МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ
КАФЕДРА ПРОГРАМУВАННЯ ТА МАТЕМАТИКИ

Пояснювальна записка

до дипломної роботи

_____бакалавр _____

(освітньо-кваліфікаційний рівень)

на тему «Розробка веб-сайту інтернет-провайдера»

Виконав: студент 4 курсу, групи ІПЗ-17д
напряму підготовки 121 „Інженерія програмного забезпечення”
_____ Айвас А.Ю.
(підпис)

Керівник,
проф, д.т.н._____ Марченко Д.М.
(підпис)

Рецензент,
доцент. каф. ПМ,
к.ф.-м.н. _____ Захожай О.І.
(підпис)

СЄВЄРОДОНЕЦЬК – 2021

ЛИСТ ПОГОДЖЕННЯ І ОЦІНЮВАННЯ
бакалаврської роботи студента гр. ІПЗ-17д Айваса А.Ю.

Науковий керівник
проф, д.т.н._____Марченко Д.М.
ПІБ, посада

Оцінка наукового керівника: _____

Рецензент Захожай О. І., к.ф.-м.н., проф. каф. ПМ СНУ ім. В. Даля
ПІБ, місто роботи, посада

Оцінка рецензента: _____

Кінцева оцінка за результатами захисту: _____

Голова ЕК,
проф. кафедри ПМ
к.т.н., доцент _____ Захожай О.І.
підпис

СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет інформаційних технологій та електроніки

Кафедра програмування та математики

Освітньо-кваліфікаційний рівень бакалавр

Напрям підготовки 121 „Інженерія програмного забезпечення”

ЗАТВЕРДЖУЮ
Завідувач кафедри ПМ,
д.т.н., доцент
_____ Лифар В.О.
«___» _____ 2021 р.

З А В Д А Н Н Я
НА ДИПЛОМНУ РОБОТУ СТУДЕНТА
Айваса Артура Юрійовича

1. Тема роботи Розробка веб-сайту інтернет-провайдера.

керівник роботи д.т.н., проф. Марченко Дмитро Миколайович

затверджені наказом вищого навчального закладу від «12» квітня 2021 року № 68/14.04

2. Строк подання студентом роботи 11 червня 2021 р.

3. Вихідні дані до роботи

Об'єктом досліджень є клієнт-серверні технології на основі веб-додатку навчальної платформи.

3.1 Літературні джерела

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

4.1 Вступ

4.2 Аналіз предметної галузі (огляд літератури), з висвітленням

наступних питань:

Поняття навчальної платформи.

Аналіз існуючих навчальних платформ.

Вимоги до навчальних платформ.

4.3 Основна частина, в якої висвітлити:

Визначення веб-сайту.

Архітектура клієнт-серверної взаємодії.

Вимоги до веб-сайту.

Розробка веб-сайту.

4.4 Висновки

4.4 Перелік використаних джерел

5. Перелік графічного матеріалу немає

6. Дата видачі завдання 23 квітня 2021 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Одержання завдання на виконання роботи	23.04.21	
2	Укладання і погодження з керівником плану і етапів виконання роботи	23.04.21	
3	Узагальнення даних літературних джерел, укладання розділу «Аналіз предметної галузі»	26.04.21	
4	Аналіз шляхів виконання завдання. Вибір і погодження з керівником оптимального шляху	03.05.21	
5	Проектування інфологічної моделі задачі що реалізується.	11.05.21	
6	Укладання програмного продукту	12.05.21	
7	Укладання, оформлення та погодження пояснювальної записки з керівником	25.05.21	
8	Здача готової пояснювальної записки на кафедру	11.06.21	
9	Укладання доповіді і презентації	14.06.21	

Студент _____ Айвас А.Ю.
(підпис)

Керівник роботи _____ Марченко Д. М.
(підпис)

РЕФЕРАТ

Робота містить: 52 сторінки основного тексту, 13 сторінок додатків, 23 рисунка, 1 таблицю, 17 використаних джерел.

Об'єктом дослідження є можливість розробки і створення web-сайту.

Метою дипломної роботи є розробка веб-сайту для Інтернет-провайдера "MegatronLink".

Дипломна робота складається з трьох основних частин.

У першій частині розглядаються сучасні технології, що використовуються при розробці веб-сайтів, та використовувані методи. Розглянуто теоретичні аспекти розробки веб-сайтів та їх використання в інформаційних цілях. Вивчаються основні типи веб-сайтів.

Також розглядаються основні етапи розробки веб-сайтів та особливості, які необхідно враховувати при розробці дизайну веб-сайту.

У цій статті розглядається програмне забезпечення, яке буде потрібно для розробки веб-сайту.

Друга частина показує модель проекту та структуру веб-сайту.

У третій частині статті описується практична частина створення веб-сайту, а саме створення прототипів, створення макета та HTML-макета веб-сайту.

Система впроваджена відповідно до всього технічного персоналу.

Результатом цієї роботи є повноцінний веб-сайт компанії «MegatronLink», наповнений необхідним контентом.

ПЕРЕЛІК ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

БД – база даних.

HTML – HyperText Markup Language, мова гіпертекстової розмітки.

XML – Extensible Markup Language, розширювана мова розмітки.

URL - Uniform Resource Locator, уніфікований покажчик інформаційного ресурсу.

API – Application Programming Interface, інтерфейс прикладного програмування.

HTTP – HyperText Transfer Protocol, протокол передачі HyperText.

IP – Internet Protocol, міжмережевий протокол

JS(JavaScript) – динамічна, об’єктно-орієнтована мова програмування.

CSS(Cascading Style Sheets) – Таблиці Каскадних Стилів.

DNS(Domain Name System) – служба імен доменів, організовує групи комп'ютерів в Інтернет за допомогою ієрархії доменів.

PHP – PHP є мовою сценаріїв, виконуваних на стороні сервера, володіє простотою і надійністю.

ЗМІСТ

ВСТУП

Актуальність досліджень.

В останні роки в Інтернеті спостерігаються суттєві зміни. Сайти, які раніше були платформою для розташування статичного вмісту, тепер стали багатофункціональними інтерактивними системами для надання різноманітної інформації.

Актуальність теми дослідження зумовлена необхідністю аналізу існуючих методів розвитку корпоративних та інформаційних ресурсів з метою виявлення найбільш ефективного методу їх створення. Метою даного дослідження є визначення ефективних методів розвитку компетентної побудови корпоративних та інформаційних Інтернет-ресурсів. Під методами створення веб-сайтів розуміють сукупність прийомів та засобів розробки.

Інтернет як засіб комунікації, що не має територіальних кордонів, дозволяє обмінюватися різними видами інформації. В останні роки Інтернет мав величезний вплив на розвиток усіх світових компаній, змінивши спосіб представлення компанії потенційним клієнтам, а також обслуговуючи існуючих клієнтів. Кількість людей, які використовують Інтернет як життєво важливий засіб отримання необхідної інформації про надані послуги, значно зросла за останні роки. Інтернет допомагає як великим учасникам ринку, так і дрібним підприємцям у зростанні бізнесу.

Щоб представити свої послуги в Інтернеті, компанія повинна придбати веб-сайт. Він буде виконувати функції веб-сайту для візиток, щоб клієнти отримували контактну інформацію та уявлення про те, в якому секторі послуг працює кампанія, а також працювати безпосередньо з існуючими клієнтами. Наявність власного веб-сайту позитивно впливає на імідж кампанії; до клієнтів ставляться з великою впевненістю. Веб-сайт також допомагає значно збільшити продажі, будучи основним інструментом для вирішення різноманітних маркетингових завдань, і значно зменшує навантаження працівників в офісі, відповідаючи на звичайні запитання клієнтів, такі як місцезнаходження, перелік послуг тощо. Крім того, робочий

процес значно скорочується завдяки наявності веб-сайту особистого кабінету для існуючих клієнтів та форм зворотного зв'язку для онлайн-додатків.

З вищевикладеного випливає, що веб-сайт є необхідним атрибутом для сучасної компанії і визначає тему дипломної роботи - "Розробка веб-сайту інтернет-провайдера".

Об’єкт досліджень: Процес розробки сучасного сайту.

Предмет досліджень: Веб-сайт для інтернет-провайдера.

Мета дослідження: Створення повністю функціонуючого веб-сайту для компанії MegatronLink.

Завдання дослідження:

- Зробити дослідження діяльності компанії «MegatronLink».
- Провести огляд сучасних методів розробки веб-сайтів
- Обрати методи розробки веб-сайту.
- Створити дизайн веб-сайту.
- Здійснити верстку веб-сайту.

Методи дослідження: Технології які використовуються для створення веб-сайту.

1 АНАЛІТИЧНИЙ ОГЛЯД

1.1 Сучасні технології, які використовуються для створення веб-сайтів.

Веб-сайт - це сукупність документів та файлів, які розташовані на сервері (комп’ютері) та об’єднані єдиним доменним ім’ям.

Доменне ім’я - це назва веб-сайту, який використовується для його пошуку в Інтернеті. Наприклад, "gmail.com" - це доменне ім'я, яке поєднує сторінки на сервері.

Сервер може бути простим комп’ютером, на якому встановлені певні програми, що дозволяють веб-сайту працювати.

Браузер надсилає конкретний запит, який є доменним іменем веб-сайту, сервер приймає його і відправляє назад деякі документи. Це працює як копія мережевого файлу, лише за протоколом http.

Усі документи на веб-сайті, які ми бачимо в браузері, написані у форматі HTML. HTML (мова розмітки HyperText) - це мова, яка є стандартом для розробки документів, які відображатимуться в Інтернет-браузерах. Ключове значення мови HTML полягає у маркуванні тексту, для чого використовуються спеціальні команди, що називаються тегами, вони відображаються графічно в кутових дужках. Найкраще в документах HTML - це те, що їх може переглядати майже будь-який браузер. Документ, написаний у форматі HTML, буде правильно відображатися на різних пристроях з різною роздільною здатністю екрану та операційними системами. Майже всі документи містять стандартні елементи, такі як блок заголовків, заголовки або списки. Теги HTML можуть використовуватися для позначення цих елементів, забезпечуючи веб-браузерам мінімальну інформацію для відображення цих елементів, зберігаючи при цьому загальну структуру та повноту інформації в документах. Браузер - це вікно, через яке ми переглядаємо веб-сайти в Інтернеті. Тобто всі сторінки повинні бути написані в HTML[1].

Є кілька способів створення HTML-документів, які ляжуть в основу нашого веб-сайту.

Основний спосіб - це ввести HTML-код в редактор. Це досить трудомісткий процес, який вимагає глибокого знання принципів побудови документів у HTML, вміння працювати з каскадними таблицями стилів CSS та принаймні javaScript.

Другий спосіб - використання CMS. Це механізм веб-сайтів, який дозволяє користувачеві створювати веб-сайти майже автоматично, передбачає менше знань для побудови та займає набагато менше часу на розробку. Крім того, при використанні системи управління вмістом набагато простіше виконувати подальше обслуговування веб-сайту, додаючи або видаляючи нову інформацію[9].

Також можливо створювати веб-сайти за допомогою спеціальних редакторів, які згруповані під загальною назвою WYSIWYG - "What You See Is What You Get", що можна перекласти як "те, що ти бачиш, те і отримуєш". ці програми виглядають як робота в графічних та текстових редакторах, але на виході виходить не зображення, а документ у форматі HTML. Цей метод дозволяє швидко отримати документ із досить складною конструкцією.

Давайте розглянемо переваги та недоліки цих методів. Спочатку здається, що переваги редакторів CMS або WYSIWYG незаперечні, але це далеко не так. Створення рукописних веб-сайтів може зайняти більше часу, але вони виявляються якіснішими.

Відзначимо переваги рукописних веб-сайтів:

Веб-сайт, створений таким чином, набагато складніше зламати, оскільки немає бази даних MySql і файлу конфігурації, і саме за допомогою цих файлів хакери намагаються впершу чергу зламати веб-сайт.

Веб-сайти, написані в HTML, з’являються набагато швидше, ніж системи управління вмістом. Це пояснюється тим, що самі файли набагато компактніші, і немає потреби запитувати базу даних MYsql.

Швидкість веб-сайту позитивно впливає на позицію результатів

пошуку. За тих самих умов веб-сайт, написаний у форматі HTML, буде вищим.

Веб-сайт займає менше місця, що дозволяє економити на планах хостингу.

Веб-сайт HTML не буде містити дублікатів сторінок, що позитивно в результатах пошуку.

Каскадні таблиці стилів CSS використовуються для опису зовнішнього вигляду елементів.

CSS є досить базовою офіційною мовою, яка була винайдена для опису зовнішнього вигляду документів. Це говорить про те, що він досить простий і складається з оригінальних примітивних структур, вивчити які не так складно. Найскладніше - це не синтаксис, не правила написання конструкцій, а величезна кількість властивостей пам'яті CSS, що виконують різні завдання. На щастя, усі правила англійською мовою з відповідним смисловим навантаженням. Простий переклад на нашу мову дає уявлення про те, що робить це правило, і навпаки - коли ми перекладаємо те, чого хочемо досягти за допомогою певної властивості, англійською мовою, існує велика ймовірність того, що ми отримаємо правильну властивість. Це значно полегшує інтуїтивне запам'ятовування правил CSS. Наприклад, якщо вам потрібно встановити колір фону, досить перекласти на англійську мову, в результаті чого ми отримуємо колір фону (окремі слова в CSS пишуться тире)[4].

CSS досить простий у використанні в документах HTML. Його можна зв’язати як зовнішній файл CSS з документом. Для цього просто введіть тег <link rel = "stylesheet" href = "nyTbMo ^ afina.css"> у тег <head>. Це найпоширеніший спосіб прив’язки таблиць стилів до документа, коли зовнішній дизайн сторінок відображається в окремому зовнішньому файлі CSS.

CSS має досить простий синтаксис. Правила реклами настільки прості, що їх можна описати одним реченням. Спочатку пишеться селектор, який

виділяє певний елемент сторінки, після фігурних дужок властивості записуються зі значеннями після двокрапки, а самі властивості відокремлюються один від одного крапкою з комою. Найскладнішою частиною декларації CSS є селектор[14].

CSS селектор - (від слова select - вибирати) - це конструкція, з якої починається кожен блок оголошень, і яка служить для вибірки елемента або однотипних елементів на сторінці для подальшої стилізації. Найчастіше в якості селектора використовується певний клас тега, наприклад:

```
1.  //HTML:
2.  <div class="my-class"></div>
3.  //CSS:
4.  .my-class {
5.
5.      background-color: #999;
7.
6.  }
9.
```

Тут селектор - це клас my-class тегу div, який отримує необхідний стиль у файлі CSS. У цьому випадку кольором тла є сірий. Відповідно, якщо на сторінці є кілька маркерів (не тільки div) з класом my-class, всі ці елементи отримають однаковий дизайн - сірий колір тла # 999.

1.2 Етапи створення веб-сайту.

Створюється модель або прототип веб-сайту

На цьому етапі окресліть зовнішній вигляд веб-сайту та загальну структуру веб-сайту, визначивши, які елементи та в якому порядку він буде містити. Для цього можна використовувати різні програми та онлайнові сервіси, але їх також можна просто відобразити на аркуші паперу.

Створення дизайну веб-сайту

Простий макет, що відображає дизайн веб-сайту, намальований у графічному редакторі, такому як Adobe Photoshop або Gimp або Adobe Experience. Це один з основних кроків у створенні веб-сайту. На цьому етапі зовнішній вигляд веб-сайту остаточно формується, додається вся графіка,

вказуються та малюються всі деталі[15].

Верстка сторінки в HTML

Макет веб-сайту, намальований у графічному редакторі, розділений на окремі графічні елементи та описаний в HTML; такі технології, як CSS та javascript, також можуть бути використані в макеті, якщо потрібна якась динамічна завантаження чи анімація[8].

1. Розробляється програмна частина або використовується двіжок;
2. Веб-сайт розміщується на хостінгу.

Підготовлені HTML-сторінки копіюються на сервер, їм присвоюється певне доменне ім'я та надається загальнодоступний доступ.

1.3 Дизайн веб-сайту

Перше, на що слід звернути увагу під час відвідування веб-сайту, - це його дизайн. Чи бажає користувач переглянути сторінку далі чи закрити її, залежить від того, наскільки продумано веб-сайт. Тому при розробці веб-сайту дуже важливо добре розуміти, для якої аудиторії він призначений, які потреби виникають у користувачів. Всі ці проблеми вирішуються Ux / Ui. UX або користувацький досвід (дослівно: "користувальницький досвід") - дисципліна, яка вивчає досвід користувача з продуктом, сприйняття та реакцію в результаті використання[6].

Інтерфейс користувача – це інтерфейс користувача (буквально "інтерфейс користувача") - як виглядає інтерфейс і яких фізичних характеристик він набуває. Визначає, якого кольору буде ваш «товар», чи буде зручно натискати кнопки, чи буде текст читабельним тощо.

UX - це концепція, яка застосовується не тільки до веб-дизайну, а й до інших областей. Досвід взаємодії - це коли фабричний фахівець керує складним обладнанням, натискаючи кілька кнопок, а кнопка відключення набагато більша. UX - це коли ви їдете ярликом до свого дому, незважаючи на те, що у нього гарний і чистий тротуар; це коли ви кладете чашку кави в спеціальні поглиблення на панелі всередині вашого автомобіля; це коли

педаль гальма ширша за педаль газу. Це все взаємодія з користувачем. Прикладів може бути багато, але суть однакова: UX - це спосіб, яким користувач може досягти мети найбільш зручним способом. І коли ви створюєте хороший дизайн, ви повинні над цим подумати. UX - це область, яка включає як дослідження, так і дизайн, а також візуалізацію та макет. В результаті роботи над UX нам потрібно отримати робочий прототип. Оскільки ми говоримо про веб-дизайн, таким прототипом є HTML-макет.

UX у веб-дизайні - це основа, на якій будується дизайн веб-сайту або програми. Це дизайн інтерфейсу з урахуванням потреб людей, для яких ви розробляєте веб-сайт. Хто такий кінцевий користувач, які цілі переслідує людина за допомогою вашого інтерфейсу, як ви можете допомогти йому досягти результату якомога швидше та зручніше? Ось основні завдання, які вирішує UX.

Багато веб-дизайнерів плутають веб-дизайн та візуальний дизайн. Тим не менш, хтось вважає, що процес створення дизайну веб-сайту полягає впершу чергу в створенні картинок, кнопок, піктограм (UI), повністю забуваючи про базу (UX), для вивчення людей, для яких це все намальовано.

Які завдання вирішує UX?

Основним завданням дизайнера є максимізація рівня задоволеності кінцевого споживача від взаємодії з товаром. Під товаром ми маємо на увазі будь-який об'єкт взаємодії з користувачем, будь то реальний продукт, послуга, веб-програма чи веб-сайт.

Робота веб-дизайнера полягає у подоланні розриву між власником бізнесу та потенційним клієнтом. Ви повинні розуміти, що основною дослідницькою роботою в UX-дизайні є вивчення потенційної аудиторії товару. Найважче тут те, що неможливо відчути, не можна виміряти - це співпереживання веб-дизайнера або спеціаліста з UX, здатність зрозуміти чужі бажання та почуття.

Правильний шлях - дослідження. Дослідження - це перший крок дизайнера в роботі над будь-яким проектом. Вам потрібно чітко показати

групу користувачів, якомога більше дізнатись аудиторію, для якої ви розробляєте інтерфейс, зрозуміти потреби цієї аудиторії. Звичайно, лише виходячи з вашого особистого досвіду та ваших почуттів, досить складно дійти до об'єктивно правильного рішення в інтерфейсі, оскільки ви не безсторонні щодо результатів своєї творчості, і ваш особистий досвід щодо конкретного товару може бути дуже обмежена. Однак, якщо ви ретельно дослідили продукт, для якого ви розробляєте веб-сайт, ви можете зробити деякі висновки, прийняти рішення. Найменше, що вам просто потрібно зробити, це вивчити поточний діловий досвід, накинути портрет пересічного споживача товару. Цей момент говорить про те, що дизайн UX дуже тісно пов'язаний з бізнесом.

Напочатку опитування вам слід поговорити з власником бізнесу, вашим клієнтом. Він спеціаліст у своїй продукції, має досвід роботи з клієнтами, знає свою аудиторію. Усі ключові моменти слід записати для подальшого аналізу. Також слід вивчити досвід роботи таких продуктів.

Wireframing, каркасне моделювання

Після того, як ви визначили аудиторію, визначили цілі та цінності людей, ви можете почати моделювати структуру інтерфейсу, так звану телеграму. Каркасна мережа використовується для розповсюдження інформації на майбутніх сторінках у порядку важливості - зверху вниз. У процесі моделювання необхідно враховувати, яка інформація буде на сторінці, визначати основну форму відображення інформації, але не вдаватися в деталі, не займатися візуалізацією.

Для створення рамки можна скористатися спеціальною програмою або намалювати її від руки на папері у коробці чи крапці, а потім відсканувати та помістити в папку матеріалів проекту. Онлайн-додаток wireframe.cc ідеальний, оскільки ви можете вносити зміни в режимі реального часу та домовлятися з клієнтом.

Необхідно опрацювати всі сторінки веб-сайту або програми, і лише після того, як фреймворк всього веб-сайту буде готовий, перейдіть до

наступного кроку.

Інтерфейс користувача, візуалізація після повної розробки UX, створюється кілька дротових фреймів і вибираються найбільш привабливі варіанти, можна починати візуалізацію. Візуалізація - це малювання дротяних рамок, створення єдиного стилю та дизайну вмісту. Іншими словами, ми починаємо працювати над інтерфейсом користувача. Найчастіше для візуалізації використовують Adobe Photoshop, Sketch.app, Inkscape + Gimp або інші інструменти. Для роботи з векторною графікою ми будемо використовувати Inkscape - для створення піктограм та іншої необхідної графіки, для роботи з растровою графікою - найкраще рішення - Adobe Photoshop[17].

У веб-дизайні існують правила гарної форми. При розробці веб-сайту веб-дизайнер дотримується деяких негласних правил дизайну. Міжрядковий інтервал, інтервали, шрифти та розміри елементів можна виміряти, і є прямо неякісні перетворення. Тому ви можете скласти набір правил, дотримуючись яких допоможе не зробити погану візуалізацію вашого дизайну. Перелічимо кілька правил:

Типографіка, текст, посилання:

- Не використовувати занадто великі заголовки;
- Не використувати шрифт меньше 12px;
- Не робіть дуже маленький або дуже великий міжрядковий інтервал;
- Не розтягуйте літери за допомогою інструмента «Перетворення», шрифт повинен бути природним пропорційним;
- Не використовуйте більше 3 шрифтів на сторінці;
- Не використовуйте занадто малий контраст, не друкуйте світло-сірим на білому або темно-сірим на чорному;
- Використовуйте інтервал між символами з обережністю якщо знаєте, що робите и обраний шрифт дозволяє зробити текст "повітряним" найбільш елегантно;
- Не робіть занадто малих відступів між абзацами, заголовками та

елементами, що називається «надати дизайну повітря»;

- Не використовуйте верхній регістр букв без необхідності;
- Не використовуйте надто складний декоративний шрифт для основних текстових полів, це повинен бути простий і зручний для читання шрифт деяких сімейств Sans або Serif;
- Усі посилання, крім елементів навігації, повинні бути виділені. Спробуйте також відформатувати посилання, які вже були відвідані, з більш темним кольором на відміну від стандартного;
- Якщо ваш веб-сайт має більше 3 рівнів ієрархії, не забувайте про так звані сухарі. Це пункти меню, які повідомлять користувачеві, на якому рівні він знаходиться.

Графіка, іконки, фотографії:

- Не використовуйте шаблони фотографій у своєму дизайні. Краще зробіть це самостійно, порекомендуйте клієнту зв’язатися з фотографом або знайти найбільш «яскраві» фотографії;
- Не використовувати іконки, зроблені з фотографій;
- Усі іконки мають бути в одному стилі;
- Не збільшуйте фотографію понад вихідний розмір;
- Не масштабуйте графіку непропорційно;
- Не застосовуйте інші режими змішування шарів, крім звичайного (Normal);
- Намагайтеся не застосовувати фільтри до зображень, які повинні мати кілька станів (наприклад, коли ви наводите курсор миші, наприклад). Всі накладання, зміни - лише шляхом накладання нового шару. Все повинно бути легко відтворено в HTML-макеті. Це не стосується ретуші та підготовки фото[13];
- Не масштабуйте фотографію, перш ніж перетворити її на смарт предмет;
- Обрізати фотографію лише за допомогою маски для обрізання фігури, оригінальну фотографію, як і в попередньому абзаці, потрібно зберегти в

розумному об’єкті;

- Якщо піктограми намальовані у Photoshop, не растеризуйте їх із зменшеним розміром. Пам'ятайте, що макет все ще має етап макета, і всі піктограми потрібно буде векторизувати. Оригінальні розумні об’єкти з піктограмами повинні бути достатньо великими для якісного відстеження;
- В ідеалі, ви повинні мати усі плоскі піктограми та графіку у форматі SVG в окремій папці;
- Колір;
- Не використовуйте чисті кольори, не забувайте намагатися досягти найприємнішого відтінку;
- Не використовуйте більше 2 кольорів з акцентом на сторінці та не більше двох темно-сірих (або чорних) кольорів для тексту. В ідеалі, лише 1 акцентний колір і 1 темно-сірий / чорний для тексту;
- Намагайтеся малювати акцентним кольором лише ті елементи, які найважливіші на сторінці, зосередьтеся на них. Це кнопки, стрілки, закладки важливих елементів, текстові посилання, піктограми інформації (маленькі піктограми, такі як «логін», «пароль», «пошта» та піктограми у формах не слід підкреслювати);

Правила роботи в графічному редакторі, організації роботи тощо:

- Назвіть шари зі значенням;
- Спробуйте розкласти сенсові блоки та компоненти інтерфейсу за групами;
- Завжди створюйте окрему папку з усіма шрифтами, що використовуються у форматі TTF або OTF. У цій папці також слід розміщувати шрифти значків;
- Створюйте макети 1 в 1 (72 ppi). При 100% візуалізації макети повинні мати той самий масштаб, що й очікуваний макет HTML;
- Не забувайте використовувати якусь мережу у своїй роботі. Ви можете розробити його самостійно, але рекомендується використовувати мережу Bootstrap. Ширину вмісту можна регулювати вручну з вихідної ширини сітки Bootstrap, тому ви можете використовувати мережеві плагіни на

випадок використання Photoshop;

- Не створюйте декоративних елементів, якщо це не має практичного сенсу. Якщо це веб-сайт із зображеннями, і вам потрібна красива фотографія, це правило можна пропустити;
- Дотримуйтесь сенсової візуальної ієрархії.
- Використовуйте правило "внутрішнього та зовнішнього", яке передбачає, що відстань між внутрішніми елементами блоків має бути менше зовнішньої відстані між блоками;
- Пам’ятайте, що люди найчастіше заходять не на головну сторінку веб-сайту, а на внутрішні, тому слід подумати про інформативні універсальні блоки - верхній, нижній, нижній боки (якщо такі є). Обмеження має бути якомога інформативнішим, але не перенасиченим. Необхідні елементи: логотип, назва проекту, навігація. Пошук на веб-сайті та інші елементи розміщуються залежно від проекту;
- Хорошим рішенням є розміщення розширеної навігаційної карти чи веб-сайту з усіма прихованими місцями в нижньому колонтитулі. Спробуйте спробувати розглянути підвал веб-сайту до найдрібніших деталей, а не робити дефіцитні вузькі підвали з логотипом та номером телефону. Краще зробити його високим і розгорнутим. Дуже часто користувачі знаходять необхідну приховану інформацію в підвалі веб-сайту.

1.4 Програмне забезпечення, необхідне для створення веб-сайту

Розглянемо програмне забезпечення, необхідне при розробці веб-сайту.

1.4.1 Утиліти

Щоб автоматизувати процес веб-розробки, вам потрібна утиліта, така як глотка.

Gulp - це інструмент, який допомагає автоматизувати буденні завдання веб-розробки. Gulp призначений для вирішення таких завдань, як:

Створення веб-сервера та автоматичне перезавантаження сторінки в браузері при збереженні коду, відстеження змін у файлах проекту;

Використання різних препроцесорів JavaScript, CSS та HTML (CoffeeScript, Less, Sass, Stylus, Jade тощо)[12];

Мінімізуйте CSS та JS-код, а також оптимізуйте окремі файли проекту в одному;

Автоматично генерувати префікси постачальника (префікси до імені властивості CSS, яке постачальники браузера додають для власних властивостей) для CSS.

Керувати файлами та папками в рамках проекту - створювати, видаляти, перейменовувати;

Запуск і моніторинг виконання зовнішніх команд операційної системи;

Робота із зображеннями - стиснення, створення спрайтів, зміна розміру (png, jpg, svg тощо);

Реалізація (надсилання на зовнішній сервер) проекту через FTP, SFTP, Git тощо. Підключення та використання в проекті нескінченної кількості утиліт, програм та плагінів Node.js та Gulp.

Створення різноманітних карт проектів та автоматизація інших посібників праці.

Можна з упевненістю сказати, що Gulp і безліч написаних для нього службових програм підходять практично для будь-яких завдань з розробки проектів будь-якої складності - від невеликого веб-сайту до великого проекту.

Кожен проект, який базується на файлі Gulp, базується на файлі gulpfile.js, який розкриває набір інструкцій з управління проектами. Я хочу відразу сказати вам, що інструкції щодо Gulp не є програмуванням, хоча, вони написані на JavaScript.

Щоб встановити gulp, потрібно встановити node.js

Node.js - це програмна платформа, заснована на движку V8 (що перекладає JavaScript у машинний код), що перетворює JavaScript із вузькоспеціалізованої мови в мову загального призначення. Node.js додає можливість JavaScript взаємодіяти з пристроями вводу-виводу через свій API

(написаний на C ++), підключати інші зовнішні бібліотеки, написані різними мовами, надаючи їм дзвінки з коду JavaScript. Node.js в основному використовується на сервері, слугуючи веб-сервером[16].

Створюючи дизайн веб-сайтів в Adobe Photoshop, вам не обійтися без використання мережевих мереж. Ви можете створити сітку самостійно за допомогою напрямних або скористатися готовим, перевіреним рішенням - готовою сіткою Bootstrap у форматі PSD.

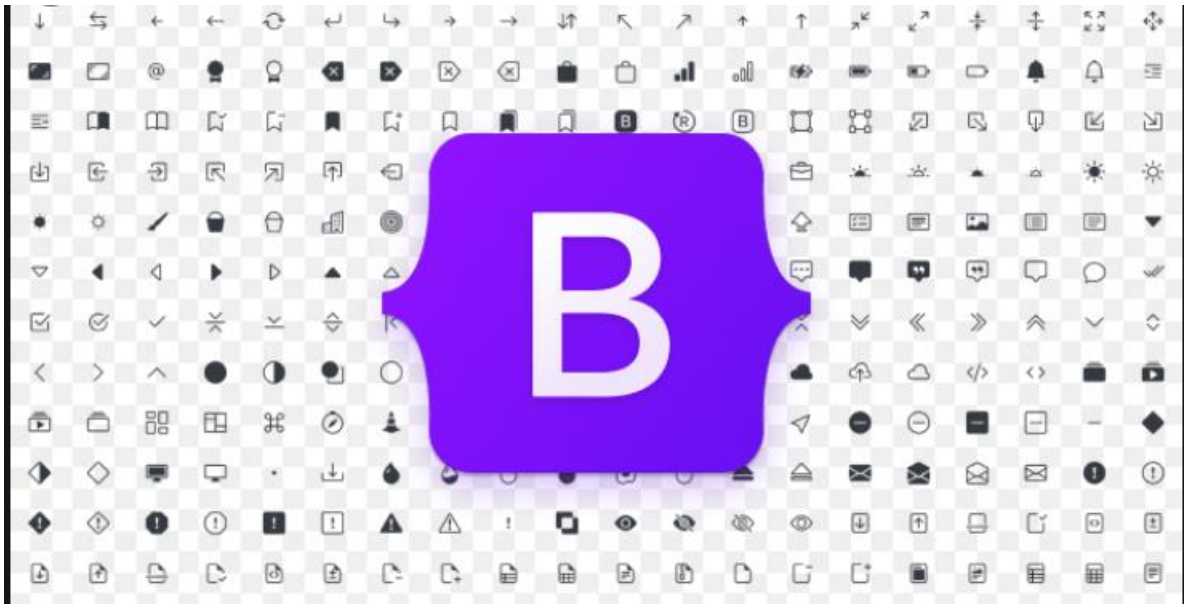

Рисунок 1.1 – Bootstrap

Незважаючи на те, що розміри сіток Bootstrap дещо стандартизують дизайн, цей варіант успішно використовується у своїй роботі величезною кількістю веб-дизайнерів по всьому світу. Крім того, така стандартизація дозволяє швидко виконати HTML-макет створеного макета, що неминуче пришвидшує роботу над проектами в контексті розробки вбудованих веб-сайтів або створення шаблонів.

Такі браузери, як Opera, Google chrome, Mozilla Firefox для перевірки правильного відображення веб-сайту в різних середовищах.

1.4.2 Текстові редактори

Ці програми будуть потрібні для написання та редагування текстових

документів, які є документами HTML. Тут підходить навіть найпростіший редактор Блокнота, але для зручності краще використовувати спеціальні програми, які призначені для написання HTML-коду. Ці програми виділяють теги, допомагають знаходити помилки у коді та покращують структуру коду. Вони також мають багато корисних функцій для прискорення процесу кодування. Прикладами таких програм є Notepad ++, Aptana Studio.

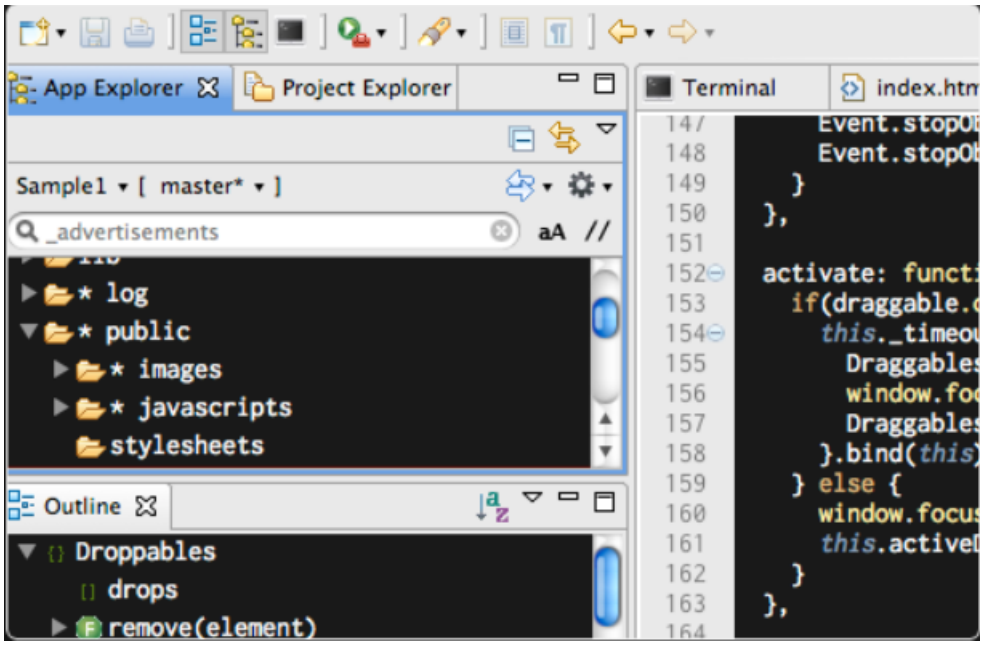

Рисунок 1.2 – Aptana Studio

Для наших цілей Sublime Text ідеально підходить, оскільки він має приємний інтерфейс, дуже гнучку настройку та функцію мультивибору, що значно пришвидшує процес розмітки.

1.4.3 Графічні редактори

Створюючи прототип і макет веб-сайту, ви можете використовувати графічні редактори, це програми, що дозволяють працювати з графікою (створювати та модифікувати різні зображення). Як і текстові редактори, вони можуть бути простими та складними. Windows Paint в основному вважається простим. До складних графічних редакторів, що працюють із растровою графікою, належать Adobe Photoshop та Gimp.

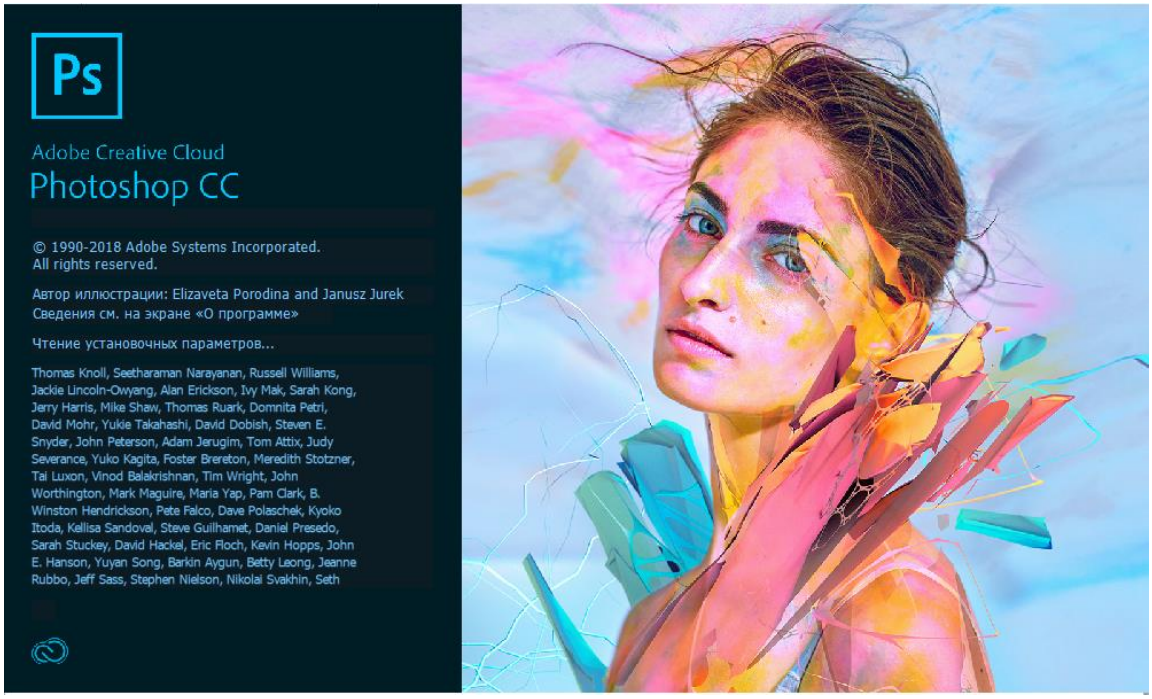

Рисунок 1.3 – Adobe Photoshop

Використовуйте для векторної графіки Inscape, Adobe Illustrator або CorelDraw.

Рисунок 1.4 – Adobe Illustrator

Ви також можете скористатися онлайн-сервісом wireframe.cc для створення прототипу веб-сайту. У ньому ви можете намалювати структуру веб-сайту, вказати розташування ключових елементів, їх розміри, приблизні кольори.

2 МОДЕЛЬ ПРОЕКТУ ТА СТРУКТУРА

2.1 Методика розробки проекту

Збір інформації означає отримання необхідної інформації від клієнта, а також з додаткових джерел.

Наступним етапом розвитку став відбір та упорядкування всієї зібраної інформації. Всі отримані дані були перетворені шляхом логічного відбору в набір інформації, відсортований за категоріями. Вся інформація чітко сортується за категоріями, потім трансформується в чітку ієрархічну структуру за категоріями. Крім того, були створені розробки готових програмних шаблонів, що містять інтерактивні компоненти.

Одним з головних і найважливіших завдань при розробці веб-сайтів є створення оптимального дизайну інтерфейсу веб-сайту. Для досягнення оптимального ефекту було розроблено кілька шаблонів графічного дизайну для передньої частини. Крім того, після узгодження з клієнтом був обраний найбільш оптимальний варіант.

Інтерфейс сайту повинен бути не тільки привабливим, але й рекламним за своїм характером і не містити зайвих елементів. Не повинно бути зайвої інформації, щоб інтерфейс не виглядав перевантаженим і сайт якнайшвидше завантажувався. У зв'язку з цим була обрана найпростіша, але найбільш функціональна версія зовнішнього інтерфейсу. Веб-сайт розроблений для кожного користувача Інтернету. Навіть недосвідчені користувачі можуть успішно переглядати його вміст. Наступним етапом розробки стало створення експериментальних програмних модулів на базі програмного забезпечення (програмний движок)[11].

Характеристикою цього етапу розвитку є активне використання досвіду інших розробників. Для цього була використана інформація, а також готові демонстраційні програмні компоненти, отримані з Інтернету шляхом пошуку необхідної інформації на веб-сайтах розробників-аматорів та

професійних розробників. Крім того, були відібрані отримані дані та підібрані найбільш відповідні, після чого були ретельно вивчені методи та принципи, що лежать в основі програмних компонентів, розроблених незалежними розробниками. Після детального вивчення принципів роботи та прикладних прийомів програмування були створені програмні шаблони та процедури збірки робочої частини програмного «двигуна» Веб-сайту. Далі йде фаза розробки, під час якої проводиться тестування та налагодження окремих програмних компонентів та блоків, призначених для побудови завершеного проекту веб-сайту. Останній етап розробки - це складання та тестування готового проекту для веб-сайту. Методом роботи на цьому етапі є багатоплатформене тестування готового програмного продукту. Готовий проект перевірявся на працездатність на локальному комп'ютері розробника, на мережевому сервері, а також на різних операційних системах із встановленим набором програмних компонентів.

2.2 Алгоритмізація задачі

Основним завданням у створенні та розробці робочої частини проекту є створення повністю функціонуючого програмного коду із зовнішнім графічним надбудовою (користувальницький інтерфейс). Це завдання ділиться на кілька етапів:

1. Створіть функціональний HTML-код для відображення зовнішніх форм та елементів керування.

2. Створення активних елементів дизайну на основі об’єктно-орієнтованих мов HTML.

Простіше кажучи, алгоритмізація задачі зводиться до вирішення двох основних проблем:

- Розробка функціонального інтерфейсу користувача спереду, в HTML

- Розробка внутрішнього серверного компонента на основі технології ASP для роботи інтерактивних елементів програмного продукту, що спеціалізується на роботі з базами даних.

Опис алгоритму

У розширеній, детальній формі алгоритм виконання завдань такий:

1. Створення шаблону зовнішнього інтерфейсу за допомогою технології HTML, розробка дизайну візуальних компонентів: зовнішній вигляд Інтернету - сторінок, активна анімація, елементи керування.

2. Встановлення процедур захисту від несанкціонованого доступу та несанкціонованої реєстрації.

3. Створення модуля ASP на форумі в Інтернеті.

4. Створення маніпулятора для всіх можливих помилок.

5. Тестування та відладка.

2.3 Структурна схема

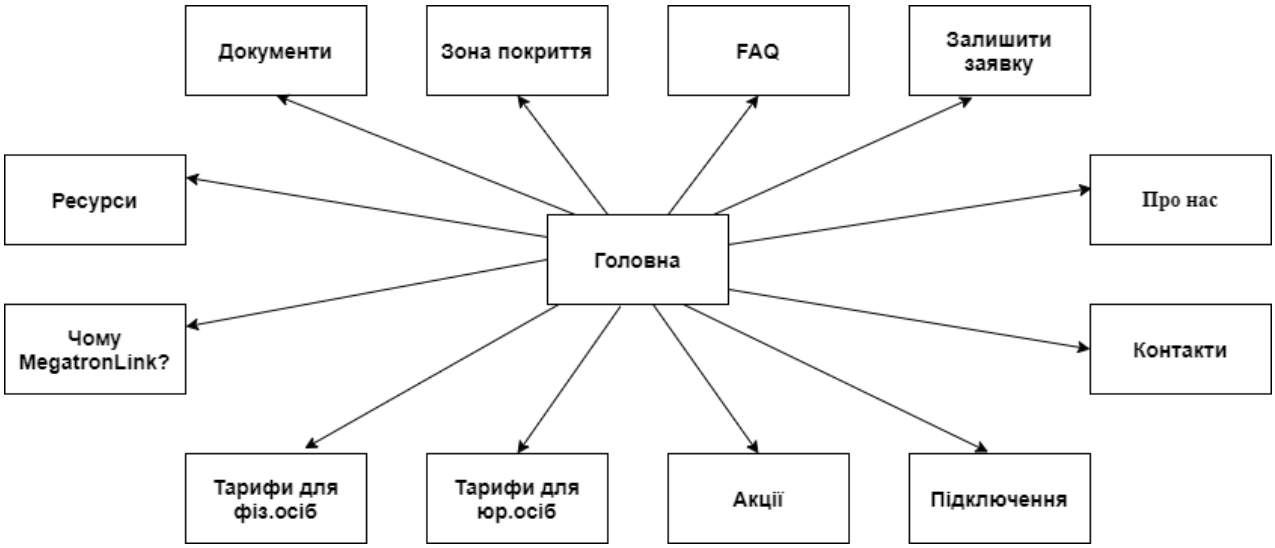

Рисунок 2.1 – Приклад структурної схеми

2.4 Дизайн

Створення дизайну - це креативна ідея, розробка базової графічної концепції дизайну сайту на прикладі головної сторінки. Адаптація елементів фірмового стилю клієнта для сайту.

Програмування означає створення базової схеми взаємодії та функціонування веб-сайту, а також розробку інтерфейсів для взаємодії з користувачем.

Тестування сайту на наявність помилок, тестування сторінок на належну роботу в різних браузерах (Internet Explorer, Opera,);

Організація роботи з розміщення проекту в Інтернеті у домені клієнта. Підсумкове тестування проекту.

2.5 Програмування

Для виконання всіх перерахованих вище функцій і методів розроблений спеціальний програмний механізм, що дозволяє реалізувати всі перераховані вище переваги.

Метод побудови веб-сайту такий: усі сторінки інформаційної частини сайту містять абсолютно однакову структуру програми та програмний код. Різниця полягає лише в текстовій інформації, відображеній на сторінках. Тому необов'язково описувати кожну інформаційну сторінку окремо. Тому буде детально описана лише одна модель програмування[5].

Усі програмні блоки описані з посиланням на програму, яка містить повний перелік усіх окремих веб-сайтів. Примітка: сторінки мають повністю однаковий програмний код, але відрізняються лише текстовим змістом теми сторінки, тому при описі програмного коду ви можете обмежитися описом лише одного файлу index.html.

2.6 Тестування та відладка

Тестування програмного забезпечення - це процес виявлення помилок у програмному забезпеченні (програмному забезпеченні). На жаль, існуючі в

даний час методи тестування програмного забезпечення не дозволяють однозначно і повністю налагодити належне функціонування аналізованої програми. Тому всі існуючі методи тестування працюють у формальному процесі тестування програмного забезпечення, що досліджується.

З точки зору ISO 9126, якість (програмного забезпечення) можна визначити як сукупну характеристику тестованого програмного забезпечення, враховуючи наступні компоненти:

надійність;

супровід;

практичність;

ефективність;

мобільність;

функціональність.

Рівні тестування:

Одиночне тестування (модульне тестування) - тестується мінімально можливий компонент тесту, наприклад, окремий клас або функція;

Тестування інтеграції – перевірка на наявність проблем в інтерфейсах та взаємодії між інтегрованими компонентами - наприклад, інформація не передається, передається неправильна інформація;

Тестування системи – інтегрована система тестується на відповідність початковим вимогам;

Альфа-тестування – імітація реальної роботи з системою внутрішніми розробниками або реальної роботи з системою потенційними користувачами клієнтом розробником. Альфа-тестування часто використовується для кінцевого продукту як внутрішній приймальний тест. Іноді альфа-тестування проводиться під налагоджувачем або середовище використовується для швидкого виявлення помилок. Виявлені помилки можна передавати тестувальникам для подальших досліджень в середовищі, подібному до того, в якому буде використовуватися програмне забезпечення;

Бета-тестування - у деяких випадках версія з обмеженими

функціональними можливостями або часом роботи розповсюджується серед певної групи людей, щоб гарантувати, що продукт містить менше помилок. Іноді бета-тестування проводиться, щоб отримати відгук про продукт від майбутніх користувачів.

Тестування білого та чорного ящиків

У термінології тестування фахівців (програмного забезпечення та деяких апаратних засобів) фрази "тестування білого ящика" та "тестування чорного ящика" стосуються того, чи має розробник тесту доступ до вихідного коду тестованого програмного забезпечення чи тестування проводиться через користувача інтерфейс або API, що надаються тестованим пристроєм.

Відладка - це етап розвитку комп’ютерної програми, під час якого виявляються, виявляються та усуваються помилки. Існують різні підходи та інструменти для налагодження; Відладчики використовуються як основний інструмент, що включає користувальницький інтерфейс для поетапного виконання програми: оператор за оператором, функція за функцією, з зупинками в деяких рядках вихідного коду або коли виконується певна умова[15].

Програмний продукт тестували як на локальному комп’ютері, так і на сервері. В результаті тестування було виявлено ряд помилок, які були усунені в процесі розробки.

Особливістю методології розробки проекту було окреме налагодження та тестування окремих програмних модулів та блоків в індивідуальному порядку з подальшим складанням компонентів та блоків в цілому.

Усуваються лише невеликі косметичні помилки: орієнтація тексту на сторінці, вибір оптимального поєднання кольорів, орфографічні помилки. В іншому випадку під час розробки цього проекту не потрібно було відлагоджувати.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Розробка прототипа веб-сайту

Після переговорів з власником компанії були визначені ключові моменти, які повинні бути відображені на веб-сайті. Оскільки веб-сайт в основному використовується для залучення нових клієнтів, інформація, необхідна клієнту для прийняття рішення про надання послуг, повинна знаходитися у верхній частині сторінки. Ця інформація являє собою короткий огляд діяльності компанії, її логотип та тарифи на надані послуги. Крім того, на першому екрані сторінки повинна бути контактна інформація для кампанії та форма зворотного зв'язку, де користувач може залишити онлайн-заявку на підключення послуг.

Для існуючих клієнтів компанії повинна бути форма для входу в особистий кабінет користувача, яка відображає поточний баланс та перелік супутніх послуг. Також повинна бути надана інформація про способи оплати та посилання на платіжні послуги.

Також на першій сторінці має бути посилання на інформацію про саму компанію та її поточні новини.

Тож коли ми вирішимо, що має бути на сторінці, ми можемо розпочати прототипування веб-сайту. Спочатку визначимось із загальною структурою веб-сайту. Це необхідно для того, щоб спочатку зрозуміти обсяг майбутнього веб-сайту, які технології будуть використовуватися та які функції буде включати веб-сайт. Існує декілька головних структур веб-сайтів:

Лінійна структура - це коли сторінки мають строго визначений порядок відносно один одного, а перехід від однієї сторінки до іншої є строго визначеним. Сфера застосування цієї структури досить обмежена. Ця структура чудово підходить для різноманітних тестів та навчальних веб-сайтів або веб-книг. Коли сторінки розташовані одна за одною, ви можете бути впевнені, що користувач не пропустить необхідний матеріал. Така структура відображена на рисунку 3.1.

Рисунок 3.1 - Лінійна структура веб-сайту

Варіант розробленої вище структури веб-сайту - це лінійна структура галузевих ресурсів. Це як дорога з безліччю гілок від неї. Користувач переходить з однієї сторінки на іншу в суворо визначеному порядку. Однак при необхідності він завжди може переїхати до іншої гілки і без особливих зусиль повернутися.

Головною перевагою розгалуженої лінійної структури є відносно проста можливість веб-майстрів створювати її на основі звичайної лінійної структури. У міру просування веб-сайту це часто необхідне. Зміст значно зростає і постає питання про вдосконалення навігації. Тому важливим процесом є те, як створити структуру веб-сайту.

Лінійна структура з альтернативами є ще однією підмножиною лінійної структури веб-сайту. Він відрізняється від лінійного тим, що користувач має більше можливостей для пошуку інформації, точніше надає можливість вибору між двома гілками. Наприклад, поділ корпоративних та приватних клієнтів на веб-сайті[10].

Найбільш поширеним використанням цієї структури є реєстрація відвідувачів веб-сайту. У цьому випадку всі користувачі починають з домашньої сторінки. Але тут є розподіл - людям пропонують форму для заповнення інформації, а представники комерційних структур - зовсім іншу.

Деревоподібна структура веб-сайту

Головною перевагою такої конструкції є її гнучкість. Він ідеально підходить для всіх типів ресурсів: для домашньої сторінки та веб-сайту для візиток та корпоративного веб-сайту, каталогу або порталу, структури веб-сайту інтернет-магазину. Також можливий негайний доступ до будь-

якого розділу або сторінки ресурсу. Модель деревоподібної структури показана на рисунку 3.2.

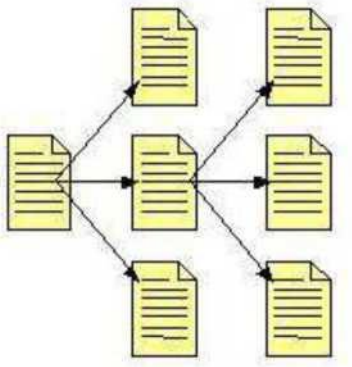

Рисунок 3.2 – Деревоподібна структура веб-сайту

Решітчаста структура (системи координат)

Це складна структура ресурсів, де всі компоненти розташовані в окремих гілках, по яких користувач може легко переміщатися (рис. 4). Цей тип структури найчастіше використовується для посилань або каталогів статей. На перший погляд може здатися, що решітчаста структура дуже зручна і прийнятна для користувачів, але все ж краще не використовувати її для звичайних веб-сайтів через те, що: створити решітчасту структуру дуже складно, оскільки доводиться копати в коді або налаштуйте для нього систему управління вмістом. Однак використання CMS допоможе вам зрозуміти структуру веб-сайту[9].

Використання структури сітки може легко заплутати не лише користувача під час пошуку інформації, а й самого веб-адміністратора при публікації вмісту.

Структуру решітки найкраще уникати для великих веб-сайтів, оскільки вони містять багато гіперпосилань.

• Буде легше зрозуміти такий ресурс, якщо представити приклад структури веб-сайту у вигляді схеми.

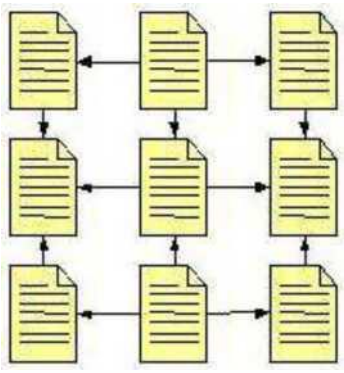

Рисунок 3.3 – Решітчаста структура веб-сайту

Змішана структура (павутиння)

Він складається з двох або більше компонентів вищезазначених структур в одній, але рідко використовується через складність реалізації. Ця структура вже набагато складніша за всі обговорені вище. Всі сторінки в ньому також розміщені на різних гілках[2].

Як правило, він використовується лише в каталогах. У той же час, ви можете переходити між гілками на глибокому рівні, використовуючи посилання на заголовки з інших розділів.

Також структуру веб-сайту можна розробити самостійно абокористувавши програму для встановлення структури веб-сайту (рис. 3.4).

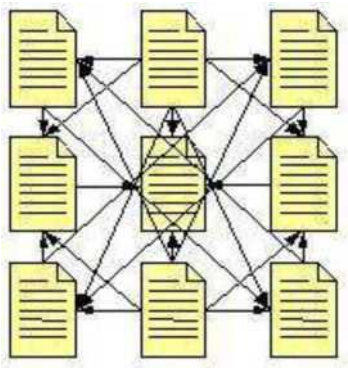

Рисунок 3.4 - Структура мережі

Для нашого веб-сайту буде використана деревоподібна структура, оскільки вона чудово підходить для веб-сайту малого бізнесу.

Структура навигації по веб-сайту

Види навігації можна розрізнити на основі двох критеріїв:

функціональний та візуальний.

За своїми функціями навігаційну систему поділяють на такі типи:

- Мова - навігація, відповідальна за мовний інтерфейс та відображення вмісту мовою, обраною користувачем.
- Основні - це найважливіші розділи веб-сайту, як правило, меню.
- Глобальний - це посилання, які повинні бути видимими з будь-якої сторінки веб-сайту, наприклад, посилання на домашню сторінку.
- Реклама - посилання для залучення відвідувачів на рекламні сторінки веб-сайту з розташуванням товарів та послуг.
- Тематична - навігація сторінками веб-сайту на певну тему (заголовок).
- Текст - гіперпосилання на текст на сторінці. Що стосується зручності використання, вони потрібні, щоб направити користувача до матеріалу, згаданого в тексті. З точки зору оптимізації, це компетентне посилання на веб-сайт.
- Орієнтовний - по-іншому, посилання. Гіперпосилання показує, в якій ділянці веб-сайту зараз знаходиться відвідувач.
- Географічний - використовується на веб-сайтах, які мають вкладки для певної країни.
- Візуальним дизайном виділяються наступні типи навігації:
- Текст - функціонально відповідає визначенню тексту. Це, мабуть, найдавніший тип навігації.
- Графічний - зараз найпопулярніший тип навігації з графічним дисплеєм, що використовується для всіх типів функціональної навігації.
- HTML-форми – допомагають в економії місця за допомогою випадаючих елементів або елементів, що відкриваються.
- Java і Flash технології – завдяки цим технологіям, ви можете організувати певну реакцію на дії, коли утримуєте курсор, натискаєте мишу або клавіатуру.

Для веб-сайту провайдера є гарною практикою використання сучасних

технологій у дизайні. Тому ми будемо використовувати плагін Mmenu для навігації. Це дозволяє вам створити гарне анімоване меню на веб-сайті та має гнучку систему налаштувань як функціональних можливостей, так і зовнішнього вигляду.

Домашня сторінка веб-сайту - це найважливіша сторінка веб-сайту. Від враження від домашньої сторінки користувач вирішить, подобається йому цей веб-сайт чи ні. Кожна домашня сторінка повинна відповідати декільком вимогам:

- Точно і стисло відображати тематику веб-сайту.
- Бути зрозумілою цільовій аудиторії, не містити нічого зайвого та не відволікати від головної інформації
- Повинна містити цінну пропозицію, яка негайно зацікавить потенційного клієнта
- Це повинно бути зручним і зрозумілим для користувача, щоб йому не довелося довго шукати елементи управління, також бажано зручне керування з мобільних пристроїв
- Мати спільний дизайн

Інші сторінки веб-сайту повинні бути об’єднані загальним дизайном, вони повинні мати пункти меню, посилання на головну сторінку та назву ресурсу. В інший час сторінки не повинні дублюватися, тобто, вони повинні мати унікальний вміст.

Структура веб-сайту интернет-провайдеру «MegatronLink» представлена на рисунку 3.5.

Рисунок 3.5 – Структура веб-сайту «MegatronLink»

Веб-сайт повинен підтримуватися усіма популярними сьогодні операційними системами - Windows, Linux, Android, IOS. Він також повинен відображатися належним чином у всіх популярних браузерах та у будь-якій роздільній здатності екрана. Тут використовується адаптивна верстка, яка дозволяє веб-сайту змінювати розмір та положення своїх елементів залежно від екрана, роздільна здатність якого відображається. Крім того, веб-сайти з адаптивним макетом мають рейтинг вище, у пошукових системах[3].

Як тільки будуть визначені основні моменти зовнішнього вигляду головної сторінки сайту, ви можете переходити до його прототипу. Інтернет-сервіс wireframe.cc. Перший екран прототипу веб-сайту показаний на рисунку 3.6.

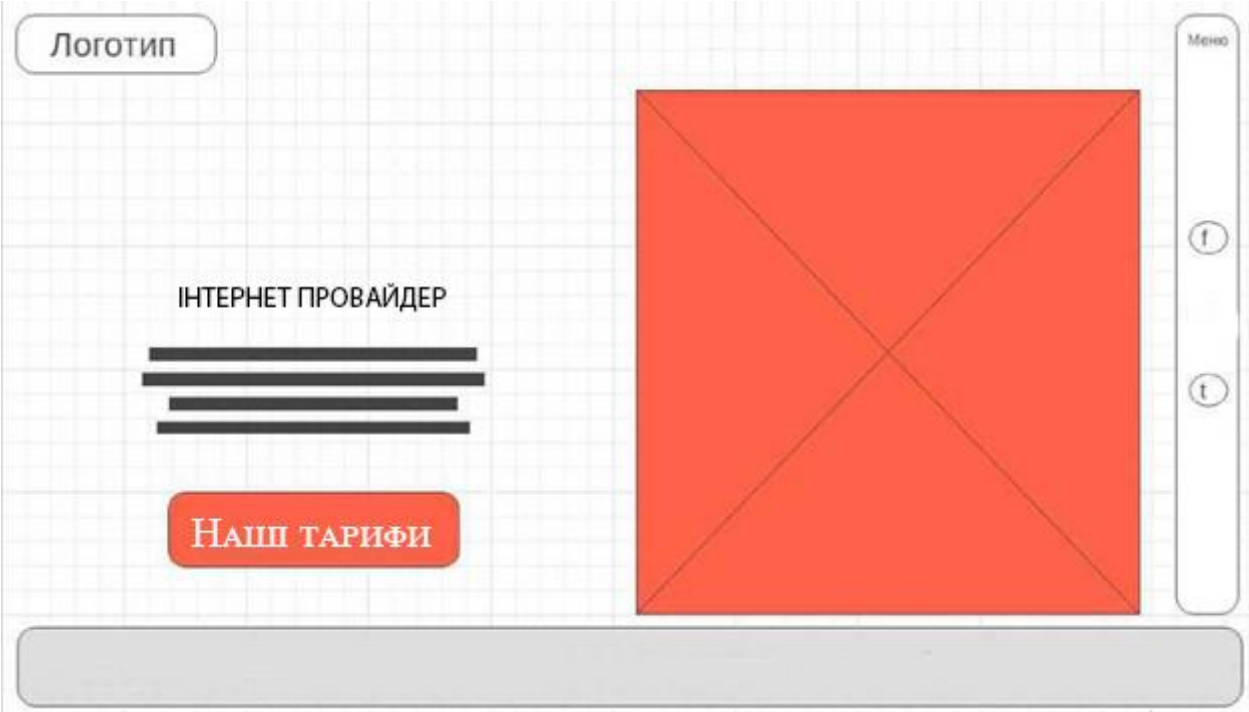

Рисунок 3.6 – Перший екран прототипа веб-сайту

Він містить логотип, контактну інформацію, напрямок діяльності компанії, має логотип та меню для навігації. Це те, що користувач побачить при відкритті веб-сайту, після чого він може перейти до детального вивчення інформації на веб-сайті, яке з’явиться на рисунку 3.7.

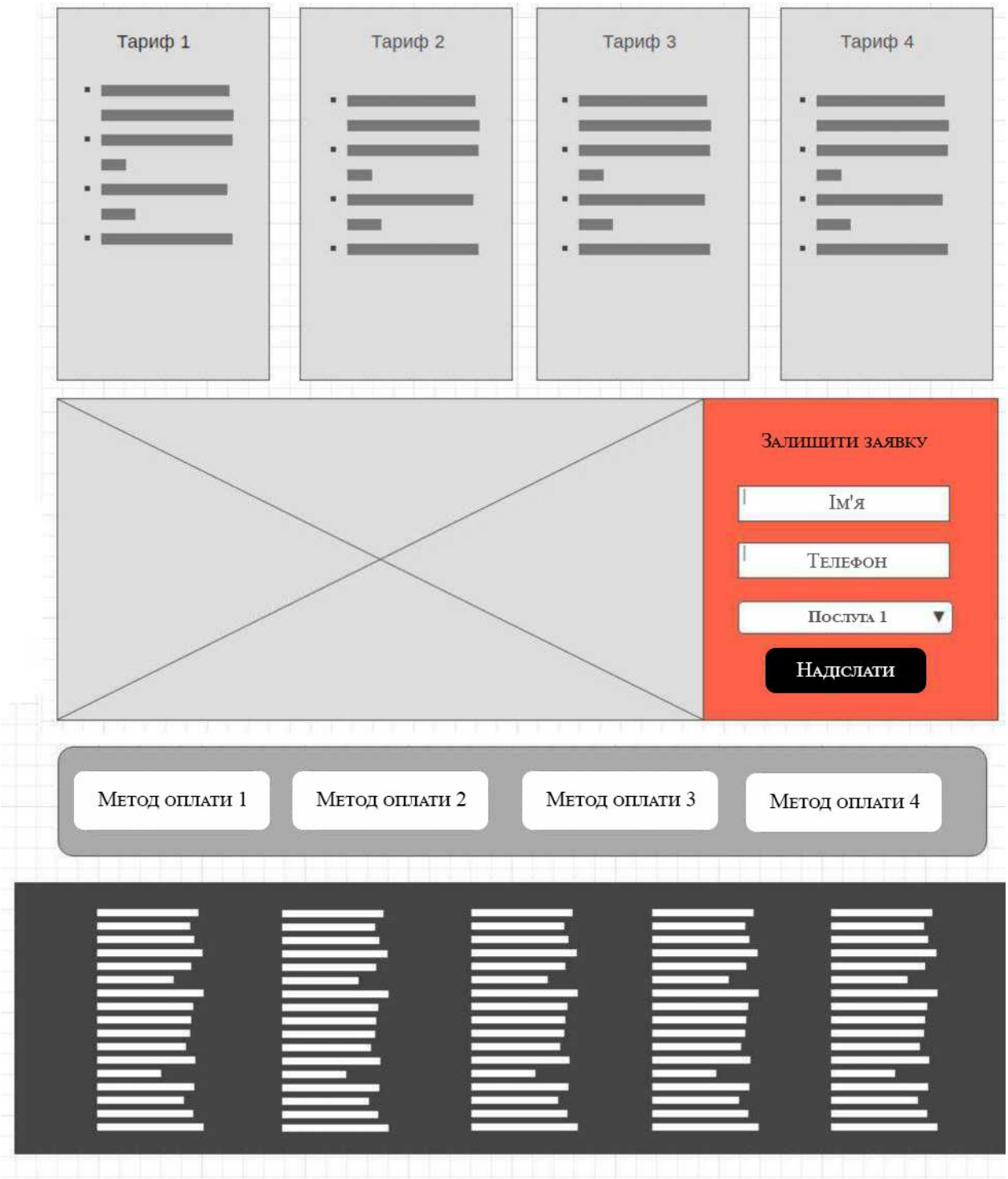

Рисунок 3.7 - Прототип низу домашньої сторінки веб-сайту.

Тут користувач може детальніше ознайомитись з діючими тарифами на послуги, що надаються компанією, поспілкуватися з кваліфікованою підтримкою, а також залишити запит через форму зворотного зв'язку та ознайомитись із способами оплати існуючих послуг. Ще користувач зможе подивитися зону покриття провайдера.

3.2 Створення дизайну веб-сайту «MegatronLink»

Як бачимо, дизайн веб-сайту виявився досить легким, на ньому мало тексту і представлена лише найнеобхідніша інформація. Після того, як прототип буде готовий, ви зможете починати робити ескіз більш детально та визначати остаточний дизайн майбутнього веб-сайту.

Візуалізація виконується в Adobe Photoshop, оскільки вона дозволяє запускати її швидко та ефективно. У дизайні ми використовуємо м'які кольори. В якості акценту будемо використовувати колір, взятий з логотипу компанії. Дизайн першого екрану показаний на рисунку 3.8.

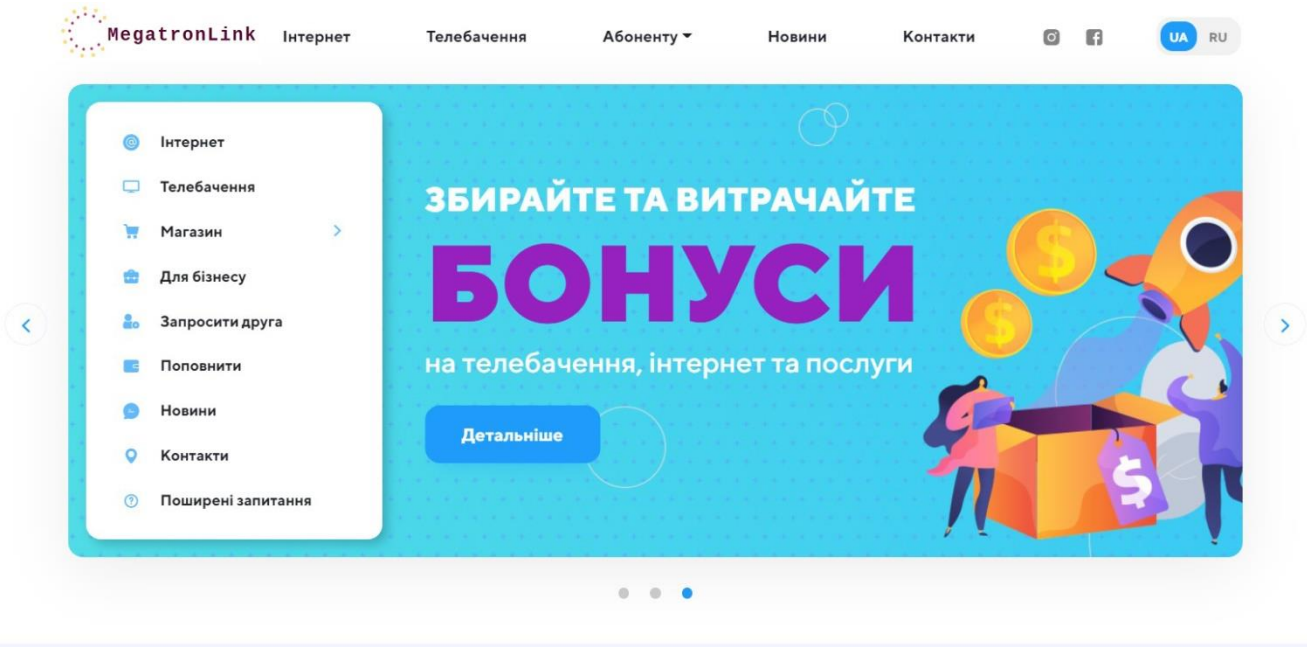

Рисунок 3.8 – Дизайн першого екрану веб-сайту «MegatronLink»

После создания макета веб-сайта в Photoshop можно приступать к вёрстке веб-сайта в HTML.

3.3 Верстка веб-сайту «MegatronLink»

Перш ніж створювати веб-сайт безпосередньо, вам слід підготувати папки, які міститимуть необхідні документи, такі як шрифти, зображення, таблиці стилів, бібліотеки та сценарії. Також повинна бути сторінка index.html, яка буде домашньою сторінкою веб-сайту. Структура папок

показана на рисунку 3.9.

Рисунок 3.9 - Структура папок

Перш ніж розпочати макет, вам потрібно запустити глоток, який створить веб-сервер на вашому комп'ютері і дозволить вам відстежувати стан макета в режимі реального часу.

Для початку ми встановлюємо всю необхідну службову інформацію у файл index.html, включаємо стилі, вказуємо заголовок веб-сторінки та вказуємо шлях до каскадної таблиці стилів. Все це робиться в тезі <head>.

На рисунку 3.10 наведено службову інформацію на веб-сторінці.

```
<meta charset="utf-8">

  <title>Exellent</title>
  <meta name="description" content="">

<meta    http-equiv="X-UA-Compatible" content="IE=edge">
<meta    name="viewport" content="width=device-width, initial-scale=I, maximum-scale=I">

<meta    property="og:image" content="path/to/image.jpg">
<link    rel="shortcut icon" href="img/favicon/favicon.ico" type="image/x-icon">
clink    rel="apple-touch-icon" href="img/favicon/apple-touch-icon.png">
clink    rel="apple-touch-icon" sizes="72x72" href="img/favicon/apple-toiich-icon-72x72.png">
clink    rel="apple-touch-icon" sizes="114xll4" href="img/favicon/apple-toiich-icon-114xll4.png">

        Chromej Firefox OS and Opera
cmeta name="theme-color" content="#lalala">
<!-- Windows Phone
; cmeta name="msapplication-navbutton-color" content="#lalala">
<!-- iOS Safari —>
cmeta name="apple-mobile-web-app-status-bar-style" content="#lalala">

cstyle>body { opacity: 0; overflow-x: hidden; } html { background-color: #fff; }</style>

c/head>
```

Рисунок 3.10 – Службова информація web-сторінки

Вам також потрібно підібрати шрифти, які будуть використовуватися в макеті, і вказати змінні, які будуть використовуватися за замовчуванням. Оскільки ми використовуємо препроцесор у стилі Sass, все це робиться у

відповідних файлах із *. sass.

На рисунку 3.11 показано підключення шрифтів. Рисунок 3.12 - це оголошення змінних.

```
^import " mixins/font-face"

+font-face("raleway", "../fonts/RalewayRegular/RalewayRegular")
+font-face("raleway", " . ./fonts/RalewayRegular/Ralewaybold",bold)
+font-face("raleway", "../fonts/RalewayRegular/RalewayExtraBold",800)
+font-face("raleway", "../fonts/RalewayRegular/RaiewayLight",300)
+font-face("raleway", "../fonts/RalewayRegular/RalewaySemiBold",600)
+font-face("raleway", "../fonts/RalewayRegular/RalewayThin",100)
+font-face("firasans", "../fonts/FiraSansRegular/FiraSansRegularr)
```

Рисунок 3.11 – Підключення шрифтів

```
$default-font: "raleway", sans-serif
^accent: #14&08b
$blue: #66c7cd
$red: #f86868
$dark: #e9e2d2
```

Рисунок 3.12 – Оголошення змінних

Крім того, зосередившись на макеті, ми починаємо створювати блоки з потрібним вмістом усередині тегу <body> та описувати властивості об’єктів у файлі main.sass. Повний HTML-код міститься в Додатку А.

Зручно закривати всі окремі елементи веб-сайту в тегах та .

Теги та не мають властивостей за замовчуванням для зовнішнього дисплея, це так звані контейнери, тому ви можете застосувати до них будь-які стилі CSS, щоб елементи всередині цих тегів мали бажаний вигляд.

Тег спочатку був призначений для розділення веб-сторінки на логічні фрагменти, такі як нижній колонтитул, нижній колонтитул тощо. Але з появою нової семантичної розмітки веб-сторінок, що використовують HTML5, необхідність у такому повсюдному використанні тегу відпала.

Наразі тег використовується для групування елементів блоків.

Тег також дозволяє поєднувати будь-який набір елементів, таких як заголовок, кількість абзаців, список в одному блоці, який потім можна розташувати на веб-сторінці, створюючи складну схему позначення.

Тег слід використовувати, лише якщо жоден інший елемент на сторінці не підходить для позначення регіону.

Тег використовується для виділення вбудованих елементів, таких як окремі слова та фрази в абзаці тексту чи заголовка.

Тег <area> Представляє гіперпосилання на текст, який відповідає певній області карти зображення або точки доступу на карті зображення. Завжди вкладено всередину тегу <map>.

Тег <address> Вказує контактну інформацію автора / власника документа або статті. Відображається курсивом у браузері.

Верстка робиться зверху вниз, тому, слідуючи макету, ми спочатку створюємо логотип, який також буде посиланням на головну сторінку веб-сайту.

Іншим важливим моментом буде оформлення меню, оскільки в цьому проекті воно анімоване і працює за допомогою спеціального плагіна.

Меню навігації має два положення: у типовому положенні меню закрите, і користувач не бачить елементів у ньому.

Для того, щоб його відкрити, слід зробити одиночний клік мишею по символу «меню», після чого має включитися анімація відкриття и після цього будуть відображені пункти меню. Також відбувається затемнення області сторінки, яка не належить до меню в положенні відкритого меню навігації і видно його пункти та користувач може обрати потрібний. Після одиночного клацання миші по потрібному полю відбувається перехід по наявні посилання і меню навігації зачиняється[7].

Цей тип меню дуже зручний, оскільки в закритому положенні він не відволікає увагу користувача і дозволяє зосередитись на важливішій інформації.

Далі слідує блок, який відображає перелік діючих тарифних планів та популярні тарифи (рис. 3.13).

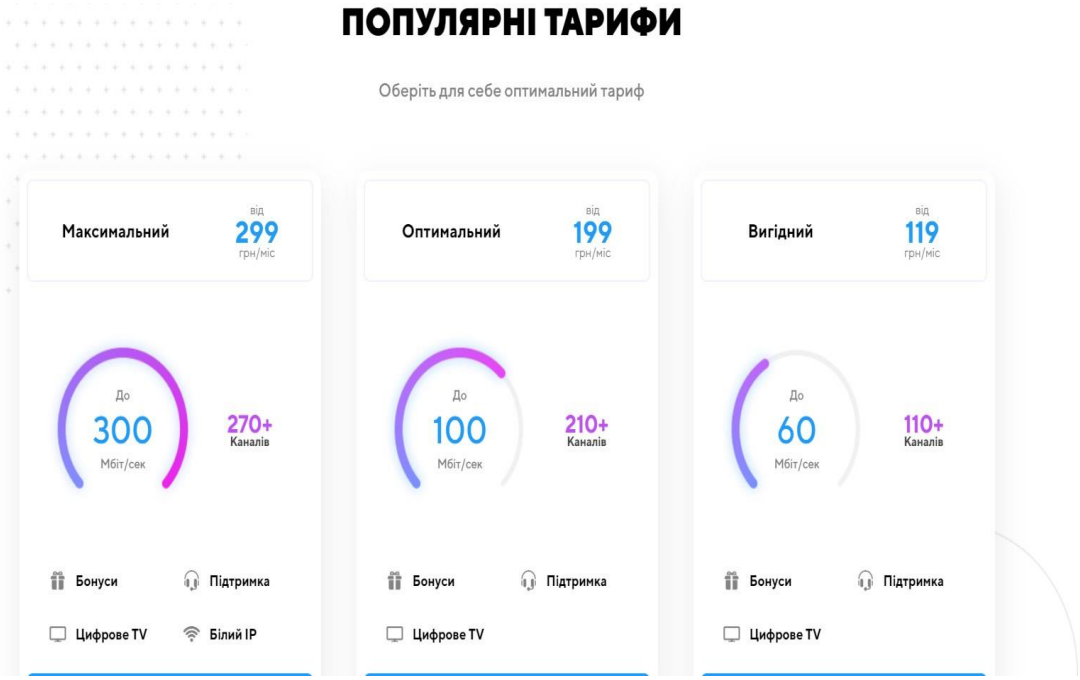

Рисунок 3.13 – Список тарифів

Ви також можете залишити запит на веб-сайті через форму зворотного зв'язку (рис. 3.14)[6].

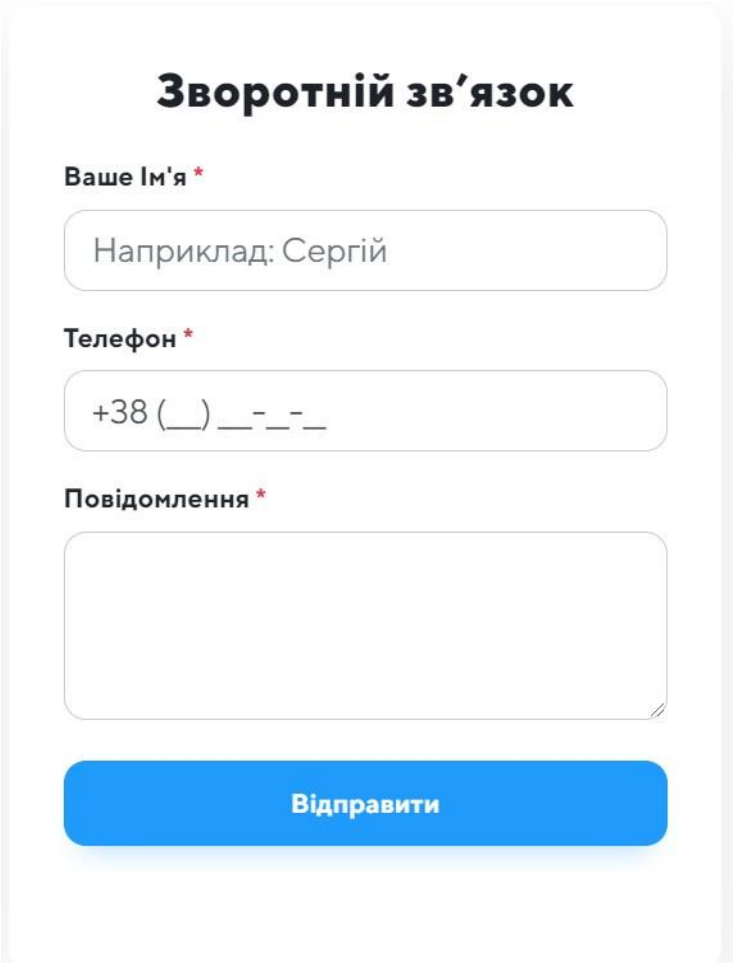

Рисунок 3.14-Форму зворотнього зв’язку

При заповненні форми буде надіслано електронне повідомлення з введеними даними, і персонал офісу зможе зв’язатися з клієнтом.

Далі подано блок-список платіжних систем, за допомогою яких можлива онлайн-оплата (рис. 3.15).

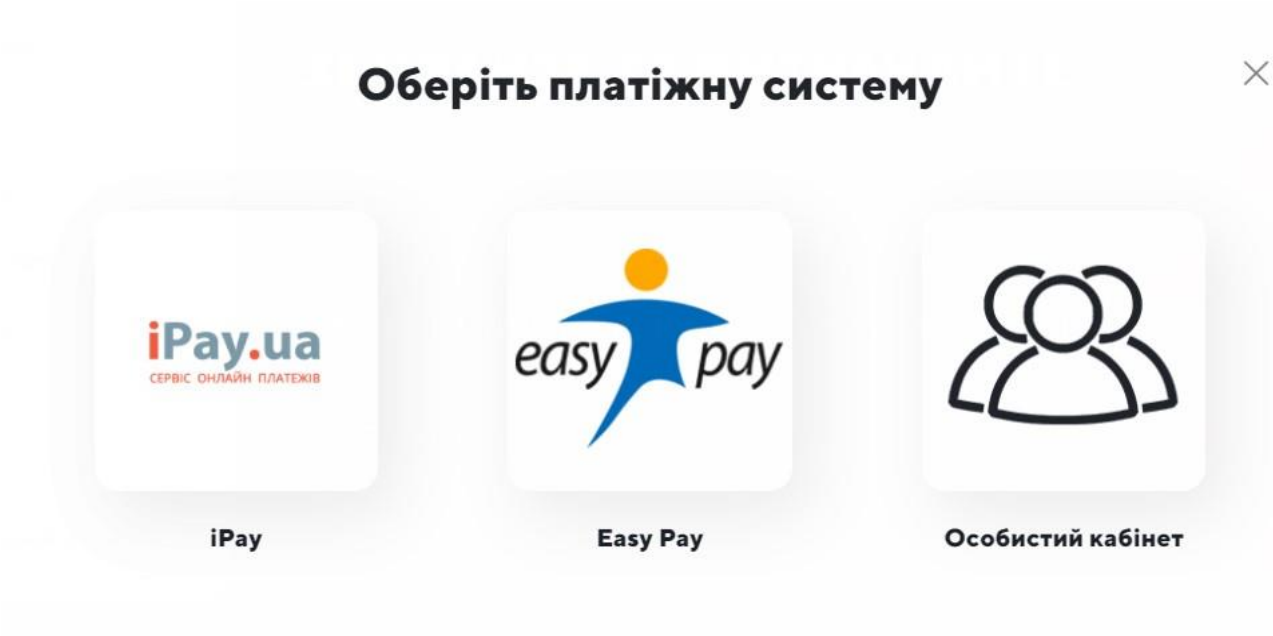

Рисунок 3.15 – Платіжні системи

За допомогою цієї сторінки користувач зможе отримати відповіді на найпоширені запитання.

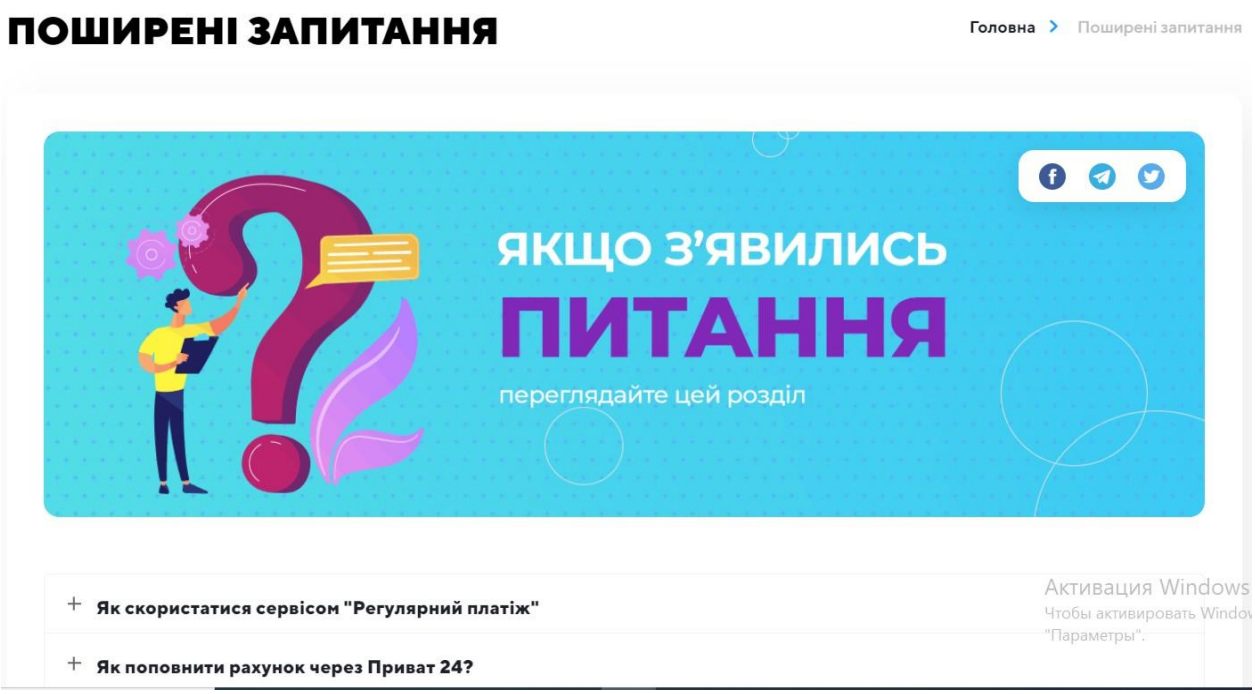

Рисунок 3.16 – FAQ

Особистий кабінет «MegatronLink»

Це затишний куточок користувача, де можна дізнатися про актуальні

акції та пропозиції, поповнити баланс або звірити історію платежів.

Доступ в особистий кабінет мають лише ті користувачі, що вже забронювали в ньому місце і є повноправними клієнтами інтернет-провайдера «MegatronLink». Якщо ж ви ще не з нами, ознайомитися з тарифами можна на сторінці «Тарифи та акції . Вибираєте вподобаний, зв’язуєтеся з оператором і вперед по безкрайніх просторах мережі Інтернет!

Для тих користувачів, хто втратив логін або пароль. Відновити їх легко і просто, якщо ви можете підтвердити свою особу. Оператор call-центру, зв’язатися з яким можна за номерами нижче задасть вам кілька запитань, і, якщо ви на них відповісте правильно, вишле нові дані авторизації на вашу електронну пошту.

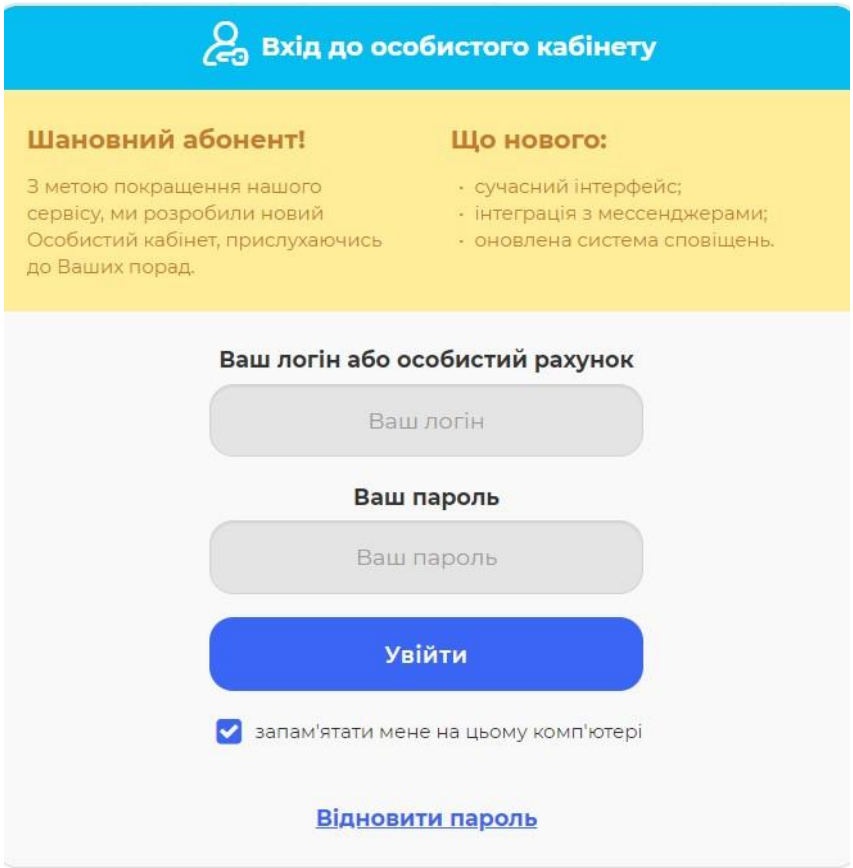

Рисунок 3.17 – Вхід до особистого кабінету

Підвал веб-сайту зберігає контактну информацію, також він включає основні пункти меню и контактну информацію. Также у цьому місці розташовується вбудована мапа, на якій можливо детально роздивитись, де

знаходиться офіс компанії (рис. 3.18).

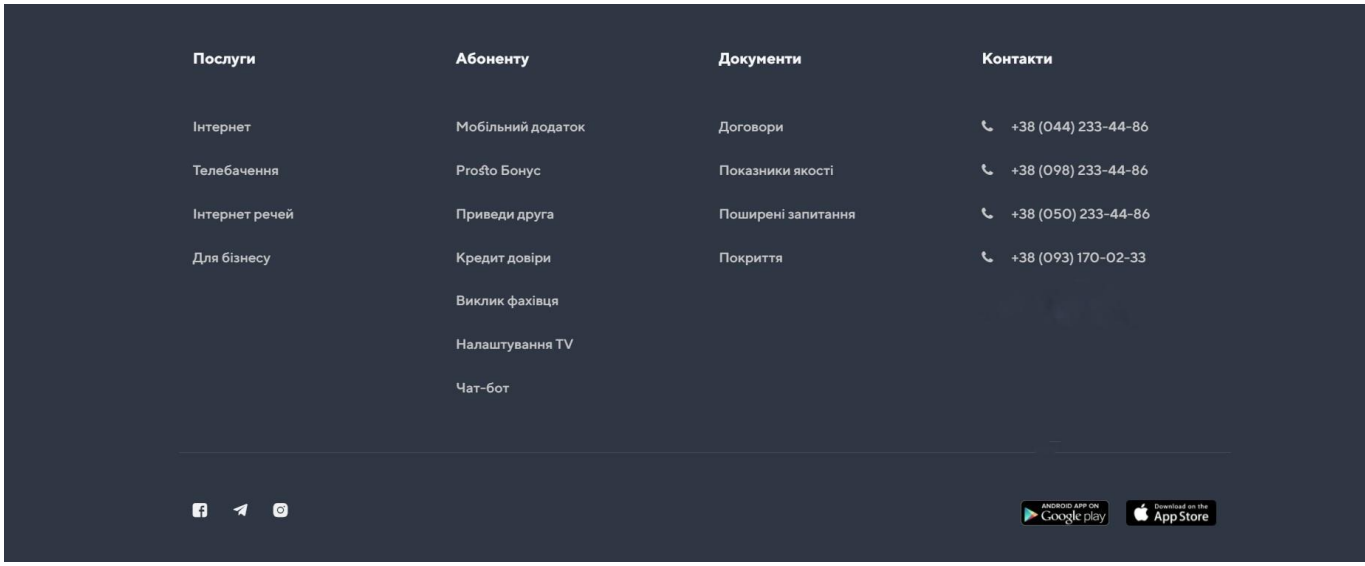

Рисунок 3.18 – Підвал веб-сайту

Весь дизайн веб-сайту виконаний в загальному і витонченому стилі та виглядає цілісно. Це забезпечується єдністю кольорової гами і невеликою гнучкостю шрифтів.

ВИСНОВКИ

Цей дипломний проект полягав у розробці інформаційного ресурсу у формі веб-сайту.

У наш час веб-сайт є ключовим атрибутом будь-якої організації, що поважає себе. Веб-сайт як представництво компанії в глобальній мережі Інтернет є одним з найважливіших джерел інформації про саму компанію. Але успіх будь-якого бізнесу залежить від того, наскільки ефективно компанія передає свою інформацію партнерам та клієнтам. У цій роботі ми розглянули основні теоретичні питання при створенні веб-сайтів - мовні та програмні засоби, що використовуються при розробці веб-ресурсів. Ми вивчали методи створення веб-сайтів, а саме метод ручного створення веб-сайту та використання систем управління вмістом (CMS). Нарешті, ми розробили логічну структуру та дизайн веб-сайту для MegatronLink. З огляду на цю структуру та дизайн, ми створили веб-сайт MegatronLink, використовуючи каскадні таблиці стилів HTML і CSS. Ми використовували різні плагіни для створення сучасного дизайну веб-сайту. Згодом ми опублікували веб-сайт в Інтернеті, протестували роботу веб-сайту та його відображення в різних браузерах та усунули виниклі проблеми. Результатом цієї роботи є повноцінний веб-сайт компанії MegatronLink, наповнений необхідним контентом.

Веб-сайт успішно виконує свої завдання з мінімальними експлуатаційними витратами.

Після завершення роботи над програмою було проведено повне тестування всієї програми.

В результаті всіх цих дій програмний продукт відповідає всім вимогам, встановленим замовником, є повністю функціональним, не потребує будь-яких модифікацій і готовий до використання.

СПИСКИ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. HTML & CSS - www.w3.org [Електронний ресурс] - Режим доступу: https://www.w3.org/standards/webdesign/htmlcss.html. – Дата звернення: 10.05.2021
2. Классификация сайтов в современном web-е [Электронный ресурс]. - Режим доступа: http://www.blog.astramg.ru/articles/1627/. – Дата звернення: 16.04.2021
3. Введение в JavaScript [Електронний ресурс] - https://developer.mozilla.org/ru/docs/Learn/JavaScript. – Дата звернення: 28.05.2021
4. Дизайн сайта [Електронний ресурс] - Режим доступу: https://www.webdesign-inspiration.com/. – Дата звернення: 03.05.2021
5. 15 Основних вимог до веб сайту - studfile.net [Електронний ресурс] - https://timeweb.com https://studfile.net/preview/6438407/page:3/. – Дата звернення: 27.04.2021
6. Що таке MySQL? - atlantic.net [Електронний ресурс] - Режим доступу: https://www.atlantic.net/what-is-mysql/. – Дата звернення: 07.05.2021
7. Джон Дакетт. «Javascript и jQuery. Интерактивная веб-разработка»
8. Введение. DOM в примерах. [Электронный ресурс]. - Режим доступа: http://javascript.ru/tutorial/dom/intro. – Дата звернення: 24.05.2021
9. Общая классификация CMS [Электронный ресурс]. - Режим доступа: http://www.solus.ru/articles_9.html. – Дата звернення: 30.04.2021
10. Дэвид Макфарланд. «Новая большая книга CSS»
11. Что такое SEO (СЕО)? Виды оптимизации сайтов. Внешняя и внутренняя оптимизация сайта [Электронный ресурс]. - Режим доступу: http://asbseo.ru/optimizaciya-i-prodvizhenie-bloga/chto-takoe-seovidyoptimizacii-sajtov.html. – Дата звернення: 23.04.2021
12. Внутренняя оптимизация сайта [Электронный ресурс]. - Режим доступа: http://seokleo.ru/inside_optimization/. – Дата звернення: 20.05.2021

13. Этан Браун. «Изучаем JavaScript. Руководство по созданию современных веб-сайтов»
14. CSS Tutorial [Електронний ресурс] - Режим доступу: https://www.w3schools.com/css/. – Дата звернення: 25.04.2021
15. Робин Никсон. «Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5»
16. JS-библиотека [Електронний ресурс] - Режим доступу: http://bouncejs.com/. – Дата звернення: 10.04.2021
17. Создание логотипов [Електронний ресурс] - Режим доступу: https://hipsterlogogenerator.com/. – Дата звернення: 12.05.2021

ДОДАТКИ

Додаток А Лістинг програмного коду

```
  $.flexslider = function(el, options) {
   var slider = $(el),
       vars = $.extend({}, $.flexslider.defaults, options),
       namespace = vars.namespace,
       touch = ("ontouchstart" in window) || window.DocumentTouch && document
 instanceof DocumentTouch,
       eventType = (touch) ? "touchend" : "click",
       vertical = vars.direction === "vertical",
       reverse = vars.reverse,
       carousel = (vars.itemWidth > 0),
       fade = vars.animation === "fade",
       asNav = vars.asNavFor !== "",
       methods = {};

   // Store a reference to the slider object
   $.data(el, "flexslider", slider);

   // Privat slider methods
   methods = {
    init: function() {
      slider.animating = false;
      slider.currentSlide = vars.startAt;
      slider.animatingTo = slider.currentSlide;
      slider.atEnd = (slider.currentSlide === 0 || slider.currentSlide === slider.last);
      slider.containerSelector = vars.selector.substr(0,vars.selector.search(' '));
      slider.slides = $(vars.selector, slider);
      slider.container = $(slider.containerSelector, slider);
      slider.count = slider.slides.length;
      // SYNC:
      slider.syncExists = $(vars.sync).length > 0;
      // SLIDE:
      if (vars.animation === "slide") vars.animation = "swing";
      slider.prop = (vertical) ? "top" : "marginLeft";
      slider.args = {};
      // SLIDESHOW:
      slider.manualPause = false;
      // TOUCH/USECSS:
      slider.transitions = !vars.video && !fade && vars.useCSS && (function() {
        var obj = document.createElement('div'),
            props = ['perspectiveProperty', 'WebkitPerspective', 'MozPerspective', 'OP
erspective', 'msPerspective'];
        for (var i in props) {
```

```
          if ( obj.style[ props[i] ] !== undefined ) {
            slider.pfx = props[i].replace('Perspective','').toLowerCase();
            slider.prop = "-" + slider.pfx + "-transform";
            return true;
          }
        }
        return false;
      }());
      // CONTROLSCONTAINER:
      if (vars.controlsContainer !== "") slider.controlsContainer = $(vars.controlsC
ontainer).length > 0 && $(vars.controlsContainer);
      // MANUAL:
      if (vars.manualControls !== "") slider.manualControls = $(vars.manualContro
ls).length > 0 && $(vars.manualControls);

      // RANDOMIZE:
      if (vars.randomize) {
        slider.slides.sort(function() { return (Math.round(Math.random())-0.5); });
        slider.container.empty().append(slider.slides);
      }

      slider.doMath();

      // ASNAV:
      if (asNav) methods.asNav.setup();

      // INIT
      slider.setup("init");

      // CONTROLNAV:
      if (vars.controlNav) methods.controlNav.setup();

      // DIRECTIONNAV:
      if (vars.directionNav) methods.directionNav.setup();

      // KEYBOARD:
      if (vars.keyboard && ($(slider.containerSelector).length === 1 || vars.multipl
eKeyboard)) {
        $(document).bind('keyup', function(event) {
          var keycode = event.keyCode;
          if (!slider.animating && (keycode === 39 || keycode === 37)) {
            var target = (keycode === 39) ? slider.getTarget('next') :
                         (keycode === 37) ? slider.getTarget('prev') : false;
            slider.flexAnimate(target, vars.pauseOnAction);
          }
```

```
      });
    }
    // MOUSEWHEEL:
    if (vars.mousewheel) {
      slider.bind('mousewheel', function(event, delta, deltaX, deltaY) {
        event.preventDefault();
        var target = (delta < 0) ? slider.getTarget('next') : slider.getTarget('prev');
        slider.flexAnimate(target, vars.pauseOnAction);
      });
    }

    // PAUSEPLAY
    if (vars.pausePlay) methods.pausePlay.setup();

    // SLIDSESHOW
    if (vars.slideshow) {
      if (vars.pauseOnHover) {
        slider.hover(function() {
          slider.pause();
        }, function() {
          if (!slider.manualPause) slider.play();
        });
      }
      // initialize animation
      (vars.initDelay > 0) ? setTimeout(slider.play, vars.initDelay) : slider.play();
    }

    // TOUCH
    if (touch && vars.touch) methods.touch();

    // FADE&&SMOOTHHEIGHT || SLIDE:
    if (!fade || (fade && vars.smoothHeight)) $(window).bind("resize focus", met
hods.resize);


    // API: start() Callback
    setTimeout(function(){
      vars.start(slider);
    }, 200);
  },
  asNav: {
    setup: function() {
      slider.asNav = true;
      slider.animatingTo = Math.floor(slider.currentSlide/slider.move);
      slider.currentItem = slider.currentSlide;
```

```
        slider.slides.removeClass(namespace + "active-
slide").eq(slider.currentItem).addClass(namespace + "active-slide");
        slider.slides.click(function(e){
          e.preventDefault();
          var $slide = $(this),
              target = $slide.index();
          if (!$(vars.asNavFor).data('flexslider').animating && !$slide.hasClass('acti
ve')) {
            slider.direction = (slider.currentItem < target) ? "next" : "prev";
            slider.flexAnimate(target, vars.pauseOnAction, false, true, true);
          }
        });
      }
    },
    controlNav: {
      setup: function() {
        if (!slider.manualControls) {
          methods.controlNav.setupPaging();
        } else { // MANUALCONTROLS:
          methods.controlNav.setupManual();
        }
      },
      setupPaging: function() {
        var type = (vars.controlNav === "thumbnails") ? 'control-thumbs' : 'control-
paging',
            j = 1,
            item;

        slider.controlNavScaffold = $('<ol class="'+ namespace + 'control-
nav ' + namespace + type + '"></ol>');

        if (slider.pagingCount > 1) {
          for (var i = 0; i < slider.pagingCount; i++) {
            item = (vars.controlNav === "thumbnails") ? '<img src="' + slider.slides.e
q(i).attr("data-thumb") + '"/>' : '<a>' + j + '</a>';
            slider.controlNavScaffold.append('<li>' + item + '</li>');
            j++;
          }
        }

        // CONTROLSCONTAINER:
        (slider.controlsContainer) ? $(slider.controlsContainer).append(slider.control
NavScaffold) : slider.append(slider.controlNavScaffold);
        methods.controlNav.set();
```

```
        methods.controlNav.active();

        slider.controlNavScaffold.delegate('a, img', eventType, function(event) {
          event.preventDefault();
          var $this = $(this),
              target = slider.controlNav.index($this);

          if (!$this.hasClass(namespace + 'active')) {
            slider.direction = (target > slider.currentSlide) ? "next" : "prev";
            slider.flexAnimate(target, vars.pauseOnAction);
          }
        });
        // Prevent iOS click event bug
        if (touch) {
          slider.controlNavScaffold.delegate('a', "click touchstart", function(event) {
            event.preventDefault();
          });
        }
      },
      setupManual: function() {
        slider.controlNav = slider.manualControls;
        methods.controlNav.active();

        slider.controlNav.live(eventType, function(event) {
          event.preventDefault();
          var $this = $(this),
              target = slider.controlNav.index($this);

          if (!$this.hasClass(namespace + 'active')) {
            (target > slider.currentSlide) ? slider.direction = "next" : slider.direction =
"prev";
            slider.flexAnimate(target, vars.pauseOnAction);
          }
        });
        // Prevent iOS click event bug
        if (touch) {
          slider.controlNav.live("click touchstart", function(event) {
            event.preventDefault();
          });
        }
      },
      set: function() {
        var selector = (vars.controlNav === "thumbnails") ? 'img' : 'a';
        slider.controlNav = $('.' + namespace + 'control-
nav li ' + selector, (slider.controlsContainer) ? slider.controlsContainer : slider);
```

```
      },
      active: function() {
        slider.controlNav.removeClass(namespace + "active").eq(slider.animatingTo
).addClass(namespace + "active");
      },
      update: function(action, pos) {
        if (slider.pagingCount > 1 && action === "add") {
          slider.controlNavScaffold.append($('<li><a>' + slider.count + '</a></li>'));
        } else if (slider.pagingCount === 1) {
          slider.controlNavScaffold.find('li').remove();
        } else {
          slider.controlNav.eq(pos).closest('li').remove();
        }
        methods.controlNav.set();
        (slider.pagingCount > 1 && slider.pagingCount !== slider.controlNav.lengt
h) ? slider.update(pos, action) : methods.controlNav.active();
      }
    },
    directionNav: {
      setup: function() {
        var directionNavScaffold = $('<ul class="' + namespace + 'direction-
nav"><li><a class="' + namespace + 'prev" href="#">' + vars.prevText + '</a></li
><li><a class="' + namespace + 'next" href="#">' + vars.nextText + '</a></li></ul
>');

        // CONTROLSCONTAINER:
        if (slider.controlsContainer) {
          $(slider.controlsContainer).append(directionNavScaffold);
          slider.directionNav = $('.' + namespace + 'direction-
nav li a', slider.controlsContainer);
        } else {
          slider.append(directionNavScaffold);
          slider.directionNav = $('.' + namespace + 'direction-nav li a', slider);
        }

        methods.directionNav.update();

        slider.directionNav.bind(eventType, function(event) {
          event.preventDefault();
          var target = ($(this).hasClass(namespace + 'next')) ? slider.getTarget('next')
: slider.getTarget('prev');
          slider.flexAnimate(target, vars.pauseOnAction);
        });
        // Prevent iOS click event bug
        if (touch) {
```

```
          slider.directionNav.bind("click touchstart", function(event) {
            event.preventDefault();
          });
        }
      },
      update: function() {
        var disabledClass = namespace + 'disabled';
        if (!vars.animationLoop) {
          if (slider.pagingCount === 1) {
           slider.directionNav.addClass(disabledClass);
          } else if (slider.animatingTo === 0) {
            slider.directionNav.removeClass(disabledClass).filter('.' + namespace + "p
rev").addClass(disabledClass);
          } else if (slider.animatingTo === slider.last) {
            slider.directionNav.removeClass(disabledClass).filter('.' + namespace + "n
ext").addClass(disabledClass);
          } else {
            slider.directionNav.removeClass(disabledClass);
          }
        }
      }
    },
    pausePlay: {
      setup: function() {
        var pausePlayScaffold = $('<div class="' + namespace + 'pauseplay"><a></a
></div>');

        // CONTROLSCONTAINER:
        if (slider.controlsContainer) {
          slider.controlsContainer.append(pausePlayScaffold);
          slider.pausePlay = $('.' + namespace + 'pauseplay a', slider.controlsContain
er);
        } else {
          slider.append(pausePlayScaffold);
          slider.pausePlay = $('.' + namespace + 'pauseplay a', slider);
        }

        // slider.pausePlay.addClass(pausePlayState).text((pausePlayState == 'pause'
) ? vars.pauseText : vars.playText);
        methods.pausePlay.update((vars.slideshow) ? namespace + 'pause' : namespa
ce + 'play');

        slider.pausePlay.bind(eventType, function(event) {
          event.preventDefault();
          if ($(this).hasClass(namespace + 'pause')) {
```

```
            slider.pause();
            slider.manualPause = true;
          } else {
            slider.play();
            slider.manualPause = false;
          }
        });
        // Prevent iOS click event bug
        if (touch) {
          slider.pausePlay.bind("click touchstart", function(event) {
            event.preventDefault();
          });
        }
      },
      update: function(state) {
        (state === "play") ? slider.pausePlay.removeClass(namespace + 'pause').add
Class(namespace + 'play').text(vars.playText) : slider.pausePlay.removeClass(nam
espace + 'play').addClass(namespace + 'pause').text(vars.pauseText);
      }
    },
    touch: function() {
      var startX,
        startY,
        offset,
        cwidth,
        dx,
        startT,
        scrolling = false;

      el.addEventListener('touchstart', onTouchStart, false);
      function onTouchStart(e) {
        if (slider.animating) {
          e.preventDefault();
        } else if (e.touches.length === 1) {
          slider.pause();
          // CAROUSEL:
          cwidth = (vertical) ? slider.h : slider. w;
          startT = Number(new Date());
          // CAROUSEL:
          offset = (carousel && reverse && slider.animatingTo === slider.last) ? 0 :
              (carousel && reverse) ? slider.limit -
 (((slider.itemW + vars.itemMargin) * slider.move) * slider.animatingTo) :
              (carousel && slider.currentSlide === slider.last) ? slider.limit :
              (carousel) ? ((slider.itemW + vars.itemMargin) * slider.move) * slider
.currentSlide :
```

```
                (reverse) ? (slider.last -
 slider.currentSlide + slider.cloneOffset) * cwidth : (slider.currentSlide + slider.clo
neOffset) * cwidth;
          startX = (vertical) ? e.touches[0].pageY : e.touches[0].pageX;
          startY = (vertical) ? e.touches[0].pageX : e.touches[0].pageY;

          el.addEventListener('touchmove', onTouchMove, false);
          el.addEventListener('touchend', onTouchEnd, false);
        }
      }

      function onTouchMove(e) {
        dx = (vertical) ? startX - e.touches[0].pageY : startX - e.touches[0].pageX;
        scrolling = (vertical) ? (Math.abs(dx) < Math.abs(e.touches[0].pageX -
 startY)) : (Math.abs(dx) < Math.abs(e.touches[0].pageY - startY));

        if (!scrolling || Number(new Date()) - startT > 500) {
          e.preventDefault();
          if (!fade && slider.transitions) {
            if (!vars.animationLoop) {
              dx = dx/((slider.currentSlide === 0 && dx < 0 || slider.currentSlide ===
slider.last && dx > 0) ? (Math.abs(dx)/cwidth+2) : 1);
            }
            slider.setProps(offset + dx, "setTouch");
          }
        }
      }

      function onTouchEnd(e) {
        if (slider.animatingTo === slider.currentSlide && !scrolling && !(dx === n
ull)) {
          var updateDx = (reverse) ? -dx : dx,
            target = (updateDx > 0) ? slider.getTarget('next') : slider.getTarget('prev')
;

          if (slider.canAdvance(target) && (Number(new Date()) -
 startT < 550 && Math.abs(updateDx) > 20 || Math.abs(updateDx) > cwidth/2)) {
            slider.flexAnimate(target, vars.pauseOnAction);
          } else {
            slider.flexAnimate(slider.currentSlide, vars.pauseOnAction, true);
          }
        }
        // finish the touch by undoing the touch session
        el.removeEventListener('touchmove', onTouchMove, false);
        el.removeEventListener('touchend', onTouchEnd, false);
```

```
        startX = null;
        startY = null;
        dx = null;
        offset = null;
      }
    },
    resize: function() {
      if (!slider.animating && slider.is(':visible')) {
        if (!carousel) slider.doMath();

        if (fade) {
          // SMOOTH HEIGHT:
          methods.smoothHeight();
        } else if (carousel) { //CAROUSEL:
          slider.slides.width(slider.computedW);
          slider.update(slider.pagingCount);
          slider.setProps();
        }
        else if (vertical) { //VERTICAL:
          slider.viewport.height(slider.h);
          slider.setProps(slider.h, "setTotal");
        } else {
          // SMOOTH HEIGHT:
          if (vars.smoothHeight) methods.smoothHeight();
          slider.newSlides.width(slider.computedW);
          slider.setProps(slider.computedW, "setTotal");
        }
      }
    },
    smoothHeight: function(dur) {
      if (!vertical || fade) {
        var $obj = (fade) ? slider : slider.viewport;
        (dur) ? $obj.animate({"height": slider.slides.eq(slider.animatingTo).height()}
, dur) : $obj.height(slider.slides.eq(slider.animatingTo).height());
      }
    },
    sync: function(action) {
      var $obj = $(vars.sync).data("flexslider"),
          target = slider.animatingTo;

      switch (action) {
        case "animate": $obj.flexAnimate(target, vars.pauseOnAction, false, true); br
eak;
        case "play": if (!$obj.playing && !$obj.asNav) { $obj.play(); } break;
        case "pause": $obj.pause(); break;
```

```
      }
    }
  }

  // public methods
  slider.flexAnimate = function(target, pause, override, withSync, fromNav) {
    if (!slider.animating && (slider.canAdvance(target) || override) && slider.is(":
visible")) {
      if (asNav && withSync) {
        var master = $(vars.asNavFor).data('flexslider');
        slider.atEnd = target === 0 || target === slider.count - 1;
        master.flexAnimate(target, true, false, true, fromNav);
        slider.direction = (slider.currentItem < target) ? "next" : "prev";
        master.direction = slider.direction;

        if (Math.ceil((target + 1)/slider.visible) -
 1 !== slider.currentSlide && target !== 0) {
          slider.currentItem = target;
          slider.slides.removeClass(namespace + "active-
slide").eq(target).addClass(namespace + "active-slide");
          target = Math.floor(target/slider.visible);
        } else {
          slider.currentItem = target;
          slider.slides.removeClass(namespace + "active-
slide").eq(target).addClass(namespace + "active-slide");
          return false;
        }
      }

      slider.animating = true;
      slider.animatingTo = target;
      // API: before() animation Callback
      vars.before(slider);

      // SLIDESHOW:
      if (pause) slider.pause();

      // SYNC:
      if (slider.syncExists && !fromNav) methods.sync("animate");

      // CONTROLNAV
      if (vars.controlNav) methods.controlNav.active();

      // !CAROUSEL:
      // CANDIDATE: slide active class (for add/remove slide)
```

```
      if (!carousel) slider.slides.removeClass(namespace + 'active-
slide').eq(target).addClass(namespace + 'active-slide');

      // INFINITE LOOP:
      // CANDIDATE: atEnd
      slider.atEnd = target === 0 || target === slider.last;

      // DIRECTIONNAV:
      if (vars.directionNav) methods.directionNav.update();

      if (target === slider.last) {
        // API: end() of cycle Callback
        vars.end(slider);
        // SLIDESHOW && !INFINITE LOOP:
        if (!vars.animationLoop) slider.pause();
      }

      // SLIDE:
      if (!fade) {
        var dimension = (vertical) ? slider.slides.filter(':first').height() : slider.comput
edW,
            margin, slideString, calcNext;

        // INFINITE LOOP / REVERSE:
        if (carousel) {
          margin = (vars.itemWidth > slider.w) ? vars.itemMargin * 2 : vars.itemMar
gin;
          calcNext = ((slider.itemW + margin) * slider.move) * slider.animatingTo;
          slideString = (calcNext > slider.limit && slider.visible !== 1) ? slider.limit
: calcNext;
        } else if (slider.currentSlide === 0 && target === slider.count -
 1 && vars.animationLoop && slider.direction !== "next") {
          slideString = (reverse) ? (slider.count + slider.cloneOffset) * dimension : 0;
        } else if (slider.currentSlide === slider.last && target === 0 && vars.animat
ionLoop && slider.direction !== "prev") {
          slideString = (reverse) ? 0 : (slider.count + 1) * dimension;
        } else {
          slideString = (reverse) ? ((slider.count - 1) -
 target + slider.cloneOffset) * dimension : (target + slider.cloneOffset) * dimension
;
        }
        slider.setProps(slideString, "", vars.animationSpeed);
        if (slider.transitions) {
          if (!vars.animationLoop || !slider.atEnd) {
            slider.animating = false;
```

```
            slider.currentSlide = slider.animatingTo;
          }
          slider.container.unbind("webkitTransitionEnd transitionend");
          slider.container.bind("webkitTransitionEnd transitionend", function() {
            slider.wrapup(dimension);
          });
        } else {
          slider.container.animate(slider.args, vars.animationSpeed, vars.easing, func
tion(){
            slider.wrapup(dimension);
          });
        }
      } else { // FADE:
        slider.slides.eq(slider.currentSlide).fadeOut(vars.animationSpeed, vars.easin
g);
        slider.slides.eq(target).fadeIn(vars.animationSpeed, vars.easing, slider.wrapu
p);
      }
      // SMOOTH HEIGHT:
      if (vars.smoothHeight) methods.smoothHeight(vars.animationSpeed);
    }
  }
  slider.wrapup = function(dimension) {
    // SLIDE:
    if (!fade && !carousel) {
      if (slider.currentSlide === 0 && slider.animatingTo === slider.last && vars.a
nimationLoop) {
        slider.setProps(dimension, "jumpEnd");
      } else if (slider.currentSlide === slider.last && slider.animatingTo === 0 &&
 vars.animationLoop) {
        slider.setProps(dimension, "jumpStart");
      }
    }
    slider.animating = false;
    slider.currentSlide = slider.animatingTo;
    // API: after() animation Callback
    vars.after(slider);
  }

  // SLIDESHOW:
  slider.animateSlides = function() {
    if (!slider.animating) slider.flexAnimate(slider.getTarget("next"));
  }
```