МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Кваліфікаційна робота бакалавра

ІНТЕРАКТИВНИЙ ВЕБ ДОДАТОК ДЛЯ ІНТЕРНЕТ ГАЛЕРЕЇ "Shuraz_art"

Здобувач освіти гр. ІН – 82 Олег ШОКУН

Науковий керівник,
кандидат фізико-математичних наук,
асистент кафедри комп'ютерних наук Ольга ШУТИЛЄВА

Завідувач кафедри
доктор технічних наук, професор Анатолій ДОВБИШ

СУМИ 2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Затверджую_____
Зав. кафедрою Довбиш А.С.
"_____" _____2022 р.

ЗАВДАННЯ

до кваліфікаційної роботи

здобувача вищої освіти четвертого курсу, групи ІН-82 спеціальності «122 – Комп'ютерні науки» денної форми навчання Шокуна Олега Ігоровича.

Тема: **«ІНТЕРАКТИВНИЙ ВЕБ ДОДАТОК ДЛЯ ІНТЕРНЕТ ГАЛЕРЕЇ "Shuraz_art"**

Затверджена наказом по СумДУ
№_____ від _____2022 р.

Зміст пояснювальної записки: 1) літературний огляд за обраною тематикою роботи; 2) постановка завдання для розробки, вибір оптимальних інструментів для розробки веб додатку; 3) практична реалізація.

Дата видачі завдання «_____»_____ 2022 р.
Керівник роботи _____Ольга ШУТИЛЄВА
Завдання прийняв до виконання _____ Олег ШОКУН

РЕФЕРАТ

Записка: 43 стор., 27 рис., 2 таблиці, 1 додаток, 20 джерел.

Об’єкт дослідження – веб додаток та способи його реалізації.

Мета роботи – Створення інтерактивного веб додатку для інтернет-галереї.

Методи дослідження – методи збору та аналізу даних.

Результати – Було створено сучасний Web-додаток з використанням web технологій React та Laravel, та інших сучасних методик таких як БЕМ.

WEB-ДОДАТОК, WEB, JAVASCRIPT,
ФРЕЙМВОРК, WEB-ПРОГРАМИ

ЗМІСТ

ВСТУП

Актуальність роботи. На даний момент сучасний світ складно уявити без використання Інтернету, зв'язку та інформаційних технологій в цілому, це свідчить про те, що бізнес та компанії розширюють свої межі, та залучаючи нову аудиторію та технології. Натомість користувач отримує більше джерел нової інформації, товарів, послуг та інших можливостей.

Щодня люди користується Інтернетом та різноманітними гаджетами, які допомагають швидко та зручно здійснювати пошук потрібної інформації, виконувати свою роботу та багато чого іншого, застосовуючи веб та мобільні додатки. З розвитком швидкості Інтернету збільшуються можливості та навантаження на системи, та але кількість бажаючих з кожним днем також неухильно збільшується попит на інструменти віддаленого доступу та обслуговування. Для підприємств на даний момент наявність веб-сайту або мобільного додатку стає невід'ємною частиною розвитку компанії, та надає великі важливості для подальшого розвитку та еволюції компанії або бізнесу . Веб-сайт став своєрідним посередником, так званим інтерфейсом, між підприємством та клієнтами.

Швидкий розвиток веб-технологій стрімко поширюється у всіх сферах життя людини, і майже найбільший вплив має на сферу надання послуг. Можливість виконувати замовлення, і навіть не підніматись зі свого робочого місця керувати своєю домівкою. Це все торкається різних сферах обслуговування, доставки, віддаленої роботи, та майже всіх сфер нашого життя, це наглядно було продемонстровано під час карантину та локдауну.

Метою кваліфікаційної роботи є створення інтерактивного веб-додатку онлайн галереї із застосуванням сучасних інструментів розробки та наповнення його відповідним контентом.

1. ЛІТЕРАТУРНИЙ ОГЛЯД

1.1 Поняття про Web 1.0

Сайт або веб-сайт – це центральне розташування веб-сторінок, які пов’язані та доступні для відвідування домашньої сторінки за допомогою браузера. Наприклад, URL-адреса веб-сайту університету МІХ.СумДУ – це https://mix.sumdu.edu.ua. З домашньої сторінки можна отримати доступ до будь-якої з веб-сторінок, що містяться на ньому.

Саме визначення Web 1.0 [1] узвичаїлося, як не дивно, після появи Web 2.0 [2]. Порівняння показало, що Інтернет став зовсім іншим тим самим можна виділити особливості та відмінності минулої «версії». Весь етап першої ітерації Мережі продовжився з 1991 до 2004 року. Web 1.0 можна описати однією фразою – Read-Only (тільки читання). Основні концепції, що користувачі мали можливість лише переглядати сторінки та взаємодіяти з контентом. В Інтернеті ще не були розвинені можливості участі користувачів у створенні контенту, вони лише споживали те, що на веб-ресурсах. Жодних авторизацій, трекерів та реєстрацій.

Дані сайтів зберігалися на серверах у файлових системах і часто видавалися у тому вигляді, в якому вони були. Ця особливість змушувала веб-майстрів при додаванні нових сторінок заново верстати ті, що вже були, задля додавання посилань. Для вирівнювання контенту застосовувалися таблиці, на сайтах не було адаптивності і часто вказувалося рекомендований дозвіл, у якому вся інформація відкривалася б коректно. Також не кожен сайт підтримувався всіма браузерами, тому веб-майстри розміщували бейджі з логотипами тих веб-браузерів, які правильно працювали з ресурсом.

Дизайн сайтів теж був дуже далекий від нинішньої різноманітності та інтерактивності. Багато сайтів нехтували HTML-розміткою і видавали користувачам «сирий» текст. CSS-стилі були непопулярними серед розробників. На сторінках переважали яскраві кольори та матеріальні

текстури, що копіюють дерево, каміння або метал. Для динамічності на сайтах розміщували GIF-анімації, які примітивно оживляли контент.

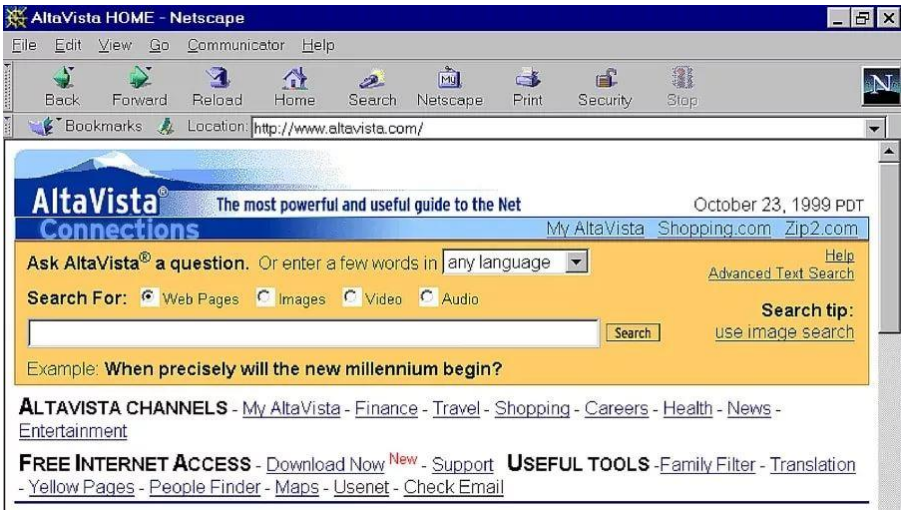

Рисунок 1.1 – Типовий сайт часів Web 1.0

На заході епохи Web 1.0 стали з'являтися форуми та чати, які дозволяли користувачам самим брати участь у формуванні контенту. Але при цьому компанія Amazon з моменту відкриття свого сайту дала можливість клієнтам залишати відгуки на товари. Певною мірою корпорація випередила час.

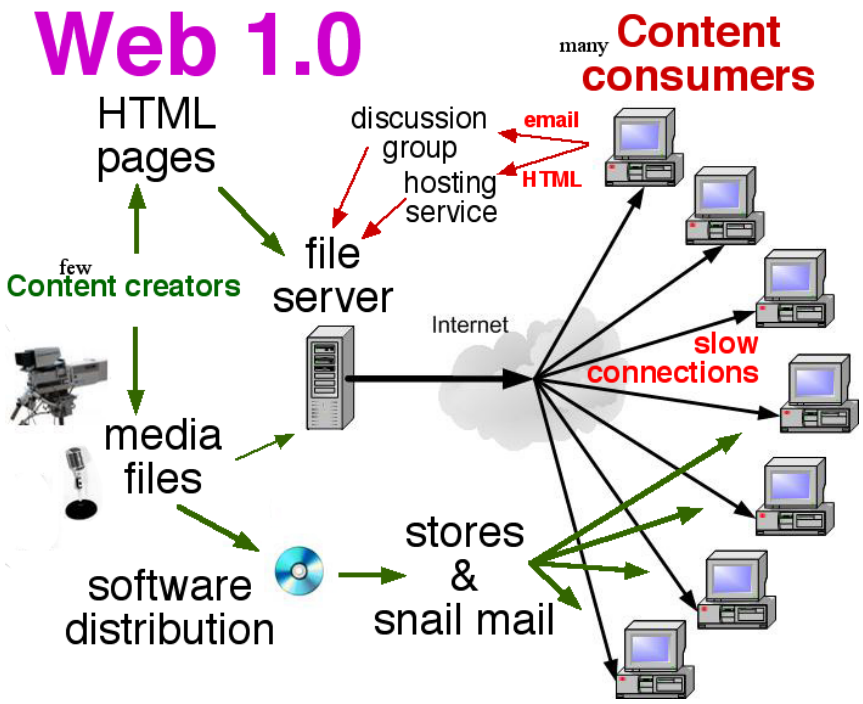

Рисунок 1.2 – Принцип роботи Web 1.0

1.2 Поняття Web 2.0

У 2005 році американський видавець та активіст руху за вільне програмне забезпечення Тім О'Райлі (Tim O'Reilly) опублікував статтю «What Is Web 2.0» [2]. У матеріалі О'Райлі зазначив, що в мережі починає з'являтися все більше сайтів, об'єднаних ідеями та єдиним принципом. У статті він чітко розділив Web 1.0 та Web 2.0 та намітив вектор розвитку. Саме за допомогою цієї статті досі визначають ключові засади «нового покоління».

Вважається, що епоха Web 2.0 почалася в 2004 році і продовжується до сьогодні. Тепер у справу включилися великі корпорації та користувачі. Корпорації почали будувати онлайн-імперії, а користувачам дозволили брати участь у створенні контенту. Web 2.0 працює за принципом Read/Write Web (читання/запис у Мережі).

В Інтернеті з'явилась можливість авторизації та створити обліковий запис практично на кожному сайті. Користувачі почали добровільно залишати свої дані та давати згоду на збір даних в обмін на зручності та можливості користування ресурсами. Компанії отримали можливість заробляти на даних, продаючи їх рекламним агентствам, а деякі відкрили власні, що допомогло повністю зосередити прибуток у своїх руках.

У Web 2.0 з'явилися соціальні функції: дедалі більше ресурсів дозволяє користувачам спілкуватися між собою, обмінюватися повідомленнями та здійснювати дзвінки. До соціалізації можна віднести і персоналізацію: користувачі можуть оформляти власні профілі, додавати на сторінки фотографії та записи, розміщувати відеоролики та статті. Користувачі публікують матеріали, отримують реакції та оцінки від інших користувачів у вигляді вподобайок та коментарів. Також варто зазначити, що сайти почали вводити системи рейтингу – Карму чи Репутацію.

Зміни очікувано вплинули і на дизайн. Зовнішній вигляд сайтів став приємнішим, стали переважати округлі форми, прості точно підібрані кольори, дизайнери почали звертати увагу не лише на зовнішній вигляд, але й

на зручність. З'явилися складні анімації і пішла епоха GIF. Незважаючи на це, фахівці зазначають, що коли у кожного з'явилася можливість створювати сайти, то в світ прийшли шаблони і це призвело до одноманітності. В цілому відзначається, що в Web 2.0 переважають патерни, що вбивають індивідуальність та оригінальність сайтів. Текст на сайтах почав виділятися в міру значущості: з’явилися заголовки, підзаголовки, різні шрифти, підкреслення та виділення. А сайти стали адаптивними: один і той самий веб-портал можна відкрити як на десктопі, так і на смартфоні.

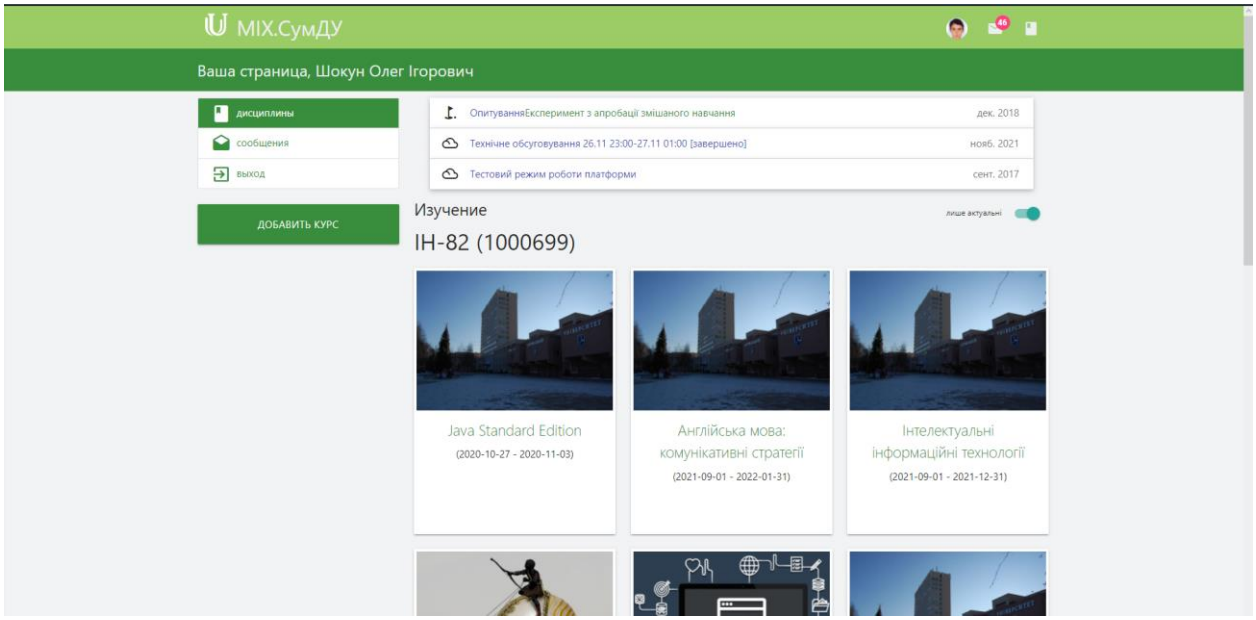

Рисунок 1.3 – МІХ.СумДУ, типовий представник епохи Web 2.0

Рисунок 1.4 – Принцип роботи Web 2.0

Для зручності користувачів з’явилися веб-служби, дані почали передаватися у форматах JSON [3] або XML, більшість операцій перейшли на сервери компаній. Використання AJAX (Asynchronous JavaScript and XML) дозволило не перезавантажуватися сторінкам щоразу, а асинхронно завантажувати ті дані, які необхідні користувачеві.

Алгоритми епохи Web 2.0 враховують безліч факторів і працюють практично скрізь, починаючи із соціальних мереж та закінчуючи маркетплейсами. Саме тому, якщо два користувача відкриють, наприклад, головну сторінку YouTube, то вміст на ній кардинально відрізнятиметься. У цьому видно ще одну відмінність Web 2.0 від першої «версії» мережі, коли ту саму сторінку переглядали тисячі користувачів.

Враховуючи зростання популярності соціальних мереж і різноманітність контенту, компанії стали уважніше ставитися до інформації, що публікується. На платформах з’явилися суворі правила та модерація. Користувачі не контролюють контент, що публікується, а компанії мають право видаляти ті дані, які порушують правила.

1.3 Поняття Web 3.0

Основні концепції Web 3.0 позначив керівник компанії Netscape Джейсон Калаканіс (Jason Calacanis). Web 3.0 знаходиться на ранній стадії, тому поки що доступні тільки первинні уявлення про технологію. Тим не менш, Калаканіс опублікував своє бачення майбутнього ще в 2007 році і вважає, що на основі Web 2.0 має з'явитися новий простір, який вирішує основні проблеми.

Основною проблемою керівник вважає знецінення ресурсів та сервісів: відносна простота створення сайтів викликала одноманітності. Тим самим О'Райлі підтримав свого часу ідеї Калаканіса, а також зазначив, що Web 3.0 має вийти за межі звичного розуміння мережі та почати «взаємодіяти з фізичним світом».

Визначальні характеристики Web 3.0:

- децентралізація: дані більше не зберігатимуться на єдиних серверах, а розподіляться між користувачами. Необхідні обчислення переїдуть із датацентрів на ноутбуки, смартфони та «розумні» гаджети користувачів. В даний час є технології, що дозволяють досягти цього, але немає єдиного рішення про те, яка з них лежатиме в основі «нового Інтернету»;
- штучний інтелект та машинне навчання: інтелектуальні алгоритми не зникнуть із мережі, а ще більше продовжуватимуть допомагати користувачам шукати необхідний контент. Деякі дослідники відзначають, що в майбутньому ШІ можна буде використовувати для виявлення рекомендованих коментарів на маркетплейсах, що допоможе створити прозоріші сервіси.;
- відкритість: програмне забезпечення буде переважно з відкритим вихідним кодом, що дозволить досконально розуміти, як влаштовані інструменти і яким чином вони взаємодіють із користувачем;
- свобода: очікується, що цензура в мережі буде скасована, і кожен матиме можливість публікувати будь-який контент, роль модерації візьме на себе суспільство, а не корпорації;
- розповсюдження: фахівці припускають, що в епоху Web 3.0 будуть практично в будь-якому місці IoT-пристрої та «розумні» гаджети;
- семантична павутина: машина погано розуміє запити природною мовою і все ще часто помиляється. Для покращення цього процесу планують використовувати технологію семантичного павутиння, коли з мережі можна отримати інформацію.

Крім ключових відмінностей, фахівці повідомляють про інші можливі зміни інші зміни. Може відбутися зміна методу авторизації в сервісах на універсальний, який буде ключем до всіх ресурсів в Мережі. Щось подібне реалізовано зараз: за допомогою облікового запису Google або Facebook можна увійти на практично будь-який сайт, але у випадку Web 3.0 єдиний обліковий запис може стати і гаманцем, і банківським додатком.

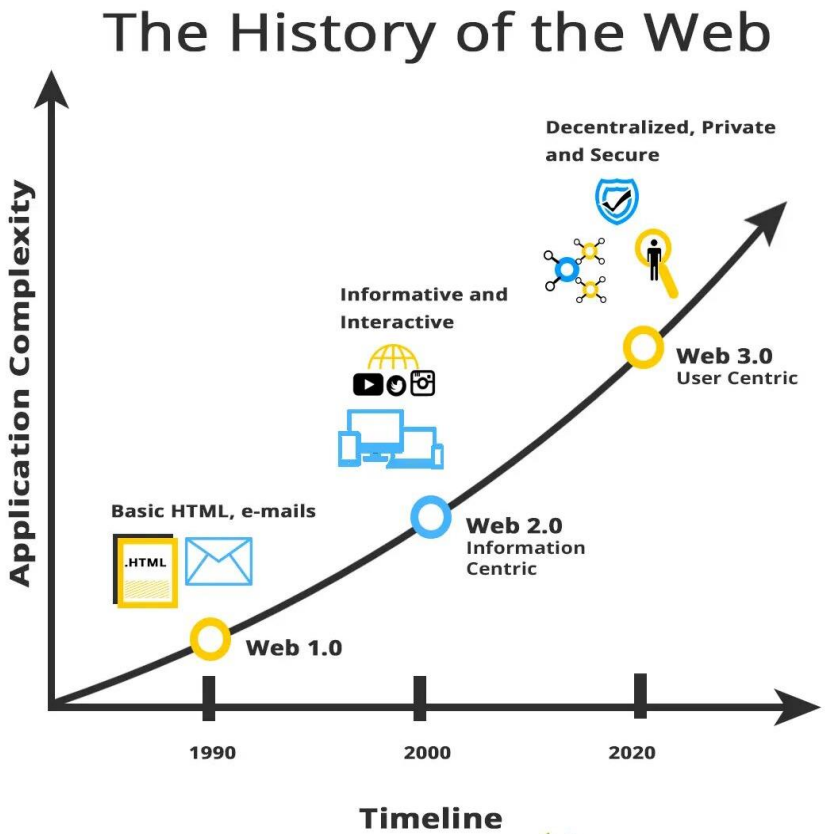

Рисунок 1.5 – Еволюція від web 1 до web 3

Зараз ідеї Web 3.0 все ще залишаються ідеями та викликають більше питань, ніж розуміння. Крім того, концепцію часто просувають криптоінвестори та NFT-ентузіасти [4], які зводять все до нового виду заробітку.

1.3.1 Різниця між веб-сайтом та веб-сторінкою

Веб-сайт відноситься до центрального розташування з більш ніж однією веб-сторінкою або кількома веб-сторінками. Наприклад, Computer Hope вважається веб-сайтом, який містить тисячі різних веб-сторінок.

Рисунок 1.6 – URL-адреси веб-сайту

У наведеному вище прикладі URL-адреси веб-сайт – computerhope.com, а веб-сторінка – "url.htm". Веб-сторінка не потребує розширення файлу, як- от .htm або .html , щоб бути веб-сторінкою. Багато сайтів призначені для показу сторінки за замовчуванням у каталозі (наприклад, index.html) або налаштовані без розширень файлів.

1.3.2 Типи веб-сайтів

Сьогодні в Інтернеті існують мільярди веб-сайтів, які можна умовно розділити на наступні категорії, оскільки певні веб-сайти можуть мати поєднання одразу кількох категорій. Наприклад, веб-сайт також може бути форумом, веб-поштою або пошуковою системою:

- Архівний сайт;
- Блог (веблог);
- Бізнес-сайт та корпоративний веб-сайт;
- Веб-сайт спільноти;
- Сайт знайомств;
- Веб-сайт електронної комерції;
- Освітній сайт;
- Ігровий сайт;
- Урядовий сайт;
- Веб-сайт допомоги та запитань;
- Шкідливий веб-сайт;
- Сайт обміну медіа;
- Веб-сайт новин;
- Персональний сайт;
- Портал;
- Сайт пошукової системи;
- Веб-сайт соціальної мережі, тощо.

1.4 Способи створення веб-сайту

1.4.1 Використання конструкторів веб-сайтів

Конструктор веб-сайтів – це інструмент, який дозволяє дуже швидко створити веб-сайт. Ці рішення зазвичай мають функціональні можливості перетягування, включають шаблони для дизайну і не вимагають знання кодування. Одним із прикладів є WP Website Builder, [5] який поставляється безкоштовно з усіма планами на DreamHost.

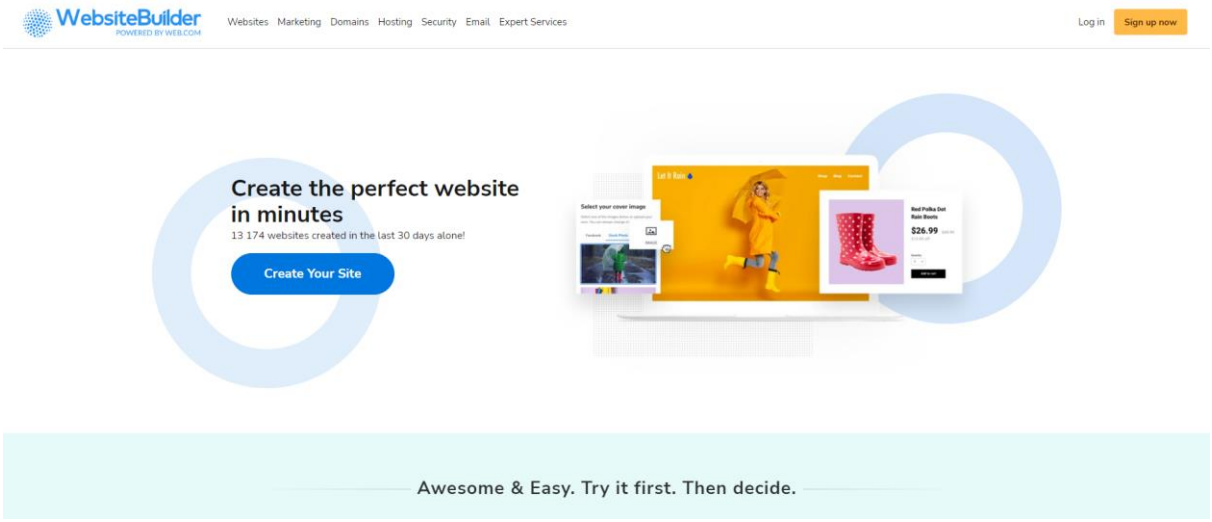

Рисунок 1.7 – WP Website Builder

Цей конструктор дозволяє вибирати з багатьох попередньо стилізованих блоків вмісту. Наприклад, можна вибрати блок контактної інформації, спеціально оформлений для веб-сайтів ресторанів. Потім налаштувати його за допомогою власних кольорів, шрифтів, зображень тощо.

Конструктори веб-сайтів можуть заощадити час і гроші, дозволяючи швидко запустити новий сайт. Крім того, за допомогою правильного конструктора можливо налаштувати елементи дизайну сайту, таким чином щоб вони відповідали бренду.

У той же час, варто зазначити, що будь-який конструктор веб-сайтів за обмежений і може не надавати такого контролю над зовнішнім виглядом і функціональністю сайту.

1.4.2 Систему керування вмістом (CMS)

Системи керування вмістом (CMS)[6] плутають із конструкторами веб-сайтів. Однак CMS – це повна платформа для створення та публікації цифрового контенту. Вони набагато складніші та гнучкіші, ніж конструктор веб-сайтів. На сьогоднішній день найпопулярнішою CMS є WordPress з часткою ринку понад 60%.

Рисунок 1.8 – WordPress

CMS побудована на потужному програмному забезпеченні баз даних. Це означає, що є можливість зберігати вміст та інші ресурси високоорганізованим способом, відображати та впорядковувати вміст унікальним чином і навіть включати користувацький код. На ринку також є інші варіанти CMS, зокрема:

- Joomla [7] – це безкоштовна CMS з відкритим кодом. Незважаючи на те, що вона схожа на WordPress, Joomla не настільки зручна для користувачів і має більш круту криву навчання.
- Drupal – є ще одним варіантом з відкритим кодом і навіть складнішим, ніж Joomla. Ця платформа, орієнтована на досвідчених веб-розробників.
- Magento – це продукт Adobe, який спеціально обслуговує веб-сайти електронної комерції та їх потреби. Існує також версія з відкритим кодом.

CMS пропонують велику гнучкість і можуть використовуватися для створення веб-сайтів практично будь-якого типу. Якщо розробляти дизайн

веб-сайту за допомогою HTML і CSS, потрібно зібрати разом деякі інструменти, зокрема:

- Редактор коду, наприклад Visual Studio та PhpStorm.
- Програма протоколу передачі файлів (FTP) [8] – це спосіб передачі файлів HTML на веб-сервер і з нього. Якщо у редакторі коду немає вбудованої цієї функції, то можна скористатися безкоштовним варіантом, FileZilla або преміальним додаткомт CuteFTP.
- Локальний веб-сервер надає змогу створювати та експериментувати на локальній машині, перш ніж запускати сайт в Інтернет.

1.4 Постановка задачі

Для того щоб досягти мети кваліфікаційної роботи необхідно буде виконати наступні етапи розробки:

- проаналізувати предметну область;
- обрати необхідні актуальні інструменти для розробки веб-сайту;
- виділити переваги і недоліки;
- створити карту сайту;
- створити додаток;
- заповнити відповідним контентом.

2. МЕТОДИКА ВИРІШЕННЯ ПОСТАВЛЕНИХ ЗАДАЧ

2.1 Аналіз технологій розробки та вибір засобів реалізації проекту

Процес створення веб-сайту є довгим та складним процесом, який проходить через різні етапи, і до кожного з цих етапів потрібні відповідні інструменти та/або технології розробки. Структура створення сайту виглядає наступним чином:

- Створення технічного завдання, опис функціоналу, який має містити веб-сайт ;
- Збір семантичного ядра і аналіз структури сайту ;
- Розробка ієрархії структури ;
- Створення URL [9] структури по визначеній ієрархії і з ключовими словами;
- Навігація сайту в HTML, CSS, верстання.
- Налаштування внутрішніх посилань.
- Наповнення контентом і тестування.
- Оптимізація структури сайту.

2.1.1 Мови програмування веб-розробці

1. JavaScript [10] – це легка, інтерпретована мова сценаріїв, яка широко використовується для покращення та додавання вмісту на веб-сторінку. Використовується різними способами, від додавання простих функцій або кнопок до створення складних анімацій. JavaScript працює поверх вбудованого механізму браузера і додає функціональність за допомогою простих команд, розміщених у тегах <script> у HTML-документі.

Якщо на комп’ютері встановлено JavaScript, то можна вносити зміни до будь-якої частини веб-сайту та створювати абсолютно нові сторінки з нуля. Більшість користувачів Інтернету вже ввімкнули JavaScript, що робить її одним із найбільш затребуваних варіантів серед дизайнерів.

За даними Failory, приблизно 79% відсотків усіх інтернет-серферів завжди мають увімкнений JavaScript. Варто врахувати, що старіші версії можуть не працювати належним чином через зміни браузера або сторонні програми, які заважають його виконанню. Тому встановлення оновлень допоможе мінімізувати проблеми з сумісністю.

Переваги використання мови JavaScript для проекту:

- Сприяє швидкому розвитку проекту
- Зменшує час компіляції
- Висока сумісність
- Підвищена швидкість
- Більш короткі коди
- Розширена функціональність

Недоліки використання мови JavaScript:

- Вимагає, щоб розробники запускали код на кількох платформах, призначених для забезпечення нульових технічних збоїв.
- Засоби налагодження потребують удосконалення.

2. HTML/CSS [11] не є мовами програмування, але забезпечать основу для розробки веб-сайтів, дає початок для написання коду, який буде робити щось, крім відображення інформації на екрані. І знання про створення базових сторінок може стати в нагоді, якщо необхідно публікувати вміст сайту в Інтернеті десь пізніше. Використання CSS дозволяє дизайнерам відокремлювати стиль від вмісту.

Це полегшує роботу кількох людей з різними частинами веб-сайту, оскільки вони можуть змінювати стилі, не порушуючи випадково інші компоненти. По суті, CSS надає способи форматування текстових елементів на сторінці, таких як заголовки або абзаци, за допомогою попередньо визначених шрифтів і кольорів. HTML і CSS разом користуються популярністю 56% серед розробників у всьому світі.

3. Python – універсальна мова, яку можна використовувати у будь-якій кількості галузей і для будь-якої кількості цілей: від веб-розробки до штучного

інтелекту і до науки про великі дані. Оскільки код Python настільки простий, то навіть новачки програмісти можуть писати чистий код, без помилок.

Переваги використання мови Python:

- Безкоштовне використання ;
- Підтримує безліч бібліотек ;
- Підвищена продуктивність;
- Висока читабельність коду;
- Пропонує розробникам портативність.

Недоліки використання мови Python для проекту:

- Високе споживання пам'яті;
- Помилки під час виконання;

2.2.1 SQL

Якщо необхідно зробити веб-проект, що включає науку про дані, або працювати з масивними наборами даних, SQL є очевидним вибором. Це завдяки базам даних запитів і можливостям аналізу даних. Це схоже на вивчення розмовної мови – єдиний спосіб вчитися – це робити. Більш просунуті версії SQL, такі як MySQL і PostgreSQL, також широко використовуються в науці даних, тому на них також можна зосередитися.

Плюси використання мови SQL для проекту:

- Дуже інтерактивний
- Швидка обробка запитів
- Хороша документація
- Простий синтаксис
- Висока портативність

Мінуси використання мови SQL для проекту:

- Недружній інтерфейс
- Конкретні версії не є економічно вигідними.

2.2 Фреймворки у веб-розробці

Розглянемо найбільш популярні фреймворки.

2.2.1 Express

Завдяки стрімкому росту популярності Node.js [12], Express швидко стає одним із найпопулярніших найкращих фреймворків для веб-розробки. Він популярний серед Accenture, IBM і Uber, а також багатьох інших компаній, а також сумісний з іншими фреймворками, такими як Kraken, Sails і Loopback. Express надає деякі основні функціональні можливості фреймворку, не приховуючи особливостей Node, і використовує надійну продуктивність асинхронного Node.js. Він досить гнучкий і підтримує повноцінні програми, а також REST API. Можливо, найбільшим недоліком Express є той факт, що немає визначеного способу виконання завдань, принаймні для початківців.

2.2.2 Django

Django – це фреймворк Model-View-Template, [13] який використовує Python для веб-розробки. Цю структуру використовують великі імена, такі як Google, Youtube та Instagram. Django має купу функцій, таких як аутентифікація та обмін повідомленнями. Він відповідає шаблону конфігурації конвенції та шаблону DRY. Django надає розробникам методи та інструменти для створення безпечного веб-сайту або реалізує функції безпеки у самій платформі, наприклад запобігання виконання коду на рівні шаблону.

2.2.3 Rails

Rails – це фреймворк Model-View-Controller, який використовує Ruby, і це популярний фреймворк, який використовують багато розробників. Airbnb, GitHub, Hulu і Shopify є основними користувачами. Rails вважається фреймворком, зручним для початківців, і той факт, що плюси і мінуси обговорюються, допомагає новачкам досить швидко розпочати веб-розробку.

Існує багато корисних розширень, подібних до бібліотеки, які розширюють функціональні можливості програми та допомагають розвиватися швидше й ефективніше.

Основний недолік розширень полягає в тому, що для їх розгортання та роботи у робочому середовищі потрібно чимало зусиль.

2.2.4 Laravel

Laravel [14] – це фреймворк Model-View-Controller, який використовує PHP, який є однією з найпопулярніших мов Інтернету. Це відносно новий в порівнянні з іншими фреймворками в цьому списку.

Laravel поставляється з підтримкою API з коробки, а також має пристойну кількість пакетів, які можуть розширити його функціональність. Laracasts – це веб-сайт з навчальними посібниками з перегляду екрана, що містить понад тисячу відео про PHP, Laravel та технології інтерфейсу в екосистемі Laravel, які можна вважати бонусом для початківців. Однак з точки зору продуктивності Laravel не порівнюється з Django або Express, що може бути недоліком для масштабних проектів.

2.2.5 Spring

Spring – це фреймворк Model-View-Controller, який використовує Java, популярну мову всіх часів. Користувачами цього фреймворку є такі веб-сайти, як Wix, TicketMaster і BillGuard. У Spring є багато допоміжних проектів, які підвищують його ефективність і дозволяють швидко масштабувати свій бізнес. Той факт, що в ньому використовується Java, строго типізована мова, є серйозним плюсом для багатьох веб-розробників. Крива навчання може бути досить крутою, особливо якщо не знати Java.

2.3 Frontend Javascript Frameworks

2.3.1 Angular

Angular [15] – це інтерфейсний фреймворк, який спеціалізується на створенні багатьох односторінкових додатків. Це яскравий фреймворк, здатний створювати повноцінні програми на стороні клієнта. Angular може використовувати Javascript, але пізніші випуски прийняли Typescript, який є наднабором Javascript. Основними недоліками Angular є його розмір порівняно з іншими фреймворками, а також той факт, що він за своєю природою не оптимізований для SEO, хоча його можна оптимізувати для SEO. Google розробив angular , і його використовують Google, Microsoft і Paypal .

2.3.2 React

React – це не фреймворк, це фронтенд-бібліотека, але багато розробників вважають його фреймворком, і його зазвичай порівнюють у цьому контексті. React був першим, хто прийняв архітектуру на основі компонентів, яку пізніше почали використовувати Angular [16] і Vue [17], а також багато інших фреймворків. Віртуальний dom React робить маніпуляцію domom набагато швидшим, і його досить легко підібрати, в основному завдяки синтаксису JSX. React можна використовувати на стороні сервера або на стороні клієнта. Він був розроблений і підтримується Facebook, а Facebook і Instagram використовують його.

2.3.3 Vue

Vue.js – один з найпопулярніших фреймворків JS. Є багато цікавих речей про Vue. По-перше, це прогресивна структура, що означає, що якщо є існуючий проект, то можна застосувати Vue для однієї частини проекту, і все буде працювати нормально. По-друге, він також надає архітектуру компонентів для гри, а екосистема Vue може допомогти створити повні інтерфейсні програми. Деякі люди обережно ставляться до використання Vue, оскільки великі компанії, такі як Facebook або Google, не підтримують його, але це швидко змінюється.

2.3.4 Ember

Ember був визнаний найкращим фреймворком Javascript ще у 2015 році. Сьогодні спільнота Ember величезна, і вона постійно розширюється, з новими функціями та випусками, які регулярно додаються. Ember володіє двостороннім зв'язуванням даних, яким може похвалитися Angular, і має безліч функцій і компонентів, які можна використовувати «з коробки». Google, Microsoft, Heroku і Netflix часто використовують цей фреймворк. Ember орієнтується на продуктивність розробника і намагається максимізувати її, усуваючи потребу витрачати час, або застосовуючи деякі найкращі методи JS у своєму базовому дизайні.

2.3.5 Backbone

Backbone – це надзвичайно легкий інтерфейсний фреймворк, який підходить для створення багатих односторінкових програм. Він відповідає шаблону MV* і частково реалізує дизайн MVC. Backbone має лише одну основну залежність, якою є бібліотека Underscore, і дозволяє створювати повні програми на стороні клієнта.

Таблиця 2.1 – Порівняння серверних фреймфорків

Backend Frameworks	Плюси	Мінуси
Ruby on Rails	• Бажано для створення прототипів • Прості процеси автоматизації тестування • Швидка розробка додатків • Фреймворк MVC і доступ до різноманітних бібліотек та інструментів	• Відсутність відповідних цитат • Погані способи завантаження • Не бажано для великих додатків
ExpressJS	• Надає пакети для створення API • Гнучкий • Простий	• Обструктивні повідомлення про помилку • Не бажано для складних додатків

Laravel	• Цикл швидкого будівництва • Добре задокументовані деталі • Узгодження щодо конфігурації та фреймворку MVC • Велика спільнота веб-розробників на GitHub • Забезпечує схему пакування із спеціальним менеджером	• Розробники можуть зіткнутися з проблемою під час виконання • Нова платформа для розробки веб-додатків • Зворотна маршрутизація стає дуже важкою
Spring	• Гнучка природа • Хороший вибір для програм Java • Найпростіший варіант співпраці з різними іншими програмами	• Може бути трохи незбалансованим • Складний для навчання
Django	• Безпечний • Чудовий вибір для MVP Забезпечує гнучкість і масштабованість • Добре документована колекція	• Монолітний • Може бути не найшвидшим

Таблиця 2.2 – Порівняльна таблиця інтерфейсів веб-фреймворків

Frontend Frameworks	Плюси	Мінуси
React	• Великий набір інструментів • Встановлено фреймворк і компоненти MVC • Створений комп'ютером DOM для покращеного перегляду • Односпрямований кодовий потік для стабільного коду	• Менш простий, ніж JavaScript • Немає чіткого документа
Flutter	• Один з найкращих варіантів для MVP • Підтримує кросплатформну розробку • Потрібно менше коду	• Підтримує лише мобільні додатки • Містить обмежені бібліотеки • Пропонує менше підтримки програм Android TV і Apple TV
Ember	• Швидкий розвиток	• Має повільну рендеринг

	• Використання аксесуарів для високої продуктивності • Має власний налагоджувач • Фреймворк MVC і добре задокументований	• Не бажано для невеликих проектів • Складний для навчання
Vue.js	• Легко виявити помилки • Швидкий підхід до будівництва • Легко засвоюється з іншими програмами • Чітко задокументовано	• Не містить стабільних компонентів
Angular	• Переважно для створення прототипів і MVP • Швидкий підхід до розвитку • Найкраще для створення односторінкових програм • Надає підтримку Typescript для створення величезних програм	• Може виникнути проблеми із завантаженням величезної кількості даних • Відсутність документації CLI • Важко навчатися

У даному веб додатку було використано такі технології як REACT та SIMFONY – аналог LARAVEL.

2.4 Карта сайту

Карта сайту або XML [19]-карта сайту – це список різних сторінок веб-сайту. XML – це скорочення від «розширюваної мови розмітки», яка є способом відображення інформації на сайті.

Пошукові системи, такі як Google, прагнуть відображати найрелевантніші результати людям за будь-яким пошуковим запитом. Щоб зробити це ефективно, вони використовують сканери сайтів для читання, упорядкування та індексації інформації в Інтернеті.

Карти сайту XML полегшують роботам пошукових систем читати вміст сайту та відповідно індексувати сторінки. Вона повідомить пошуковим системам розташування сторінки на веб-сайті, час її оновлення, частоту оновлення та важливість сторінки, оскільки вона пов’язана з іншими

сторінками вашого сайту. Без належної карти сайту боти Google можуть подумати, що на вашому сайті є дубльований вміст, що насправді зашкодить вашому рейтингу SEO [20].

Розроблювана карта для нашого веб-сайту має вигляд:

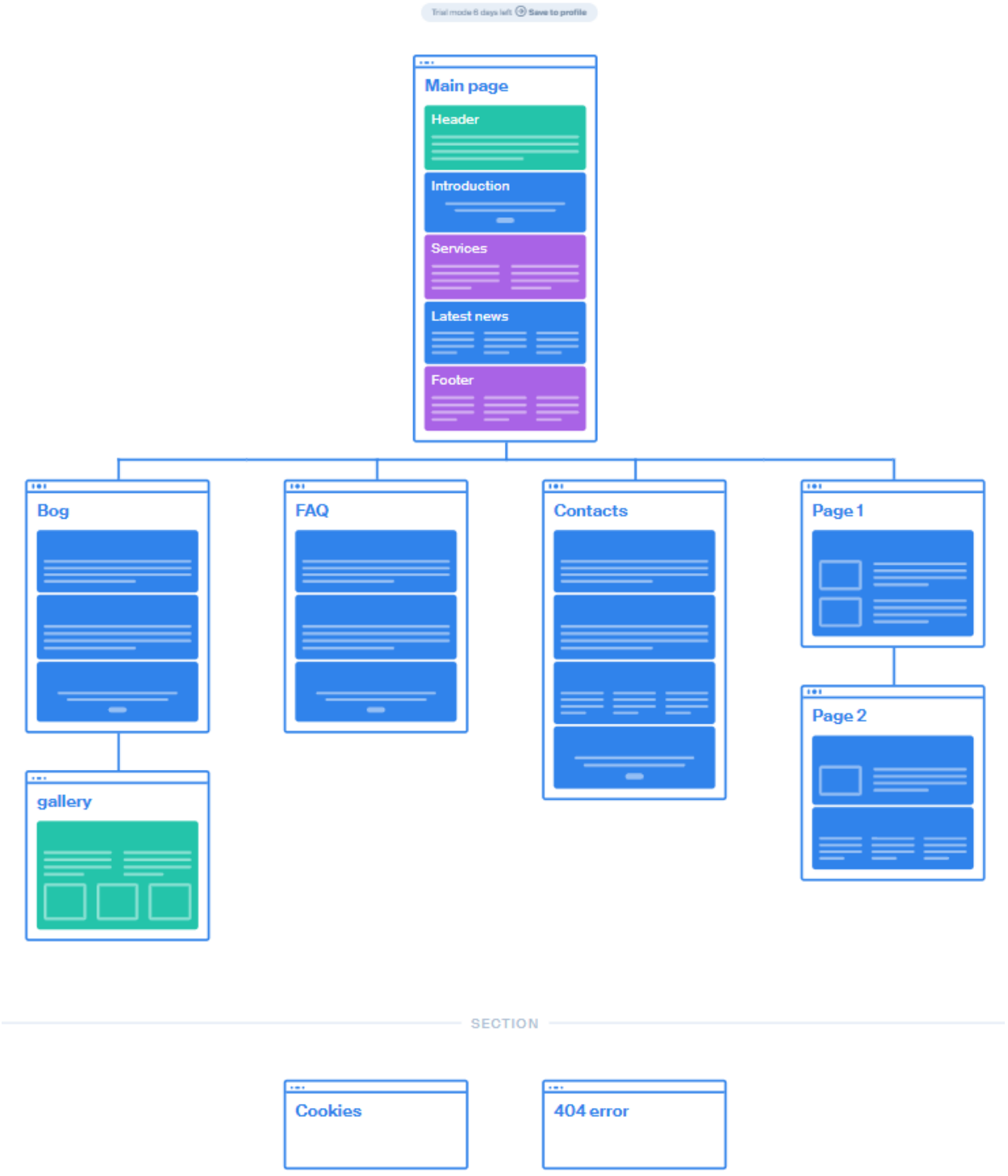

Рисунок 2.1 – Зображення карти сайту

3. ПРАКТИЧНА РЕАЛІЗАЦІЯ

У процесі створення веб додатку використовуємо програму Visual Studio Code, яку попередньо налаштували під проект. Після налаштування редактора коду починаємо створювати додаток, за картою сайту яку створили раніше (рис.2.1). Попередньо створюємо проект для сайту за всіма правилами і розділяєм на теки за потребою.

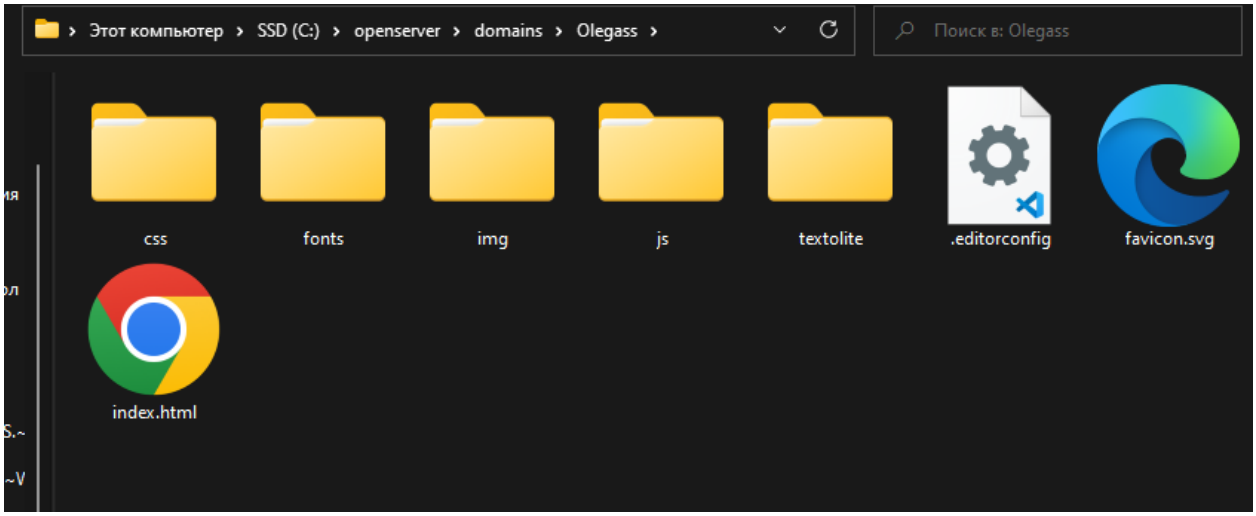

Рисунок 3.1 – Тека з проектом

Починаємо створювати веб додаток з головного меню навігації, для цього створюємо меню та додаємо всі якірні посилання, додаємо логотип та закінчуємо створювати header на фоновому інтерактивному зображенні.

```
https://aka.ms/powershell
Type 'help' to get help.

   A new PowerShell stable release is available: v7.2.4
   Upgrade now, or check out the release page at:
     https://aka.ms/PowerShell-Release?tag=v7.2.4

PS C:\OpenServer\domains\Olegass> npx create-react-app my-app
>> cd my-app
>> npm start
```

Рисунок 3.2 – Початок завантаження та налаштування REACT

За допомогою команд завантажуємо всі потрібні файли для REACT.

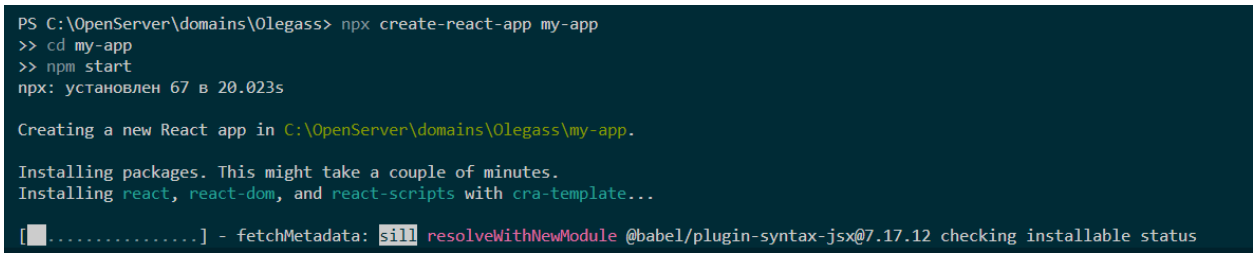

Рисунок 3.3 – Процес встановлення REACT

Після воду команд на спостерігаємо процес встановлення REACT.

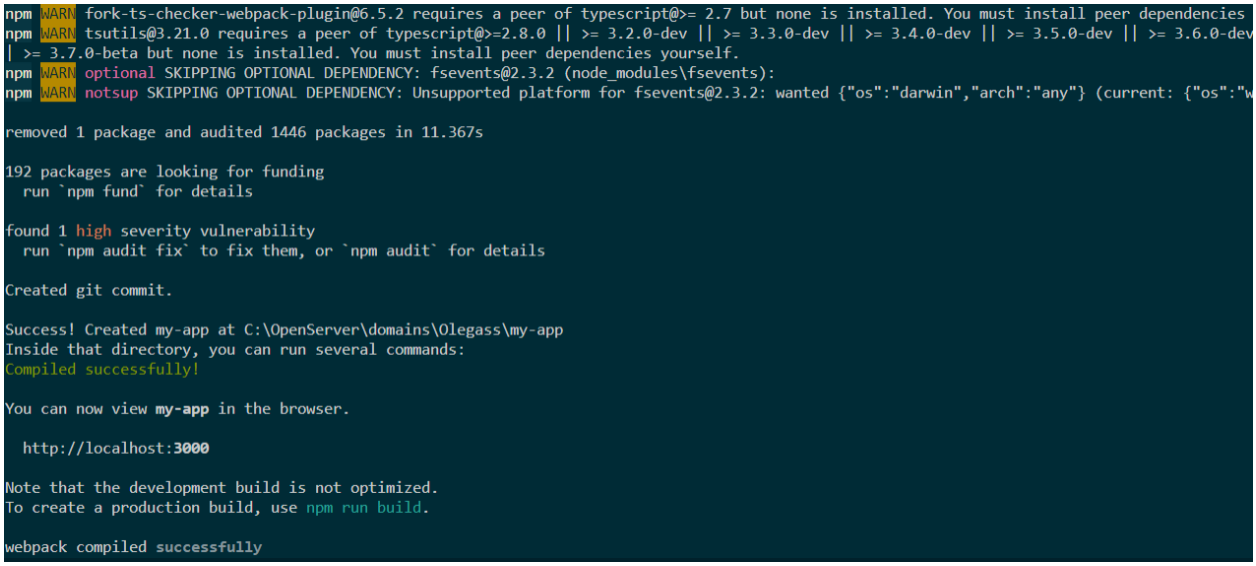

Рисунок 3.4 – Відкриття локального хосту

Після завантажееня файлів відбувається відкриття локального хосту.

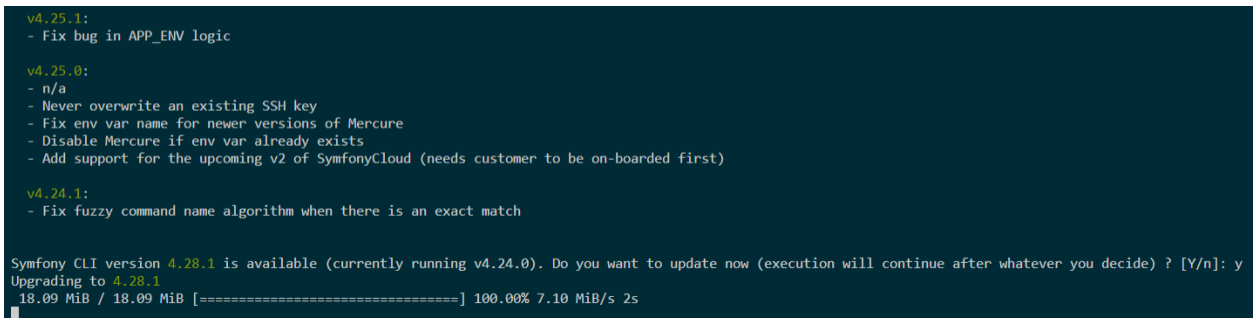

Рисунок 3.5 – Завантаження Symfony

Після завантаження REACT також заватажуємо скелет проекту Symfony, за токим принципом як і реакт, тільки пропередньо налаштувавши проєкт.

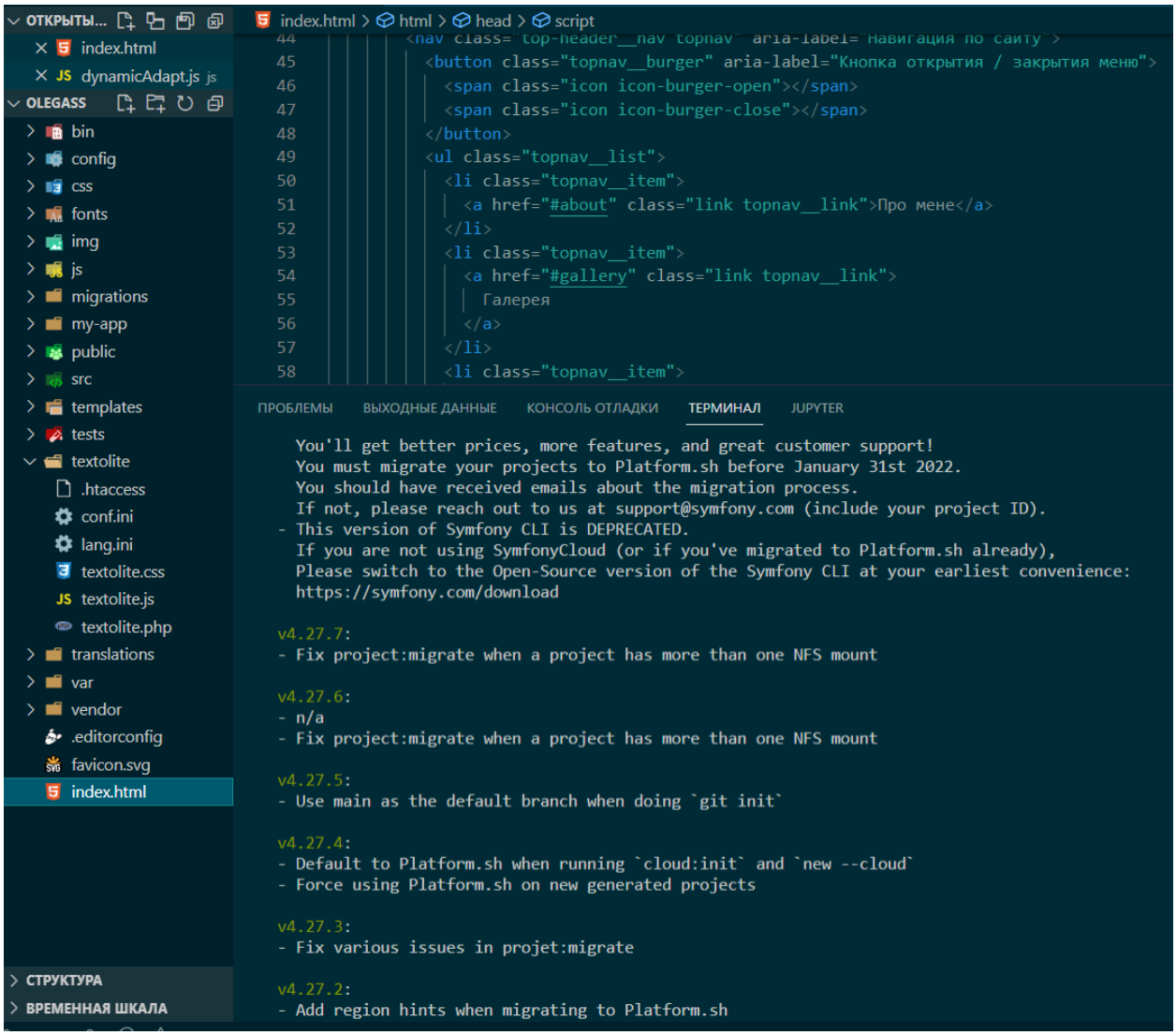

Рисунок 3.6 – Структура проєкту

Після звантаження всіх потрібних файлів можемо побачити структуру проекту з скелетом Symfony

.

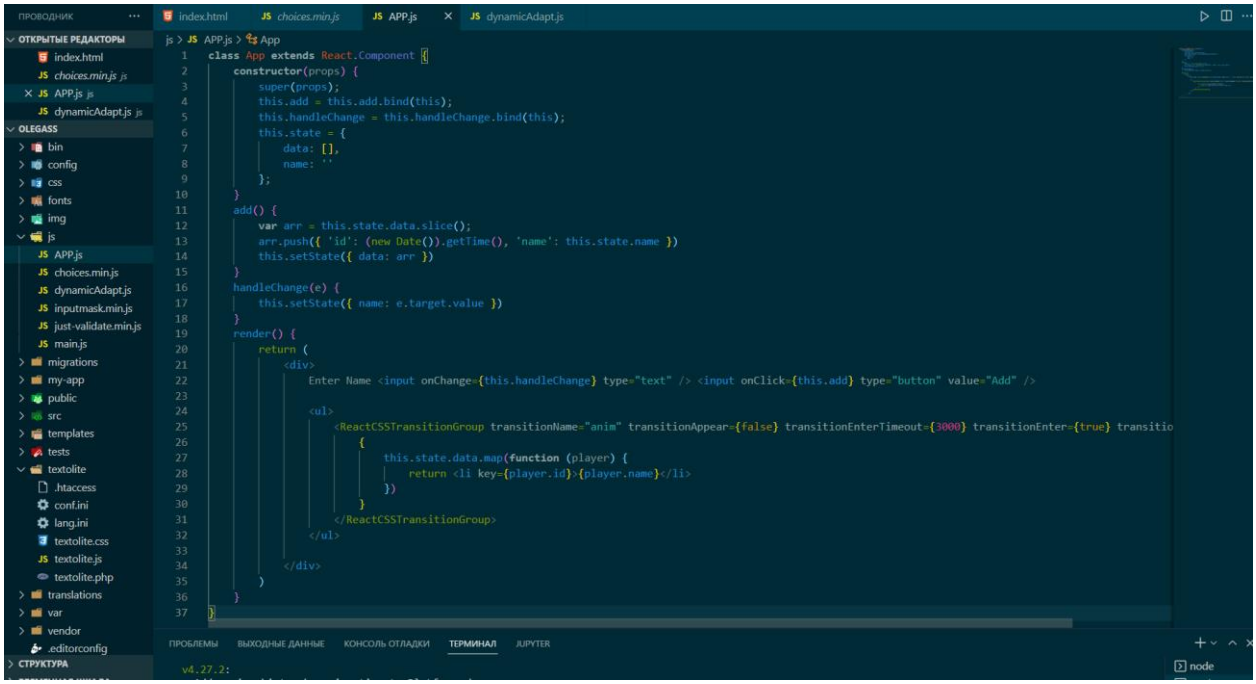

Рисунок 3.7 – Код анімації

Додаємо в структура проекту потрібні нам JS файли та додаємо анімації на іншу логіку на REACT.

Рисунок 3.8 – Header веб додатку

Далі створюємо дві секції:

1. «Про мене» – детально описуємо інформацію про автора даної галереї.

2. «Галерея» – створюємо саму галерею з інтерактивним пошуком та оглядом зображень.

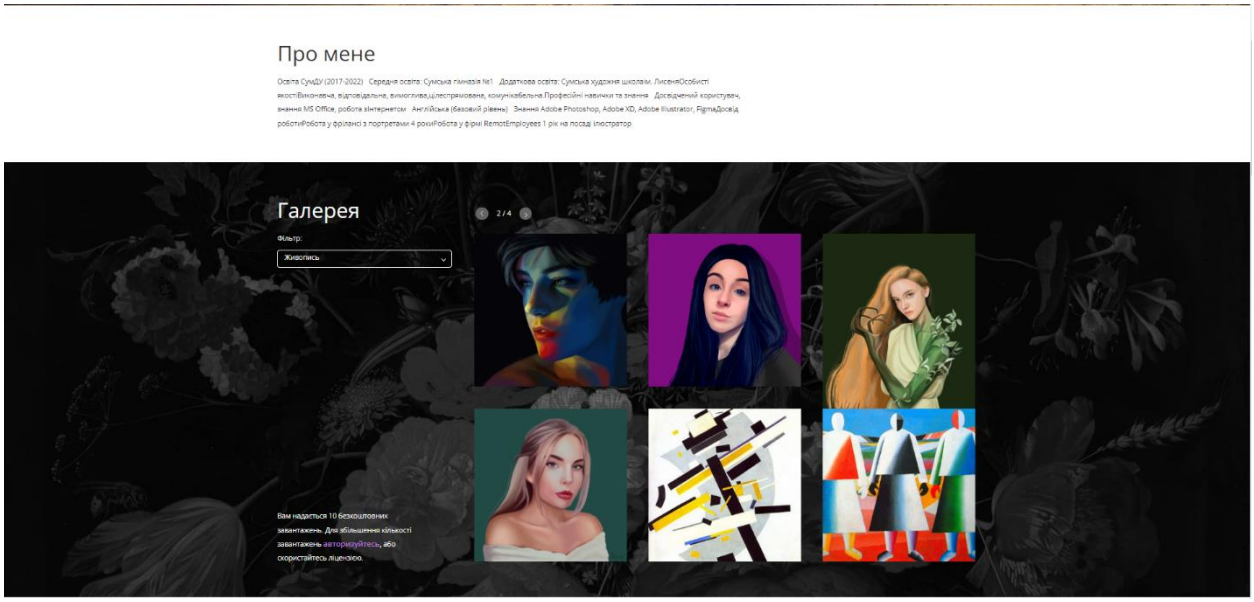

Рисунок 3.9 – Секції з контентом сайту

Потім створюємо додаткову секцію інтерактивної галереї, з широким оглядом робіт та датою їх опублікування з тиражом в різних країнах світу, які зображені на інтерактивному каталозі. Маємо детальний опис роботи з характером персонажу, який на ній зображений, в даному випадку та обширним каталогом всіх робіт.(рис. 3.10)

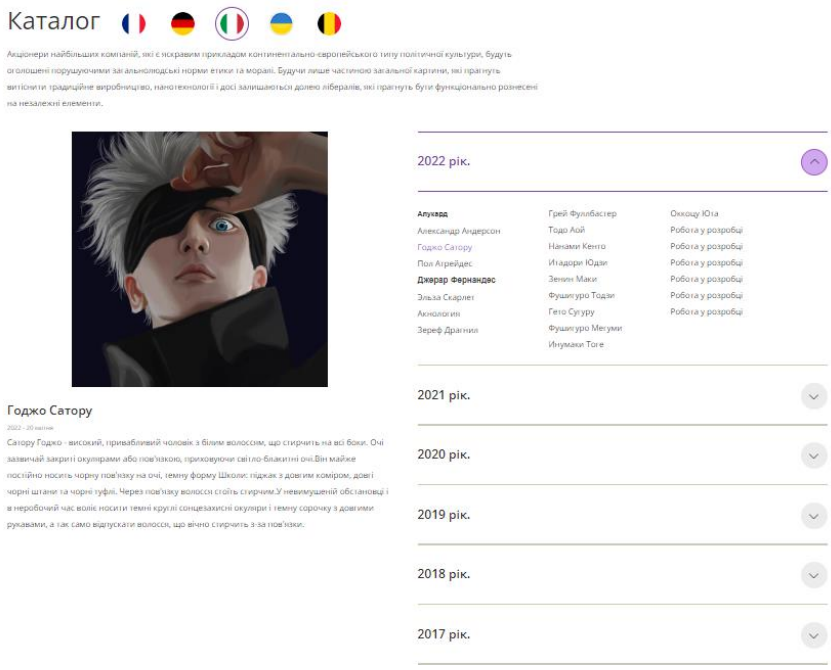

Рисунок 3.10 – Інтерактивна секція каталогу робіт

Наступна секція була створена, щоб читач веб додатку дізнавався основні події в художньому світі та крокував в ногу з сучасним мистецтвом.

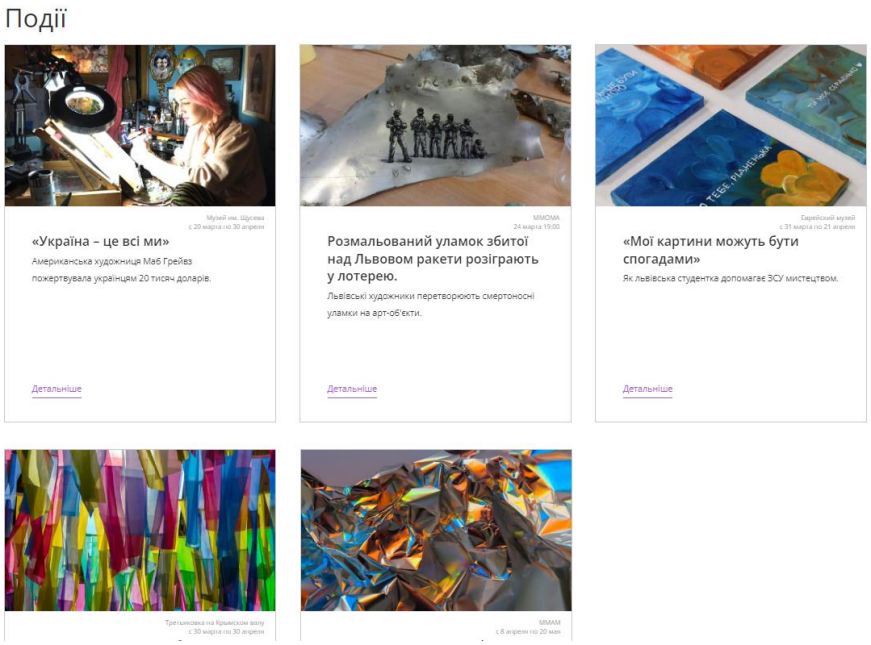

Рисунок 3.11 – Секція з новинами

Дана секція зроблена з метою придбання художньої літератури та робіт автора. Нижче буде наведений приклад міні – магазину.

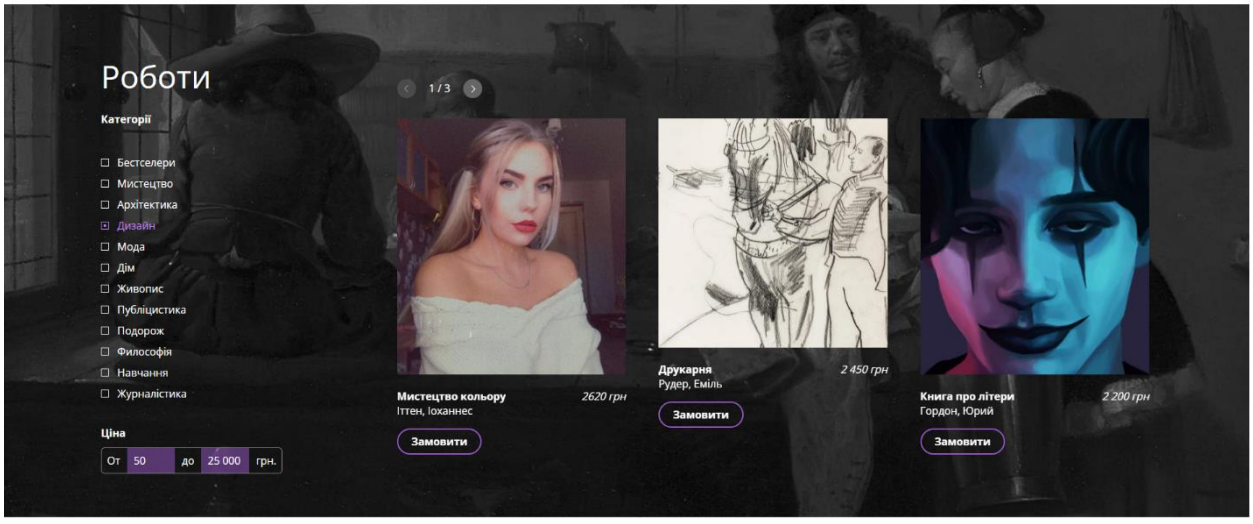

Рисунок 3.12 – Магазин робіт

Створення «підвалу» з контактами автора по яким користувач може зв’язатися, уточнити потрібні параметри та замовити роботу.

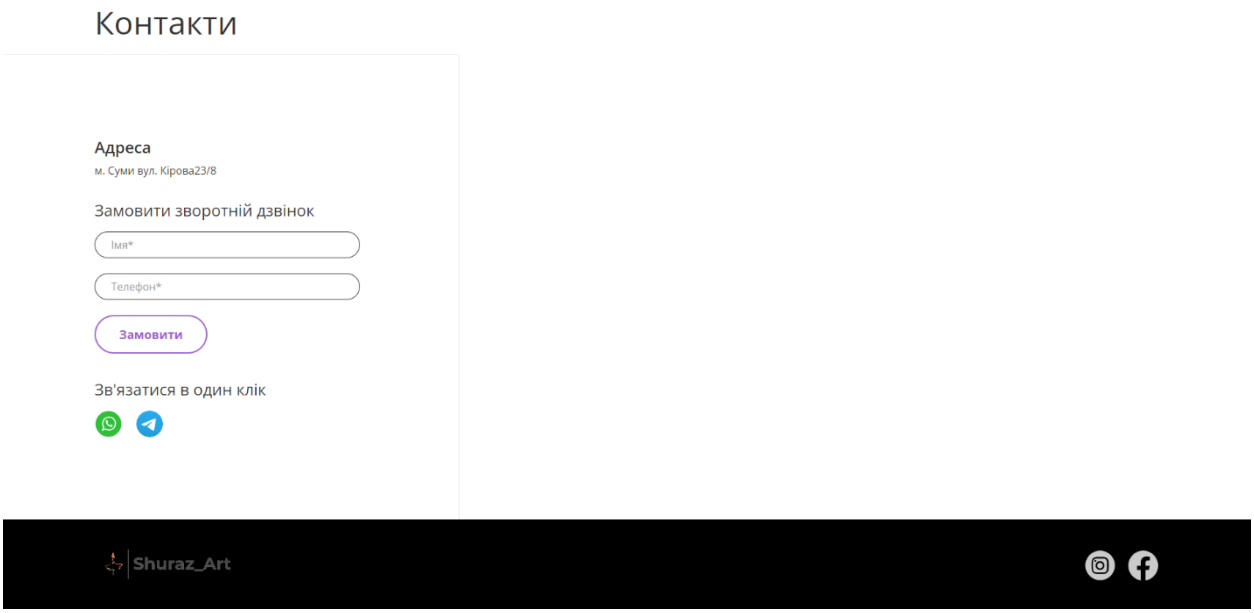

Рисунок 3.13 – Контакти

Також додаємо функції редагування та адміністрації контенту веб додатку за допомогою панелі адміністратора

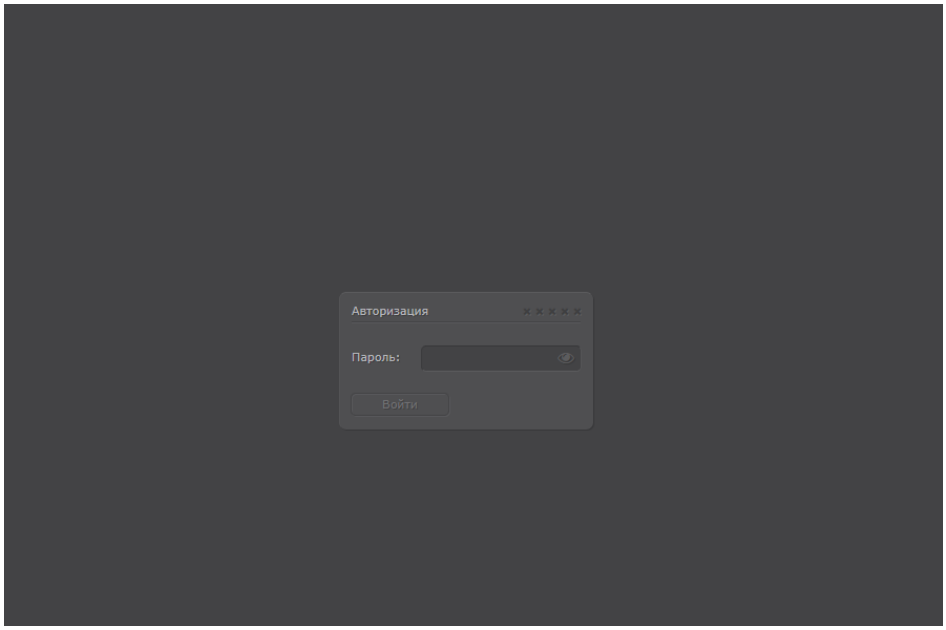

Рисунок 3.14 – Авторизація для входу в адмін панель

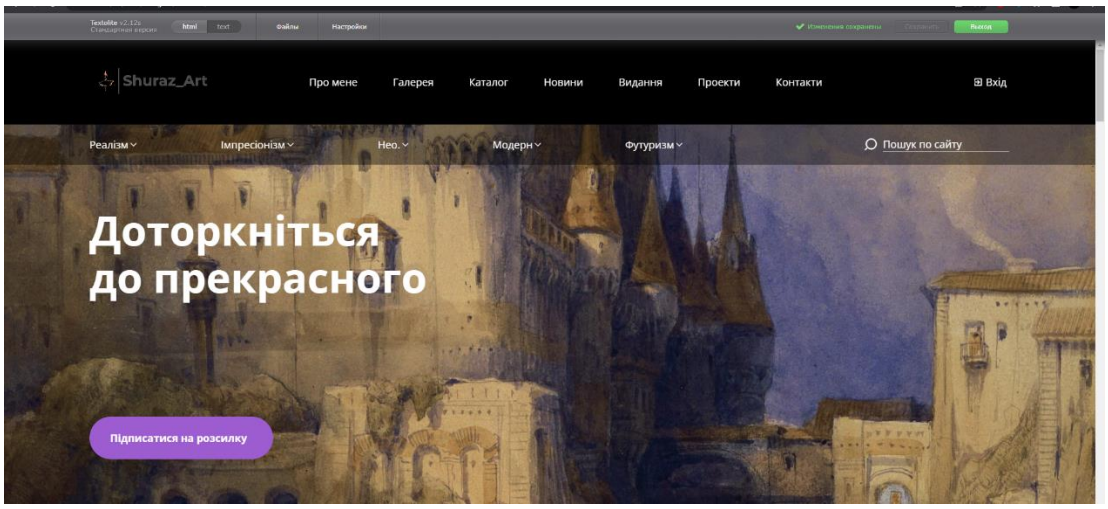

Рисунок 3.15 – Вигдяд адмін панелі

Про мене

Освіта СумДУ (2017-2022) Середня освіта: Сумііііііська гімназія №1 Додаткова освіта: Сумська художня школаім. ЛисеняОсобисті якостіВиконавча, відповідальна, вимоглива,цілеспрямована, комунікабельна.Професійні навички та знання Досвідчений користувач, знання MS Office, робота зІнтернетом Англійська (базовий рівень) Знання Adobe Photoshop, Adobe XD, Adobe Illustrator, FigmaДосвід роботиРобота у фрілансі з портретами 4 рокиРобота у фірмі RemotEmployees 1 рік на посаді ілюстратор

Рисунок 3.16 – Принцип роботи адмін панелі (редагування тексту)

ВИСНОВКИ

У даній кваліфікаційній роботі були розглянуті сучасні WEB технології, найпопулярніші веб-фреймворки та мови програмування. Також був створений детальний опис всіх актуальних способів створення сайтів, їх порівняння, переваги та недоліки. Була створена перша версія карти сайту, яка може змінюватись по мірі розробки та додавання нових функцій.

У ході практичної реалізації проєкту було створено та мануально протестовано веб-додаток інтернет галерії, який повністю відповідає поставленим задачам для розробки.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. web-creator.ru [Електронний ресурс]. – Режим доступу: https://web-creator.ru/articles/about_frameworks
2. habr.com [Електронний ресурс]. – Режим доступу: https://habr.com/ru/post/464417/
3. web-creator.ru [Електронний ресурс]. – Режим доступу: https://web-creator.ru/articles/yii
4. jetbrains.com [Електронний ресурс]. – Режим доступу: https://www.jetbrains.com/phpstorm/
5. CMS List. Огляд cms. Сайт про системи управління сайтом. [Електронний ресурс] Режим доступу: http://www.cmslist.ru
6. MySQL 4 - Строковые функции [Електронний ресурс] Режим доступу: http://www.codenet.ru/db/mysql/mystring4
7. Web-Розробка [Електронний ресурс] Режим доступу: http://fcit.tneu.org/web-rozrobka/
8. Види тестування ПО [Електронний ресурс]. Режим доступу: http://qalearning.com.ua/theory/lectures/material/testing-types-functional/
9. Перенесення файлів з localhost на сервер [Електронний ресурс]. Режим доступу: http://joomlaportal.ru/faq/installation-and-update/51-perenos-sajta-slocalhost-na-server
10. Веб Database Application with PHP and MYSQL», 2nd Edition By David Lane, Hugh E. Williams. © O'Reilly, May 2004. ISBN: 0-596-00543-1.
11. Markus Egger — MVVM Survival Guide for Enterprise Architectures in Silverlight and WPF [Електронний ресурс]. — 2012. — Режим доступу: https://www.packtpub.com/application-development/mvvm-survival-guide-enterprisearchitectures-silverlight-and-wpf.
12. Кейт Джонс. DOM Scripting: Web Design with JavaScript and the Document Object Model. / К. Джонс — Перше, 2005. — 368 с.

13. Anthony Gore — Full-Stack Vue.js 2 and Laravel 5 [Електронний ресурс]. —2015. — Режим доступу: https://bit.ly/2OEODzR.
14. Martin Fowler — GUI Architectures. Часть 1 [Електронний ресурс]. — 2009.— Режим доступу: https://bit.ly/2CvCk1e.
15. Vue.js Material Component Framework — Vuetify.js [Електронний ресурс]. —2016. — Режим доступу: https://vuetifyjs.com.
16. Дмитрий Котеров, Алексей Костарев PHP. В подлиннике. / Д. Котеров, А.Костарев — Спб.: «БХВ-Петербург», 2005. — 1120 с.
17. Колисниченко Д. Н. Самоучитель PHP 5 / Д. Н. Колисниченко — СПб.: Наука и Техника, 2007. — 640 с.
18. Кузнецов Максим, Симдянов Игорь PHP 5/6. / М. Кузнецов, И. Симдянов —Спб.: «БХВ-Петербург», 2009. — 1024 с.
19. Л. Аткінсон, З. Сураскін. PHP5. Бібліотека професіоналу. / Л. Аткінсон, З. Сураскін — М. : «Вільямс», 2006 — 543 с.
20. Скотт Хокінс. Адміністрування веб-сервера Apache і керівництво по електронній комерції. / С. Хокінс. — М.: «Вільямс», 2001. — 336 с.

ДОДАТОК А

```
"use strict";

function DynamicAdapt(type) {
  this.type = type;
}

DynamicAdapt.prototype.init = function () {
  const _this = this;
  // масив об’єктів
  this.objects = [];
  this.daClassname = "_dynamic_adapt_";
  // масив DOM-елементів
  this.nodes = document.querySelectorAll("[data-da]");

  // заповнення objects об’єктами
  for (let i = 0; i < this.nodes.length; i++) {
    const node = this.nodes[i];
    const data = node.dataset.da.trim();
    const dataArray = data.split(",");
    const object = {};
    object.element = node;
    object.parent = node.parentNode;
    object.destination = document.querySelector(dataArray[0].trim());
    object.breakpoint = dataArray[1] ? dataArray[1].trim() : "767";
    object.place = dataArray[2] ? dataArray[2].trim() : "last";
    object.index = this.indexInParent(object.parent, object.element);
    this.objects.push(object);
  }

  this.arraySort(this.objects);

  // масив унікальних медіа-запитів
  this.mediaQueries = Array.prototype.map.call(this.objects, function (item) {
    return '(' + this.type + "-width: " + item.breakpoint + "px)," + item.breakpoint;
  }, this);
  this.mediaQueries = Array.prototype.filter.call(this.mediaQueries, function (item,
index, self) {
    return Array.prototype.indexOf.call(self, item) === index;
  });

  // виклик обробника при першому запуску
  for (let i = 0; i < this.mediaQueries.length; i++) {
    const media = this.mediaQueries[i];
    const mediaSplit = String.prototype.split.call(media, ',');
```

```
    const matchMedia = window.matchMedia(mediaSplit[0]);
    const mediaBreakpoint = mediaSplit[1];

    // масив об’єктів з оптимальним breakpoint
    const objectsFilter = Array.prototype.filter.call(this.objects, function (item) {
      return item.breakpoint === mediaBreakpoint;
    });
    matchMedia.addListener(function () {
      _this.mediaHandler(matchMedia, objectsFilter);
    });
    this.mediaHandler(matchMedia, objectsFilter);
  }
};

DynamicAdapt.prototype.mediaHandler = function (matchMedia, objects) {
  if (matchMedia.matches) {
    for (let i = 0; i < objects.length; i++) {
      const object = objects[i];
      object.index = this.indexInParent(object.parent, object.element);
      this.moveTo(object.place, object.element, object.destination);
    }
  } else {
    for (let i = 0; i < objects.length; i++) {
      const object = objects[i];
      if (object.element.classList.contains(this.daClassname)) {
        this.moveBack(object.parent, object.element, object.index);
      }
    }
  }
};

// Функція переміщення
DynamicAdapt.prototype.moveTo = function (place, element, destination) {
  element.classList.add(this.daClassname);
  if (place === 'last'  place >= destination.children.length) {
    destination.insertAdjacentElement('beforeend', element);
    return;
  }
  if (place === 'first') {
    destination.insertAdjacentElement('afterbegin', element);
    return;
  }
  destination.children[place].insertAdjacentElement('beforebegin', element);
}
```

```
// Функція повернення
DynamicAdapt.prototype.moveBack = function (parent, element, index) {
  element.classList.remove(this.daClassname);
  if (parent.children[index] !== undefined) {
    parent.children[index].insertAdjacentElement('beforebegin', element);
  } else {
    parent.insertAdjacentElement('beforeend', element);
  }
}

// Функція отримання індексу всередині предка
DynamicAdapt.prototype.indexInParent = function (parent, element) {
  const array = Array.prototype.slice.call(parent.children);
  return Array.prototype.indexOf.call(array, element);
};

// Функція сортування масиву за breakpoint та place
// за збільшенням для this.type = min
// за зменшенням для this.type = max
DynamicAdapt.prototype.arraySort = function (arr) {
  if (this.type === "min") {
    Array.prototype.sort.call(arr, function (a, b) {
      if (a.breakpoint === b.breakpoint) {
        if (a.place === b.place) {
          return 0;
        }

        if (a.place === "first"  b.place === "last") {
          return -1;
        }

if (a.place === "last"  b.place === "first") {
          return 1;
        }

        return a.place - b.place;
      }

      return a.breakpoint - b.breakpoint;
    });
  } else {
    Array.prototype.sort.call(arr, function (a, b) {
      if (a.breakpoint === b.breakpoint) {
        if (a.place === b.place) {
          return 0;
```

```
        }

        if (a.place === "first"  b.place === "last") {
          return 1;
        }

        if (a.place === "last" || b.place === "first") {
          return -1;
        }
       return b.place - a.place;
       }
       return b.breakpoint - a.breakpoint;
     });
     return;
   }
};

const da = new DynamicAdapt("max");
da.init();
```