостинг, так і приватний сервер. Оскільки розроблюваний продукт виготовляється для конкретної компанії «LLC InbicoEurope» та вони мають власний орендований приватний сервер, на котрому бажають розмістити web-сайт.

З отриманими правами доступу до сайту, одразу можна переходити до створення відповідного каталогу та налаштування прав доступу для інших користувачів, кому буде необхідно мати можливість редагувати відкритий код сайту. Використовуватися буде сервер METANET.

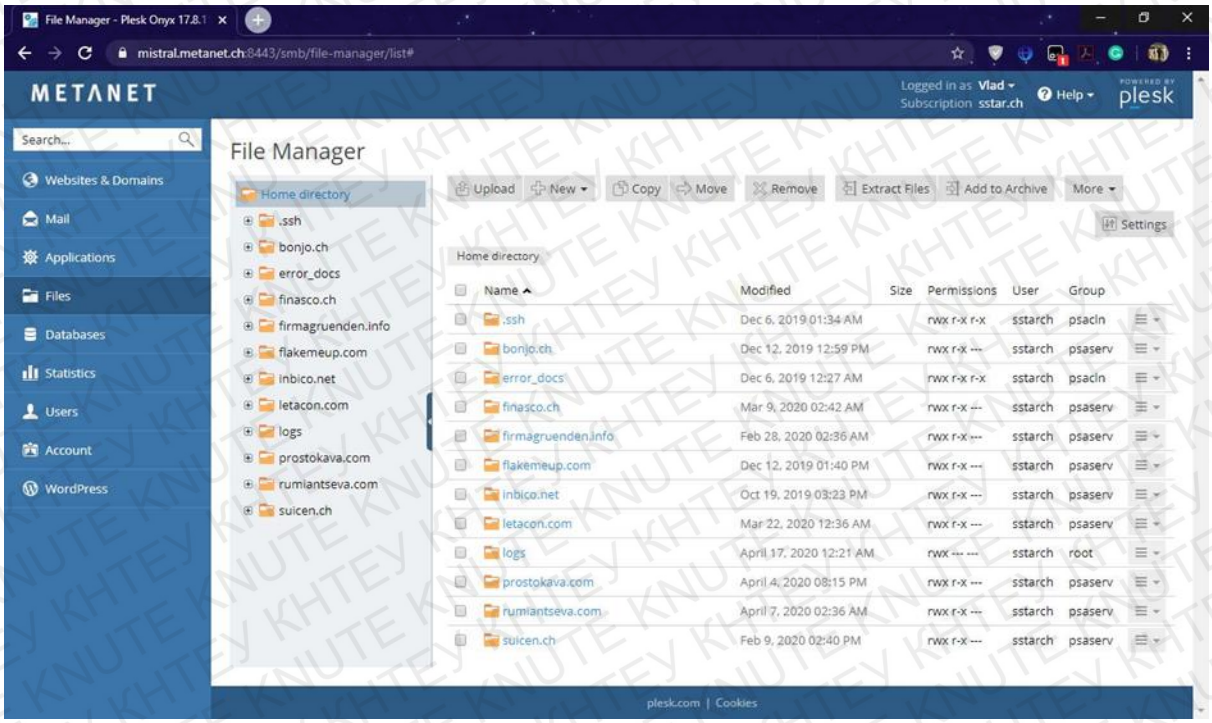

Рис.3.1. Корінна папка web-серверу METANET

Створивши папку «prostokava.com», та виставивши йому в налаштуваннях рівень доступу root, можна переходити до створення та подальшого завантаження необхідних матеріалів на сам сервер.

Обов'язковим етапом створення web-сайту є створення декількох файлів, таких як: *sitemap.xml* та *robots.txt*. Web-ресурс звісно може існувати без цих двох файлів, проте витративши небагато часу на їх створення, адміністратор може забезпечити сайту кращі умови існування.

Sitemap або ж XML-карта сайту –це файл з інформацією для пошукових систем про сторінки, які необхідно проіндексувати. Іншими словами, карта сайту - список всіх сторінок в форматі XML, доступних для сканування пошуковим роботом.

Оскільки наш сайт буде створено односторінковим, цей етап можна пропустити, переходячи одразу до створення наступного файлу – robots.txt.

Файл *robots.txt* або індексний файл - звичайний текстовий документ в кодуванні UTF-8, діє для протоколів HTTP, HTTPS, а також FTP. Файл дає пошуковим роботам рекомендації: що варто сканувати. Якщо файл буде містити символи не в UTF-8, а в іншому кодуванні, пошукові роботи можуть неправильно їх обробити. Правила, перераховані у файлі robots.txt, дійсні лише щодо того хоста, протоколу і номера порту, де розміщений файл.

Файл повинен розташовуватися в кореневому каталозі в вигляді звичайного текстового документа.

Аналогічно попередньому файлу, так як наявна лише одна сторінка то прописати в цьому файлі необхідно саме те, що повинні знаходити пошукові роботи при запиті користувача в пошуковій системі.

Гарним тоном для будь-якого розробника буде створення окремих каталогів в папці сайту на сервері для правильного місцезнаходження файлів та документів, котрі будуть використовуватись. Тому було створено наступні директорії, що зображено на Рис.3.2.

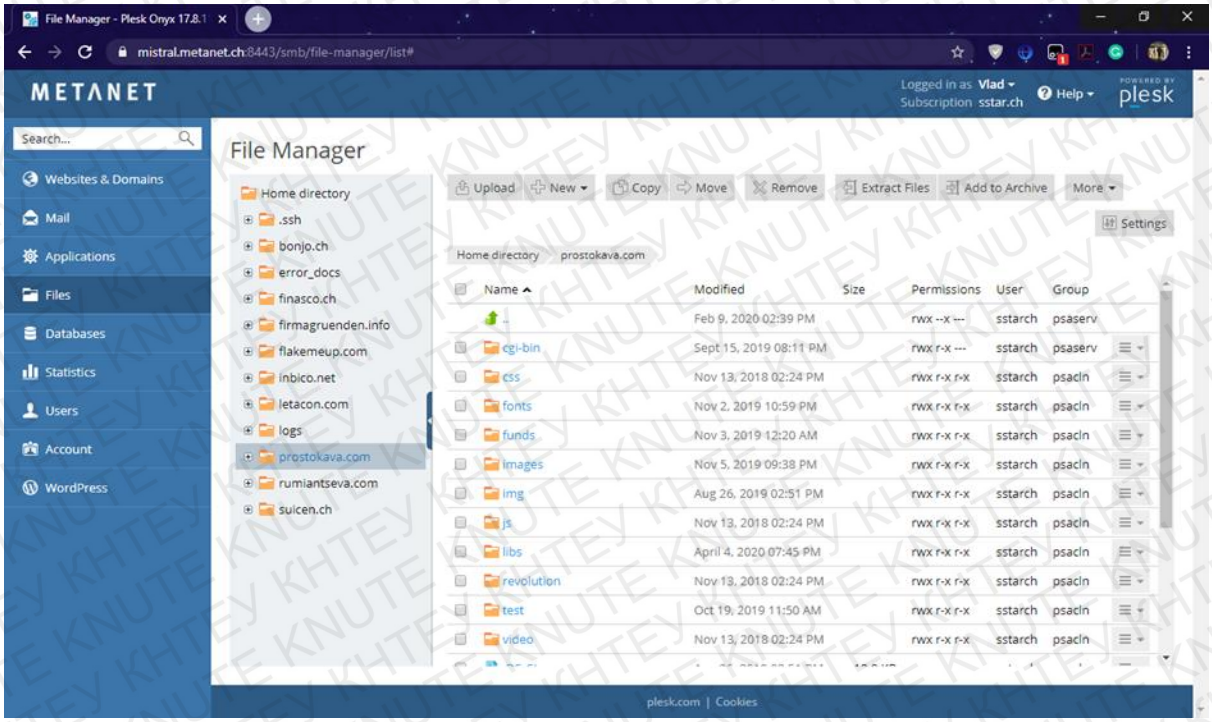

Рис.3.2. Створені директорії для ієрархічної структури сайту

Отже, створивши необхідні для існування web-ресурсу каталоги, отримання необхідних прав доступу та створення файлу *robots.txt*, етап налаштування серверу вважається завершеним.

3.2. Розробка та верстка сайту

3.2.1. Створення розмітки та розстановка елементів сторінки

Оскільки головним інструментом розробки був обраний «SublimeText 3», тому майже весь процес створення та написання коду буде відбуватись саме там.

Маючи макет сайту, можна переходити до розмітки сторінки.

Розмітка сторінки, як випливає з самого терміну – це розстановка міток, в нашому випадку в коді HTML документа, тобто web-сторінки. Мітками тут виступають теги, що дозволяють визначити межі дії розмітки або створити елемент HTML документа.

Переходячи до створення розмітки сторінки, необхідно визначитись з тим, які блоки необхідні для розміщення елементів. Перший екран відповідно до макету сайту буде поділений на два вертикальні блоки. В лівому буде

знаходитись вся інформація (кнопки, логотип, діаграма, текст), а правий блок буде повністю відведена на відеоплеєр. Для того, щоб витримувати якісь відповідні пропорції в розмірах елементів, перший блок буде займати 33% від розміру екрану користувача, а другий інші 67%. Таким чином можна легко поділити екран в пропорції 1:2.

В свою чергу перший блок буде складатись з горизонтальних підблоків, кожен з яких буде містити відповідно один елемент. Вони будуть також мати відносні розміри у відсотках, що допоможе забезпечити адаптивність сайту під різні розширення екранів користувачів. На Рис.3.3. зображена попередня розмітка першого екрану.

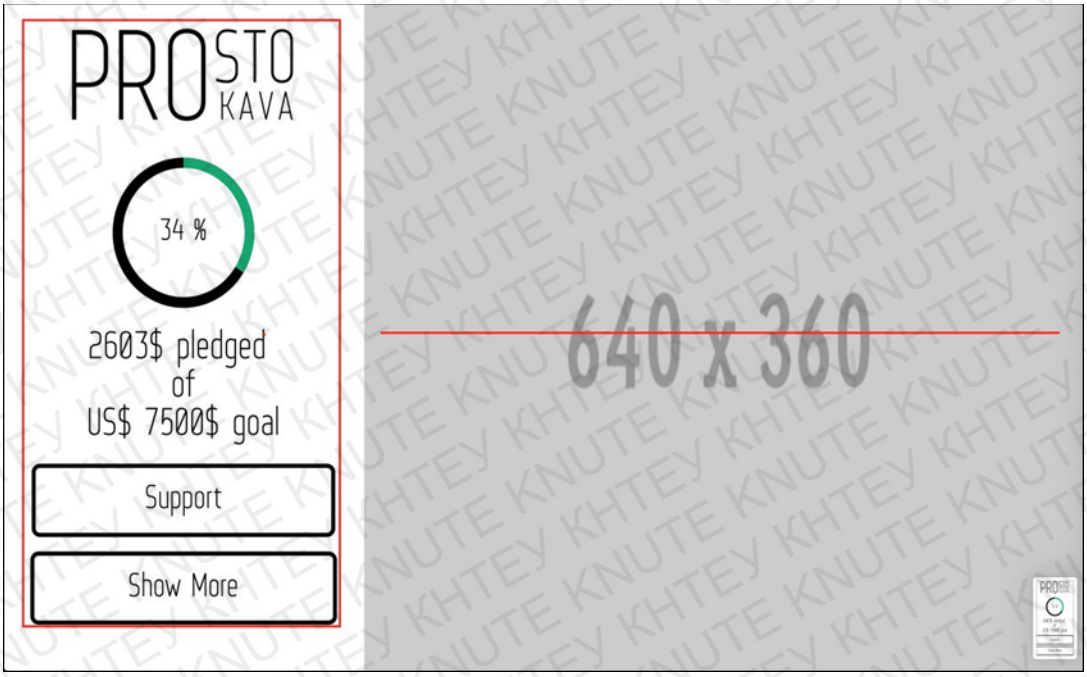

Рис.3.3. Розмітка першого екрану сайту

Наступним етапом, є розмітка іншої частини сайту, котра буде містити всю інформацію, що необхідно донести до користувача. Ще з макету випливало те, що ц частина сторінки буде мати певні поля, для кращої читабельності інформації та візуальної централізації елементів.

Тому вся інша сторінка буде поділена на 3 колонки на які відповідно буде виділено 15% на першу, 70% на центральну та 15% на останню. Перша та остання не будуть містити жодної інформації, вони будуть забезпечувати

одинакові поля для центральної, котра і буде в свою чергу містити всю інформацію.

Розмітка центральної частини буде максимально простою, що іноді є корисним, адже чим менше непотрібних елементів на сторінці, тим швидше відбувається її завантаження в браузері. Кожен наступний елемент буде розміщуватись за попереднім. Для всіх блоків будуть використовуватись теги в середині яких вже будуть розміщені відповідні елементи такі як: текст, зображення, анімація або ж відео.

Під цим великим блоком, найнижчим та останнім блоком сторінки буде окремий блок для футера, що буде містити посилання на сторінки в соціальних мережах. Він буде мати ширину 100% від ширини сторінки та відносно невелику висоту.

Останнім в розмітці сторінки необхідно ще раз повторно передивитись всю щойно створену структуру. Якщо ніде нічого не було залишене з відкритим тегом або ж просто без необхідного блоку, то можна переходити до наступного етапу.

3.2.2. Наповнення сторінки матеріалом

Після створення розмітки сторінки, можна переходити до її наповнення необхідною інформацією.

Отримані матеріали від розробника проекту одразу необхідно поділити за категоріями. Всі зображення та гіф-анімація будуть зберігатись на сервері в окремій папці *«images»*, в той час як відеоматеріали будуть знаходитись в теці *«video»*. Ці прості розмежування файлів, що використовуються як елементи сторінки, дозволяють розробникам краще підтримувати структуру web-сторінок. Якщо буде якась необхідність швидко змінити якийсь елемент, то навіть адміністратор, котрий не розробляв сайт, зможе знайти місцезнаходження шуканого файлу.

Текстову інформацію одразу необхідно розміщувати прямо у відповідних тегах. Зображення та гіф-анімація також розміщуються просто в тегах за

допомогою власного тегу <*img*>, ціль існування якого і є в завантаженні зображень на сторінку.

Відеоматеріали ж необхідно завантажувати інакшим чином. Існує одразу декілька способів розміщення відео на web-сайті. Перший і найпоширеніший – YouTube-програвач. Якщо завантажити необхідний відеоматеріал на YouTube, то там автоматично генерується тег для вставки відео на сторінці. Цей спосіб є найпростішим, адже він не навантажує сервер, тому що відео зберігається і завантажується на серверах YouTube, тому сторінка з будь якою потужністю серверу може собі дозволити такий спосіб вбудовування відео. Також цей тег одразу підбирає розміри програвача та дозволяє розробнику, шляхом невеликих змін налаштувати відео під свої потреби.

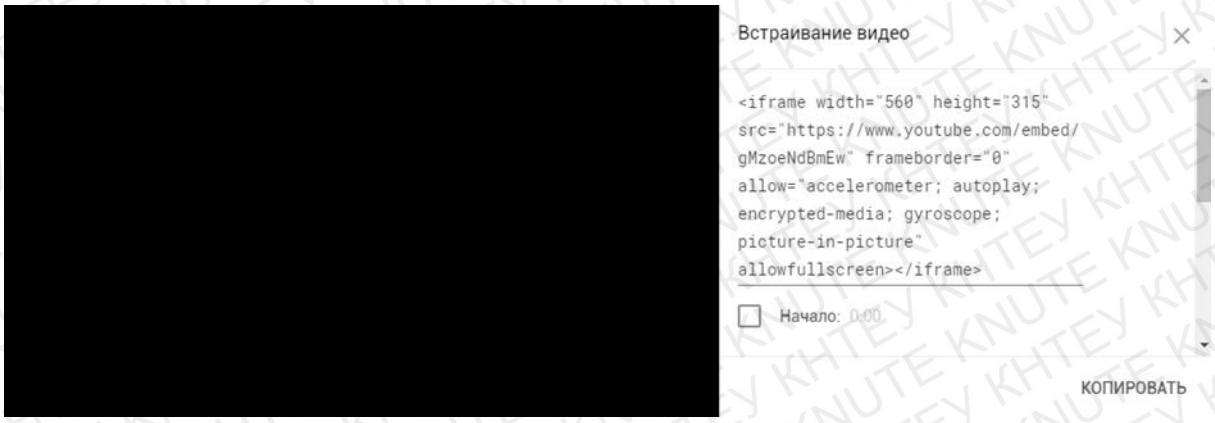

Рис.3.4. Автоматично створений тег вбудови відео від YouTube

Але існує і другий, менш поширений та менш зручний спосіб вбудовування відео на сторінку. Використання HTML5 плеєру. Цей плеєр є більш зручним для змінення його розмірів, пропорцій та положення на сторінці, проте він вимагає завантаження всього відео на сервер. Якщо сервер має слабку потужність, то він буде погано завантажувати відео в програвачі. Але на відміну від програвача YouTube в даному плеєрі все залежить від серверу сайту, тому він є повністю незалежним.

І хоча перший спосіб є набагато простішим у використанні, проте для виконання вимог даного проекту використовуватись буде саме другий спосіб з плеєром від HTML5. Оскільки в проекті будуть присутні два відео, вони обоє будуть завантажені даним методом.

Для внесення новизни та індивідуальності в сторінку, було вирішено змінити можливості користувача у взаємодії з медіа плеєром. А саме прибрати у відео програвача полосу прокрутки. Таким чином користувач матиме можливість продивитись відео лише повністю від початку до кінця.

Реалізація цього дуже проста, вона вимагає відключення одного з атрибутів тегу *<video>*, а саме написання наступного коду в файлі стилів CSS:

video: { display: none !important;}

Для кращого оформлення головного відео з першого екрану, було вирішено створити модальне вікно, в якому буде завантажуватись відео.

Модальне вікно відкривається при натисканні на фрейм першого відео. На ньому розміщується централізований блок, що містить в собі програвач відео.

Для виокремлення цього блоку від сторінки на фоні використовувалось заповнення всього фону напівпрозорим елементом, котрий містив в собі функцію, що закривала модальне вікно при натисканні, тим самим забезпечивши можливість користувача закрити відео та просуватись й далі по сторінці.

Рис.3.5.Модальне вікно для програвання першого відеоматеріалу

Проте головною проблемою використання даного прийому є те, що після цього користувач втрачає можливість запуску та паузи відео. Тому головним пріоритетом стала розробка функції, що дозволяє запускати та зупиняти програвач при натисканні на фрейм з відео.

Для створення даної функції використовувалась мова JavaScript та її бібліотека jQuery.

Тегу <*video*> було присвоєне власне значення *id=«vidplay1»*, що дозволяє ідентифікувати цей елемент окремо від інших серед всього документу. Також необхідно було створити змінну типу *bool*, котра приймала б всього два значення і відповідала б за стан відео зараз. Якщо стан змінної *false*, тоді наступним натисканням буде перемикання відео на паузу, а якщо стан змінної *true*, тоді відео необхідно буде запускати.

Реалізація цієї ідеї була можлива за допомогою функцій бібліотеки jQuery, а саме:

1) перш за все, необхідно створити функцію, котра буде запускатись після натискання користувачем в рамках фрейму з відео;

2) після цього йде перевірка умови*: if (radio == true)*, якщо значення змінної true, тоді виконується пункт №3 з запуском відео, якщо ж значення змінної інше, то виконується пункт №4.

3) при виконанні умови (*radio == true*) запускається наступна функція: *$('#vidplay1').get(0).play()*, котра виділяє елемент з *id=«vidplay1»,* отримує доступ до керування відео через функцію *get(0)* та запускає відео через вбудовану функцію *play*(). Після чого обов'язковою є заміна значення змінної *radio* на *false*, адже при наступному натисканні, необхідно буде уже зупиняти відео.

4) В разі невиконанні умови (*radio == true*), а оскільки змінна типу *bool* може приймати лише два значення, то при (*radio == false*) запускається наступна функція: *$('#vidplay1').get(0).pause()*, котра виділяє елемент з *id=«vidplay1»*, також отримує доступ до керування відео через

функцію *get*(0) та зупиняє відео через вбудовану функцію *pause*(). Після цього також обов'язковою є заміна значення змінної на протилежну.

Дана функція копіюється та дещо видозмінюється для наступного відео, проте її функціональність та спосіб реалізації залишаються такими ж. Таким чином, була розроблена функція, що дозволяє керувати відео без полоси прокрутки програвача.

Щоб користувач зрозумів, що по відео необхідно натиснути, була використана можливість, надана атрибутом *poster*, котрий дозволяє встановлювати будь яке зображення як заставку на відео до його першого запуску, тоді *poster* пропадає і відео йде у звичайному режимі. Заставка навмисно була створена таким чином, щоб вона нагадувала вікно програвача та давала візуальні підказку користувачеві в запуску відео.

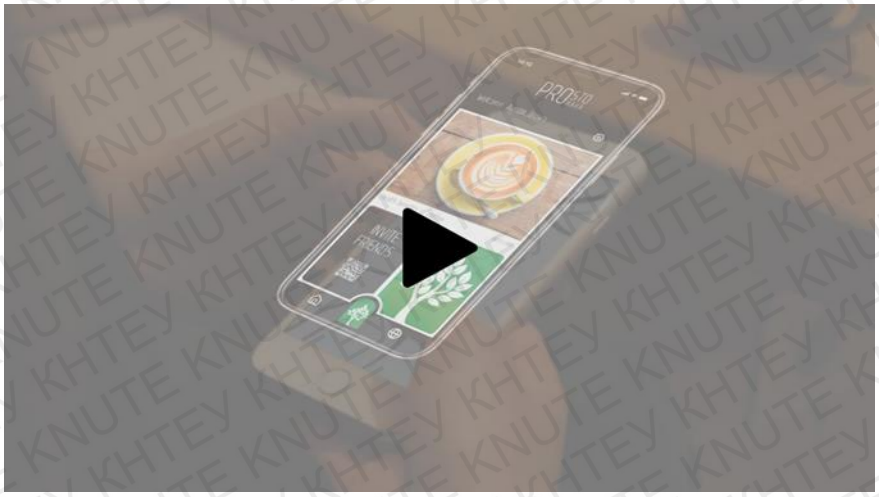

Рис.3.6. Приклад заставки в атрибуті poster на відео

3.2.3. Створення стилів та дизайну сторінки

Створення дизайну сайту – найголовніший етап в створенні сторінки. Саме від дизайну, грамотного поєднання кольорів, розстановки матеріалів, шрифтів, зображень, поєднання їх в одну композицію і робить сторінку привабливою для користувача. Саме дизайн показує рівень розробника та цінність компанії. Тому до цього питання завжди слід підходити ретельно [13].

Оскільки макет та розмітка сторінки вже створені, необхідно було підібрати необхідні стилі, які б задовольнили вимоги замовника та перетворили звичайний каркас розмітку на візуально завершений продукт.

Перш за все, для забезпечення повної ширини сайту та неможливості користувача пропустити важливу інформацію, було вирішено прибрати бічну полосу прокрутки. Реалізується це просто, прописавши в стилях наступний код: *html: overflow hidden.*

Наступним важливим кроком є підбір шрифтів, котрі будуть використовуватись на сторінці. Завдяки можливості обирати серед більш ніж восьмисот шрифтів на сайті fonts.google.com, було відібрано три головних шрифти, котрі будуть застосовані в різних частинах сайту.

Перший шрифт буде використаний на першій сторінці відображаючи дані по збору коштів, другий буде застосований до заголовків та написів на кнопках, а третім буде відформатований весь текстовий матеріал з блоків.

Прописавши елементам з однаковим оформленням класи та id ідентифікатори можна переходити в файл *style.css*, де зберігаються та визначаються стилі сторінки. Там необхідно прописати всі належні стилі, вирівнювання тексту, положення елементів, відносні та абсолютні розміри елементів, центрування існуючих блоків, заміну кольорів тексту та фонових елементів. Таким чином відбувається процес стилізації сайту та його перетворення з макету на готовий кінцевий продукт.

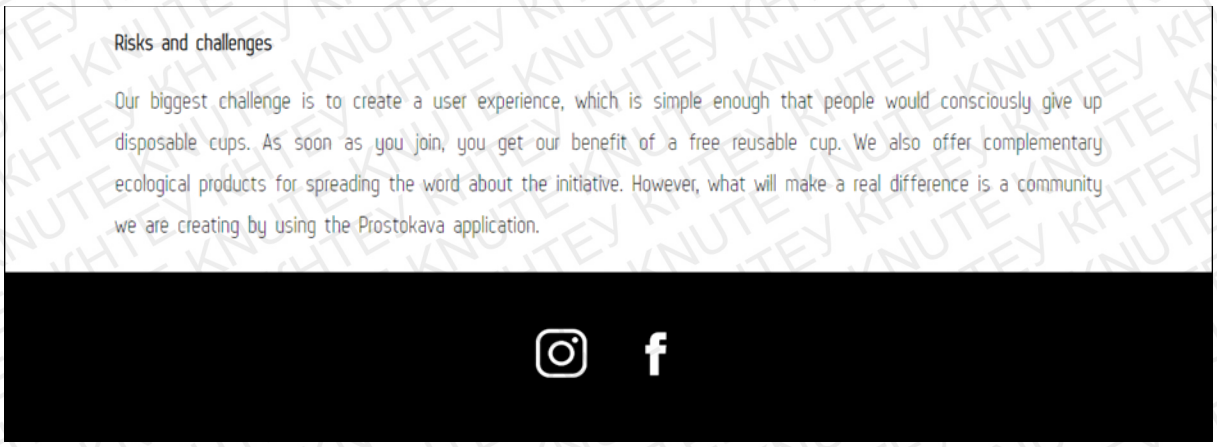

Рис.3.7. Футер з гіперпосиланнями

В нижній частині сайту, а точніше в футері, де повинні були міститись посилання на соціальні мережі, необхідно розмістити векторні зображення цих мереж та підключити гіперпосилання на них.

3.3. Робота з API-технологіями

3.3.1. Впровадження API-технологій до сайту

Завершивши верстку сайту, тобто створення візуальної частини сайту або *front-end*, можна переходити до створення технічної частини розробки. Для використання API-технологій необхідна певна підготовка.

Перш за все необхідно, щоб головний файл сайту мав розширення не.*html*, як у звичайних сторінок, а *.php*, щоб мати можливість використовувати функції мови PHP.

Необхідно перевірити, чи є в розробника всі належні права доступу. В разі їх відсутності, необхідно перевірити та задати на сервері повторне привласнення прав між користувачами та адміністраторами серверу.

Після цього необхідно створити папку, котра буде містити код, котрий буде робити запит, отримувати та розкодовувати відповідь та передавати її в головний файл для використання в виведенні інформації на екран.

Для реалізації запиту була використана розроблена раніше технологія для отримання відкритої інформації з web-сайтів.

Для того, щоб вона могла виконувати свої функції, їй необхідно мати посилання на конкретну сторінку, котру необхідно прописати в правильному місці в коді головної сторінки. Тоді відбувається звернення до вказаного сайту, де з допомогою прописаних функцій відбувається пошук потрібного елементу серед всіх елементів сайту. Після знаходження відповідного елементу, функція переходить до того, що виокремлює текстову частину, що знаходиться в даному блоці та копіює її в свій буфер обміну, після чого надсилає собі ж назад на сервер[18].

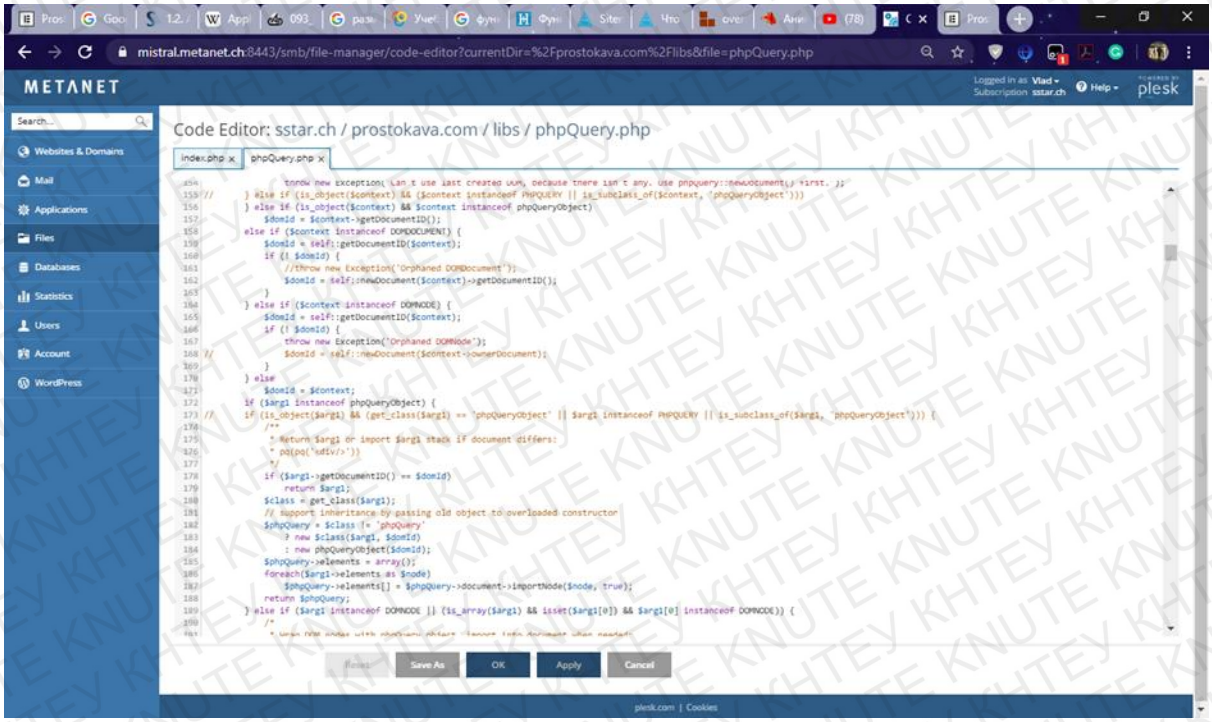

Рис.3.8. Вбудований серверний редактор коду з функціями API

Отримавши дані, починається процес їх обробки. Він поділяється на декілька етапів:

1. Перевірка отриманих даних. Після отримання даних необхідно одразу перевірити їх на коректність та відповідність цифровому типу. Якщо серед вивантажених з буферу обміну даних є не лише числова інформація, тоді відбувається перехід до етапу №2.

2. Якщо серед числової інформації є текстові, або інші недопустимі символи, тоді отримані дані перетворюються з числового типу даних *int* в текстовий тип *string*.

3. Наступним кроком є виокремлення виключно числової інформації та видалення всього зайвого з допомогою вбудованих функцій та використання циклу *for*(). Він проходить по кожному символу змінної *string* та визначає які символи є числами.

4. Після виділення позицій виключно числових даних, відбувається видалення зайвих символів, та конвертація отриманого результату до числового формату *int*.

5. Створення копії даної змінної з отриманою інформацією, для забезпечення надійності в подальшій роботі з даними.

Таким чином, отримана інформація перетворюється на коректні дані, що будуть використовуватись в побудові діаграми.

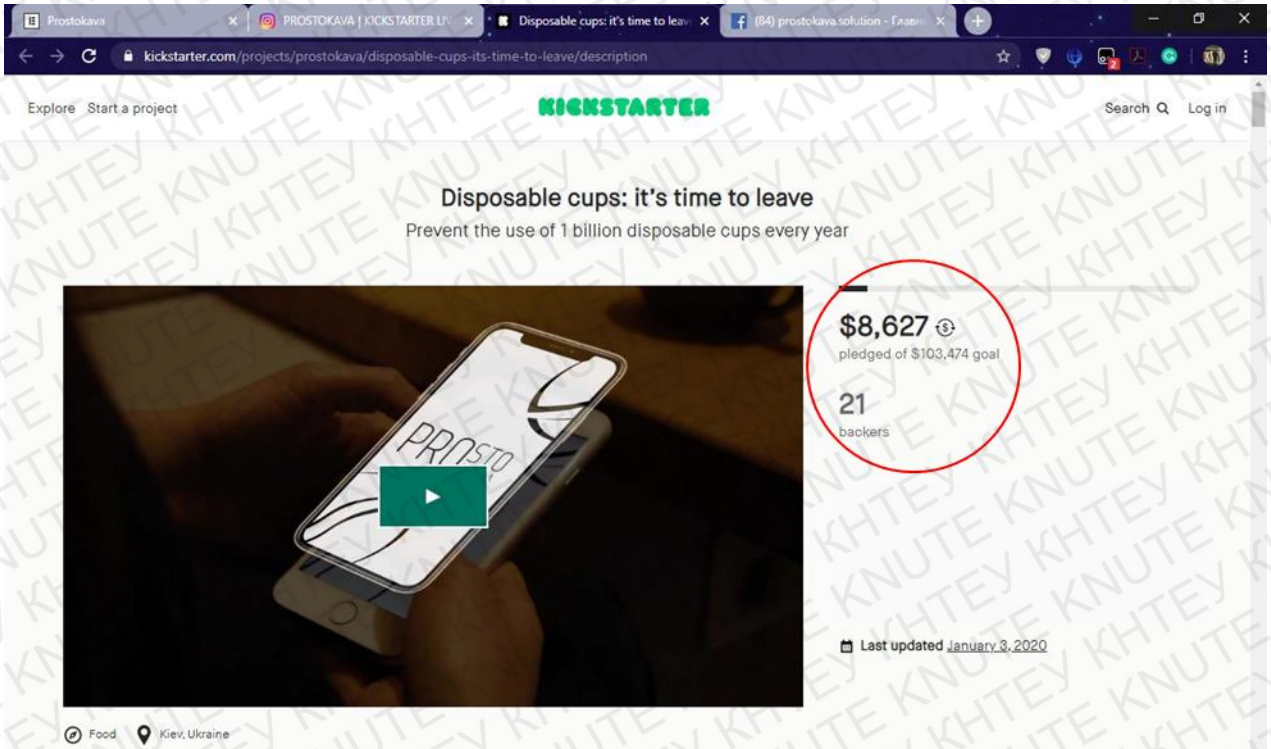

Рис.3.9. Сторінка збору коштів на kickstarter.com, з виділеними числовими даними для копіювання

Після отримання оброблених коректних даних, можна переходити до їх вставлення на сторінку.

Для виведення наявної інформації на екран буде використовуватись самий звичайний метод мови PHP *echo*(), котрий просто виводить на екран інформацію. Для правильного оформлення, використання цієї функції буде одразу прописане в тег, де повинна буде знаходитись інформація, а за допомогою стилів, котрі будуть надані блоку в якому це відбуватиметься, можна буде легко форматувати та змінювати текст для відповідності всьому стилю сайту. Нижче наведено приклад виведення інформація на сайт

<?php echo $project -> percentFunded >

За допомогою функцій мови PHP стала можлива передача змінних між різними файлами, що знаходяться на сервері. Саме завдяки цьому передача

одразу багатьох змінних не викликає жодних проблем та навіть не знижує швидкість завантаження сайту, адже вся передача даних відбувається в асинхронному потоці, котрий проходить одразу паралельно до завантаження сторінки.

3.3.2. Використання API-технологій для створення кругової діаграми

Після передачі та вставки даних з Kickstarter.com до сторінки, ще одним надзвичайно важливим етапом було створення кругової діаграми. Адже навіть після вставки даних, сторінка не виглядала належним презентабельним чином.

Для розробки кругової діаграми прогресу збору коштів було вирішено не звертатись до інших сторонніх мов програмування та застосувати одну з раніше використовуваних – CSS. Оскільки головне призначення даної формальної мови це опис зовнішнього вигляду сайту та робота з створеними векторними елементами, вона ідеально підходить для створення кругової діаграми.

Оскільки CSS не є мовою, де можна передавати змінні або проводити будь-які прикладні дії, для візуально динамічної кругової діаграми було вирішено діяти наступним чином.

Було створена візуалізація кожного з випадку заповнення діаграми. Оскільки вона може набувати лише значень від 0% до 100%, необхідно було зробити сто один варіант зображення діаграми, використовуючи правильний кут заповнення. Проте навіть створення сто однієї діаграми є достатньо довгим та нераціональним вчинком, тому був знайдений спосіб скоротити кількість діаграм вдвічі.

Після того, як діаграма перейде за 50%, вона пройде рівно половину і тому можна використовувати відштовхування від точки 50% як від нової стартової точки і на кожний випадок, коли відсоток більше за п’ятдесят заповнювати праву половину як шаблон і уже далі формувати іншу половину залежно від отриманого відсотка. До прикладу: якщо діаграма повинна бути заповнена на 64%, то вона автоматично заповнюється на половину, адже це

число більше за 50% та вже потім будується відповідна діаграма на 14% відштовхуючись від нової стартової точки в 50%.

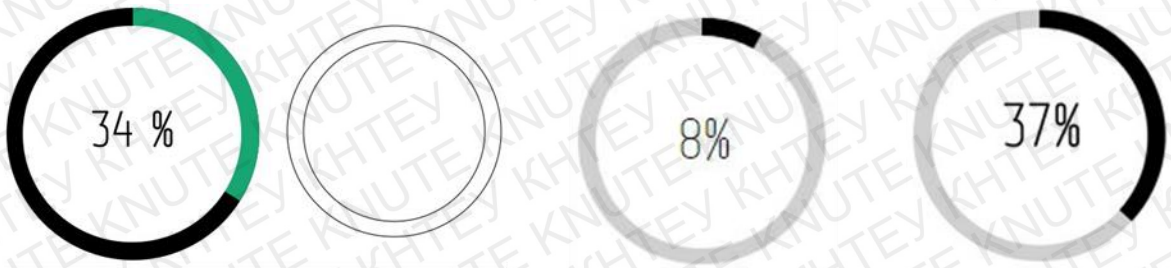

Рис.3.10. Прототипи та можливі варіанти дизайну кругової діаграми

Таким чином, створивши діаграму для кожного з можливих випадків, можна було переходити до її впровадження на сайт.

Оскільки, як було вище зазначено, мова CSS не є прикладною і не може робити жодних перетворень та змінних, всі такі дії було відведено для виконання мові JavaScript. Кожній з варіантів діаграм було присвоєно своє ім'я класу, котре показувало те, для якого конкретно відсотку створений цей варіант. Написана на мові JavaScript функція, виконувала перевірку отриманих API-даних, а конкретніше вона ділила кількість вже зібраних коштів на загальну суму та вираховувала процент.

Після цього вона переходила до перевірки числа на те чи воно більше за 50% чи ні. Незалежно від відповіді, яку отримувала функція, вона виконувала перетворення назв класу елементів. Це вбудована функція котра доступна лише мові JavaScript та дозволяє змінювати назву класу будь-якого об'єкту на іншу назву класу.

Оскільки на кожен варіант діаграми було створено окремий клас, таким чином змінивши ім'я класу блоку на інше, можна було змінити зовнішній вигляд і діаграми, тим самим візуально створюючи динамічну кругову діаграму.

3.4. Розробка форми замовлення продукту

Наступним етапом, можливо навіть найважливішим етапом, є розробка форми для замовлення продукту з розроблюваного web-ресурсу. Звісно, продукція, представлена на створюваному сайті, буде доступна до придбання з

полиць магазинів та споживання в відповідних кафетеріях “*Prostokava*”, проте на даному етапі збору коштів на їх побудову, продукція буде доступна до придбання на даному web-ресурсі.

Перш за все для створення належної форми замовлення, спочатку необхідно створити тег <*form*> та прописати всі поля вводу текстової інформації. Для цього використовувалось поєднання тегів <*input*> та <*label*>.

Після створення та належного оформлення, необхідно було створити та стилізувати кнопки, котрі б відповідали за додання та видалення вкладень до замовлення, а також кнопку відправки форми. Оскільки призначення останньої кнопки значно відрізняється від всіх інших, наявних на сторінці, вона була створена за допомогою тегу <*input type=”submit”*>та стилізовано під всі інші кнопки, котрі були створені тегом <*button*>.

Кнопки «*Add documents*» та«*Reset documents*» створені відповідно для додавання вкладених файлів та для видалення помилково вкладених файлів. Реалізація їх взаємодії з користувачем була розроблена з допомогою поєднання мови програмування JavaScript та PHP.

Наступним кроком розроблення форми було додавання власне товарів на сторінку. Зважаючи на їх відносно невелику кількість, для кращого візуального оформлення було обрано використовувати повноцінні зображення прототипів продукції. Тому в правій частині екрану було створено блок, котрий містить зображення трьох товарів.

Під кожним з товарів було створено лічильник їх кількості. Натискаючи на стрілки вліво та вправо, користувач може змінювати кількість товару для замовлення.

Після створення, розміщення та стилізації всіх елементів форми, або ж після завершенняfront-end частини, прийшов час для роботи над back-end частиною. Функції, котрі працюють з елементами вводу інформації, оброблюють його та відправляють на сервер, створювались за допомогою мови PHP.

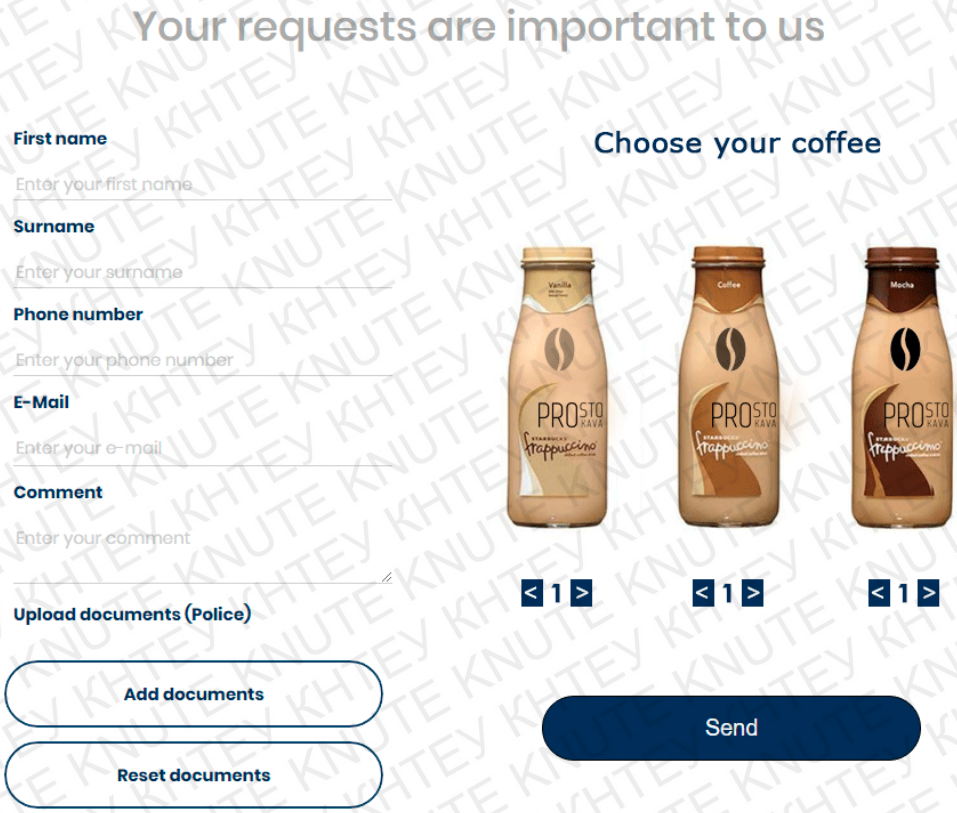

Рис. 3.11. Форма замовлення продукту

Процедура створення та відправлення форми виглядає наступним чином:

1) користувач заповнює текстові поля та обирає необхідну продукцію;

2) перший етап завершується лише в момент натискання користувачем кнопки «*SEND*», котра і є початком роботи PHP функцій на сервері;

3) після натискання кнопки «*SEND*» функції PHP присвоюють новим пустим змінним значення відповідних заповнених користувачем полів.

4) використовуючи раніше створений шаблон замовлення, нові змінні із введеними даними вносяться в відповідні місця шаблону, заповнюючи пробіли та формуючи лист замовлення для його

відправлення користувачеві на вказану електронну адресу та збереження його копії на сервері для подальшого опрацювання.

5) на головній сторінці, після натискання кнопки «*SEND*» з'являється подяка за замовлення користувачеві та сторінка оновлюється очищаючи всі поля вводу для наступного замовлення.

```
    $('#contactme_form').submit(function(e){
e.preventDefault();
var data = $('#contactme_form');
var formData = new FormData();

formData.append('vorname', data[0][0].value);
formData.append('nachname', data[0][1].value);
formData.append('telefonnummer', data[0][2].value);
formData.append('email', data[0][3].value);
formData.append('aktuelle', data[0][4].value);
formData.append('kommentar', data[0][5].value);


if( data[0][6].files[0] ){
      var policephoto0 = data[0][6].files[0];
      var policephotoName = policephoto0.name;
      var policephotoExt = policephotoName.split(".").pop().toLowerCase();

      var policephotoSize = policephoto0.size;
      if(policephotoSize > 15000000){
          alert("Image File Size is very big");
      } else {
          formData.append('policephoto0', policephoto0);
           console.log("0");
      }
    } else {
      console.log("There is no policephoto photo");
    }
      var policephotoSize = policephoto4.size;
      if(policephotoSize > 15000000){
          alert("Image File Size is very big");
      } else {
          formData.append('policephoto4', policephoto4);
           console.log("4");
      }
    } else {
      console.log("There is no policephoto photo");
    }

    $('#popup-position').fadeIn(200);

  $.ajax({
    type: 'POST',
    url: 'contactme.php',
    enctype: 'multipart/form-data',
    dataType: 'text',
    processData: false,
    contentType: false,
    cache: false,
    data: formData
  });
document.getElementById('btnEstimatePicture1').value = "";
        var data = $('#contactme_form');
        var upload = document.getElementById('cont');
        upload.innerHTML="";
document.getElementById("contactme_form").reset();
```

Рис. 3.12. Функції обробки та відправки форми замовлення в програмному середовищі Sublime Text 3

3.5. Розробка мобільної версії сайту

Після завершення повної розробки сайту, для створення мобільної версії сайту необхідно було створити функцію, котра буде постійно перевіряти розширення екрану користувача, і якщо він захоче зменшити або розширити розмір вікна браузера, таким чином, коли розширення стане менше вказаного показника, розстановка елементів зміниться і з десктопної версії сайт одразу перетвориться в мобільну версію.

Це може бути легко реалізоване за допомогою функції *document.body.clientWidth* на мові JavaScript або функції *$(window).width()* з бібліотеки *jQuery*. Поставивши цю функцію на постійне виконання з інтервалом в одну секунду, можна отримати постійну перевірку розширення екрану користувача.

При роздільній здатності в 800 пікселів та більше завантажується десктопна версія, та лише коли розширення менше вказаного, завантажується

мобільна версія. Як вже вказувалось раніше, все окрім першого екрану розташовувалось послідовно, слідуючи один за іншим, тому таке розташування елементів ідеально підходить для мобільної версії сайту.

Тому попрацювати необхідно лише з першим екраном. Виходячи з створеного макету мобільної версії сайту, на першому екрані повинні залишатись лише логотип, діаграма та кнопки навігації. Для цього було вказано в CSS наступне *@mediaonlyscreenand (max-width: 800px){…}*.

Рис.3.13. Мобільна версія сайту

Вказані в цьому коді стилі будуть застосовані лише в тому випадку, коли розмір екрану буде менше за 800 пікселів. Тому вказавши тут стилі, котрі повністю змінюють розмітку сторінки, це ніяк не відображається на десктопній версії сайту та не впливає на попередню розстановку елементів, а завдяки тому, що всі елементи мали відносні розміри, вони не будуть змінювати своє положення та свої розміри на різних пристроях.

ВИСНОВКИ ТА РЕЗУЛЬТАТИ

В ході виконання випускного кваліфікаційного проекту були досліджені питання, що стосуються web-орієнтованих інформаційних систем, були розглянуті основні способи проектування та створення web-ресурсу, основні підходи, переваги та недоліки створення web-орієнтованої інформаційної системи закладу громадського харчування, принципи, котрі використовують при розробці web-ресурсу для подальшого збору коштів шляхом краудфандингу.

Було проведено доцільне обґрунтування та комплексна розробка web-сайту, з урахуванням сучасних світових тенденцій побудови організаційних та функціональних інформаційних структур підприємств.

Для отримання якісного продукту доцільно використовувати найбільш підходящі для даних цілей програмні засоби. Саме тому даний проект вмістив у себе технології і засоби розробки web-сторінок, такі як HTML, JavaScript, CSS, PHP, jQuery та технології вибірки інформації з сторонніх сайтів API.

В другому розділі було проведене проектування web-ресурсу для представлення та реалізації нового харчового продукту. Було розроблено десктопний макет та макет мобільної версії створюваного сайту. Оглянуто інструментальних засобів для використання технологій взаємодії модулів.

В третьому розділі було проведене повне створення, розробка та впровадження всіх необхідних для реалізації технологій з їх подальшим налаштуванням. Під час створення web-сайту була виконана наступна робота:

Проведено аналіз наявних web-ресурсів зі схожою тематикою.

Проаналізовані першоджерела та визначені основні напрямки виконання роботи.

Проаналізована робота та використання API-технологій.

Створені нові функції для альтернативних варіантів взаємодії користувача з відеоматеріалами.

Створено форму зворотного зв'язку для можливості онлайн замовлення користувачем представленого харчового продукту.

Розроблено web-сайт для забезпечення інвестування з допомогою краудфандингу з використанням API-технологій.

Обґрунтовується вибір програмних засобів реалізації системи, розроблено структуру сайту, описано частини системи і моменти її реалізації. Розроблено та реалізовано інформаційну модель системи.

Створена web-орієнтована інформаційна система буде презентувати новий харчовий продукт та виводити його на Інтернет ринок. Завдяки розробленим функціям та можливостям сайту, користувач зможе перейти на сайт збору коштів для подальшого інвестування, замовити продукт онлайн з допомогою форми зворотного зв'язку, повністю ознайомитись зі всіма етапами створення та розробки презентованого продукту та заохотить потенційного клієнта до популяризації даного ресурсу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Fielding J. Beginning Responsive Web Design with HTML5 and CSS3 / Jonathan Fielding. – New York: Apress, 2014. – 328 с.
2. Сухов К. HTML5 – путеводитель по технологии – М.: ДМК Пресс, 2014. – 352 с.: ил.
3. Никсон Робин «Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSSиHTML5». 5-е изд. – СПб,: Питер, 2019. – 816 с.: ил.
4. Системы управления контентом Joomla и Wordpress. – Режим доступу: http://coolreferat.com/. – Дата доступу:
5. Итан Маркотт Отзывчивый веб-сайт / Итан Маркотт – М:Манн,Иванов и Фербер, 2012. – С. 176
6. Castledine E. Build Mobile Websites and Apps for Smart Devices / E. Castledine, M. Wheeler, M. Eftos. – Melbourne: SitePoint, 2011. – 256 с.
7. Крокфорд Дуглас «Как устроен JavaScript». – СПб,: Питер, 2019. – 304 с.
8. Phil Smith, «Making Site-Specific Theatre and Performance: A Handbook». – Macmillan International Higher Education: 2018.– 264 с.
9. Peleg D. Mastering SublimeText / Dan Peleg. – Birmingham: Packt Publishing Ltd, 2013. – 110 с.
10. Patrick C. M. New Perspectives HTML5 and CSS3: Comprehensive / Carey M. Patrick. – Boston, USA: Cengage Learning, 2016. – 872 с. – (7th).
11. Кит Вуд, «Расширение библиотеки jQuery» / пер. с анг. Киселева А.Н. – М.: ДМК Пресс, 2014. – 400с.: ил.
12. Duckett J. JavaScript and JQuery: Interactive Front-End Web Development / Jon Duckett. – Indianapolis: John Wiley & Sons, 2014. – 640 с.
13. Koloro.ua [Електронний ресурс] : [Інтернет-портал]. – Електронні дані. – [Київ : Брендингове агенство KOLORO 2015-2020]. – Режим доступу: https://koloro.ua/blog/dizain/kak-sdelat-maket-sayta-crazy-ideas.html (дата звернення 21.04.2020) – Как сделать макет сайта, если вы новичок.

14. Michel J. P. Web Service API sand Libraries / Jason Paul Michel. – Chicago: American Library Association, 2013. – 152 с.
15. Html-plus.ua [Електронний ресурс] : [Інтернет-портал]. – Електронні дані. – [Київ : HTML-PLUS.UA 2020]. – Режим доступу: http://html-plus.in.ua/funkcii-v-javascript (дата звернення 13.04.2020) - Функции в JavaScript.
16. «PHP and MySQL Web Development (4th Edition)», Luke Welling, Laura Thomson 848 стр., с ил.; ISBN 978-5-8459-1574-0, 978-0-672-32916-6.
17. Стратегія і основні кроки при розробці web-сайту. – Режим доступу:http://ruszura.in.ua/neobhidno-znaty/stratehiya-i-osnovni-kroky-pry-rozrobtsiweb-sajta.html. Дата доступу: 20.03.2020
18. Тесленко Поняття і класифікація інформаційно-пошукових систем [Електронний ресурс] / Тесленко // Інформаційні системи в аграрному менеджменті. – Режим доступу : http://www.library.if.ua/book/100/6867.html
19. Создание сайта, web-дизайн. – Режим доступу: http://www.artus.ru/. – Дата доступу:12.03.2020
20. Веру Лиа, «Секреты CSS. Идеальные решения ежедневных задач». – СПб.: Питер, 2016. – 336 с.
21. Nixon R. Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5 / Robin Nixon. – Sebastopol: "O'ReillyMedia, Inc.", 2018. – 832 с. – (Fifth edition).
22. Петин В.А. API Яндекс, Google и других популярных веб-сервисов. Готовые решения для вашего сайта / Виктор Александрович Петин. – Петербург: БХВ, 2012. – 480 с
23. “Веб Database Application with PHP and MYSQL”, 2nd Edition By David Lane, Hugh E. Williams. © O'Reilly, May 2004. ISBN: 0-596-00543-1.
24. Басюк Т.М. Принципи побудови системи аналізу та просування інтернет ресурсів / Т.М. Басюк // Вісник Нац. ун-ту “Львівська політехніка” “Комп’ютерні науки та інформаційні технології”. – 2012.– № 784.

25. Hutchison E. Drawing for Landscape Architecture: Sketch to Screen to Site / Edward Hutchison. – London: Thames & Hudson, 2019. – 256 с.
26. Інформаційні системи та бази даних / Навчальна електронна бібліотека факультету кібернетики Київського національного університету імені Тараса Шевченка [Електронний ресурс]. – Режим доступу: http://www.unicyb.kiev.ua/Library/BD/intr.doc.
27. Pfeiffer S. Beginning HTML5 Media: Make the most of the new video and audio standards for the Web / S. Pfeiffer, T. Green. – NewYork: Apress, 2015. – 304 с. – (2th).
28. Базы данных: разработка и управление: Книга/ Хансем Г., Хансем Дж. –М.: Бином, 2010. – 704 с.
29. Professional Microsoft SQL Server 2014 Administration / A. Jorgensen, B. Ball, S. Wort, R. LoForte. – Indianapolis: John Wiley & Sons, 2014. – 936 с.
30. Офіційний сайт Google Analytics. – Режим доступу: http://www.google.com/analytics/. – Дата доступу : 25.03.2020