ЗВО УНІВЕРСИТЕТ КОРОЛЯ ДАНИЛА

Факультет суспільних та прикладних наук

Кафедра інформаційних технологій

на правах рукопису

Німець Павло Васильович

УДК 004.738.5

Розробка веб-сайту для фахівця з веб-дизайну

Спеціальність 121 – «Інженерія програмного забезпечення»

Кваліфікаційна робота на здобуття кваліфікації бакалавр

Нормоконтроль

_____Стисло О.В.

(підпис, дата, розшифрування підпису)

Студент

_____Німець П.В.

(підпис, дата, розшифрування підпису)

Допускається до захисту

Завідувач кафедри

_____к.т.н., доц. Ващишак С.П.

(підпис, дата, розшифрування підпису)

Керівник роботи

асистент каф. ІТ

_____ Витвицький Р.І.

(підпис, дата, розшифрування підпису)

Івано-Франківськ – 2024

ЗВО УНІВЕРСИТЕТ КОРОЛЯ ДАНИЛА

Факультет суспільних та прикладних наук

Кафедра інформаційних технологій

Освітній ступінь: «бакалавр»

Спеціальність: 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____

«____ » _____ 2024 року

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Німець Павло Васильович

(прізвище, ім’я, по батькові)

1. Тема кваліфікаційної роботи:

Розробка веб-сайту для фахівця з веб-дизайну

керівник роботи:

Витвицький Роман Ігорович, асистент каф. ІТ

затверджена наказом вищого навчального закладу від « 12 » березня 2024 року № 19/1

2. Термін подання студентом роботи 05.06.2024

3. Вихідні дані роботи: Python, Django, HTML, CSS, JavaScript, Vue.js

4. Зміст кваліфікаційної роботи (перелік питань, які потрібно розробити)

1. Опис наявних аналогів
2. Розробка прототипу сайту
3. Реалізація сайту

5. Дата видачі завдання 14.03.2024

КОНСУЛЬТАНТИ РОЗДІЛІВ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Розділ	Консультант (прізвище, ініціали та посада)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1.	Огляд та аналіз існуючих аналогів	24.03.2024	Виконано
2.	Проектування прототипу сайту	10.04.2024	Виконано
3.	Розробка сайту	24.04.2024	Виконано
4.	Оформлення пояснювальної записки	15.05.2024	Виконано
5.	Оформлення графічного матеріалу та підготовка до захисту роботи	25.05.2024	Виконано

Студент _____ Німець П.В.

(підпис) (прізвище та ініціали)

Керівник роботи _____ Витвицький Р.І.

(підпис) (прізвище та ініціали)

Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

АНОТАЦІЯ

В результаті роботи був реалізований сайт-візитка для веб-дизайнера з можливість редагування портфоліо через адмін панель Django, що збільшило зручність в оновленні робіт.

В першому розділі нашого дослідження був проведений детальний аналіз конкурентів де було виявлено їхні сильні та слабкі сторони. Це дозволило зробити правильний аналіз та не допуститися таких самих помилок.

В другому розділі розповідалося про етапи створення веб-сайту та про вибір мов програмування для його створення.

Третій розділ присвячений створенню прототипу сайту його структурі та реалізації. За допомогою таких мов програмування як: Python, фреймворк Django, HTML, CSS, JavaScript та Vue.js .

КЛЮЧОВІ СЛОВА: PYTHON, DJANGO, HTML, CSS, JAVASCRIPT, VUE.JS.

SUMMARY

As a result of the work, a business card site for a web designer was implemented with the ability to edit the portfolio through the Django admin panel, which increased the convenience of updating works.

In the first section of our research, a detailed analysis of competitors was conducted to identify their strengths and weaknesses. This allowed us to make the right analysis and avoid the same mistakes.

The second section described the stages of website development and the choice of programming languages for its creation.

The third section is devoted to the creation of a website prototype, its structure and implementation. Using such programming languages as: Python, Django framework, HTML, CSS, JavaScript and Vue.js.

KEYWORDS: PYTHON, DJANGO, HTML, CSS, JAVASCRIPT, VUE.JS.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

HTML – Hyper Text Markup Language

CSS – Cascading Style Sheets

SQL – Structured Query Language

ВСТУП

Актуальність теми. Сьогодні веб-дизайн є невід'ємною частиною будь-якого онлайн-бізнесу. Сайт - це візитна картка будь-якої компанії чи приватного фахівця, його віртуальний представник в Інтернеті. Від того, наскільки сайт якісний, зручний та привабливий, залежить перше враження користувачів, а значить і їхнє рішення про співпрацю.

Мета роботи. дослідити актуальність теми "Сайт-візитка для веб-дизайнера" та розробити власний сайт.

Об’єкт роботи. Сайт-візитка.

Предмет роботи. Сайт-візитка з можливість логіну адміністратора сайту та редагування інформації на сайті.

Завдання роботи. Відповідно до вибраної теми в роботі покладені такі задачі як: а) проаналізувати сайти конкурентів; б) виділити основні функції на сайту; в) розробити дизайн; г) написати сайт.

Методи роботи. Для вирішення поставленого завдання були використані мова програмування Python, база даних MySQL, фреймворк Django, а також HTML і CSS.

Результати роботи. Результатом роботи буде створений сайт-візитка з можливістю логіну адміністратору сайту для подальшого його наповнення інформацією.

Апробація результатів дослідження. Матеріали кваліфікаційної роботи були представлені на Всеукраїнській науково-практичній інтернет-конференції «IT екосистема: цифровізація бізнес-процесів в умовах війни», яка відбулася 23-24 листопада 2023 року в Університеті Короля Данила.

Структура роботи. Розділи – 3. Загальний обсяг основної частини – 60 сторінок. Список використаних джерел – 22.

РОЗДІЛ 1. САЙТИ-ВІЗИТКИ: СУТНІСТЬ ТА ОСОБЛИВОСТІ

1.1 Що таке сайт-візитка? Які його функції?

Сайт-візитка - це невеликий веб-сайт, який містить основну інформацію про компанію, організацію або приватну особу. Він зазвичай складається з однієї або декількох сторінок і містить наступну інформацію: логотип, контактну інформацію, опис діяльності, перелік послуг та портфоліо.

Сайт візитка має також багато функцій, наприклад представлення в інтернеті. Бо сайт візитка – візитна картка вашого бренду в інтернеті. Він дає можливість потенційним клієнт дізнатися про вас. Також він виконує роль інформатора який несе цінну інформацію про вас цільовій аудиторії. Це може бути інформація про вас та вашу компанію, послуги які ви надаєте, ваш досвід роботи, освіту, навички тощо. Сайт-візитка дає можливість потенційним клієнтам зв’язатися з вами в будь який зручний для них спосіб, на сайті ви можете залишити безліч способів зв’язку з вами, від електронної адреси до посилання на Instagram. Також навість такого сайту значно підвищує авторитет та довіру. Це показує, що ви серйозно ставитесь до своєї справи. І ще сайт візитку можна використовувати для збору даних про ваших клієнтів, маючи форму звернень через email на сайті, або можливість реєстрації аккаунта, можна в подальшому використовувати ці дані для маркетингу. І ще сайт візитка може використовуватися як портфоліо.

1.2 Переваги та недоліки сайтів-візиток

Перш ніж створювати сайт-візитку, важливо зважити всі його переваги та недоліки. В нього є декілька переваг. Перша перевага - це ціна. Створення сайту-візитки набагато дешевше ніж створення повноцінного сайту. Друге перевага - це простота у створенні. Не обов’язково бути програмістом щоб

написати власний сайт, зараз існує багато конструкторів сайтів які дозволяють створи будь який сайт без знань програмування. Сайт-візитка це переважно одно сторінковий сайт який буде не важко зробити навіть в конструкторі сайтів.

Тепер розберемо недоліки сайту-візитки. Перший недолік – це обмежений функціонал. Сайт-візитка не може мати складну структуру або бути багатофункціональним. Він підходить тільки для представлення основної інформації про вас або ваш бізнес. Другий недолік – це невеликий обсяг інформації на сайті. Якщо ви захочети створити такий сайт для себе або для свого бізнесу то вам треба розуміти, що на такому сайті не можна буде реалізовано багато функціоналу. І останній недолік – це необхідність постійного оновлення. Інформація на такому сайті має бути постійно бути актуальна і нова. Якщо ви не будете часто оновлювати веб-сайт він може здатися занедбаним та неактуальним для користувача.

1.3 Огляд та аналіз існуючих аналогів і конкурентів

При створенні сайту-візитки важливо провести аналіз конкурентів. Це допоможе вам зрозуміти, що пропонують інші, визначити їхні сильні та слабкі сторони, а також знайти унікальні особливості, які допоможуть нам виділитися на фоні конкурентів. Для аналізу я підібрав декілька сайтів які на мою думку мають правильну структуру сайту-візитки. Перший сайт який я хочу розібрати це сайт-візитка українського UI/UX дизайнера Андрія Боженка.

Зайшовши на сайт Андрія перше, що ми бачимо це слоган та короткий опис діяльності автора (рис. 1.1).

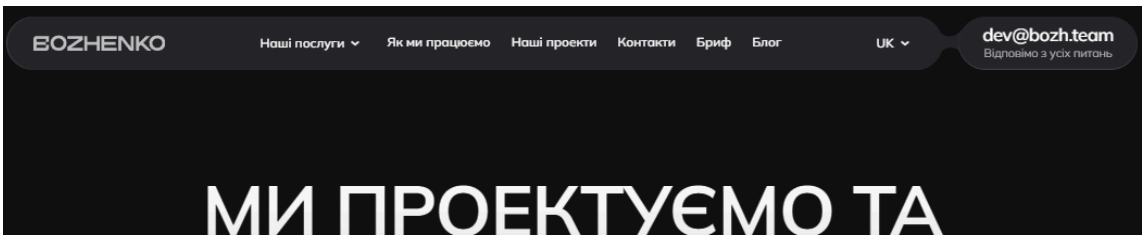


Рисунок 1.1 – Слоган з описом

В хедері з лівого боку ми бачимо логотип компанії (рис. 1.2).

Рисунок 1.2 – Логотип

Крім логотипа в хедері розміщені посилання на інформацію яка допоможе клієнту ознайомитися з автором, дізнатися про умови співпраці, минулі проєкти, контакти і таке інше (рис. 1.1).

Гортаючи сайт, одразу після слогана ми бачимо блок з послугами які надає Андрій (рис. 1.3).

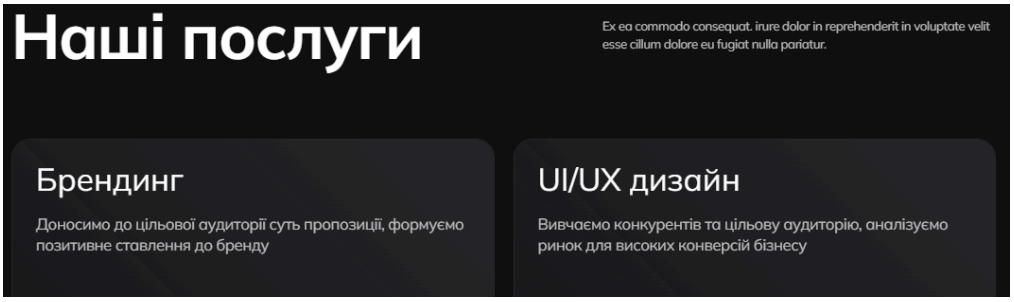


Рисунок 1.3 – Послуги

Після послуг, йде блок в якому розповідається як ведеться робота автора із замовником (рис. 1.4).

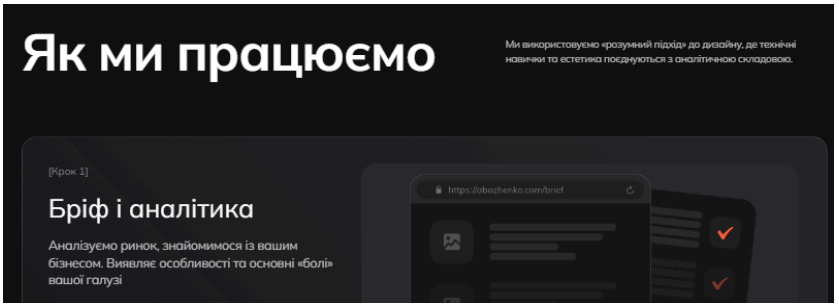


Рисунок 1.4 – Як ми працюємо

Дуже добре, що автор одразу детально розписує, яка робота буде проводитися на кожному етапі.

Передостанній блок присвячений підбірці часто задаваних питань клієнтів автору (рис. 1.5).

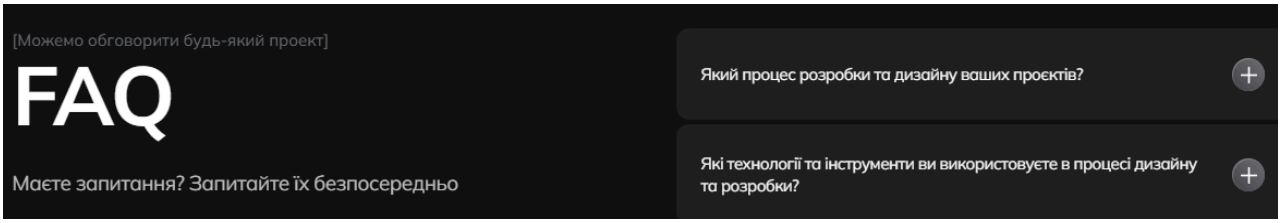


Рисунок 1.5 – FAQ

Хороше рішення створити такий блок, клієнт може прочитати відповідь на питання яке його могло бентежити і в подальшій роботі автор не буде витрачати час на обговорення питання, а зможе потрати цей час на клієнта.

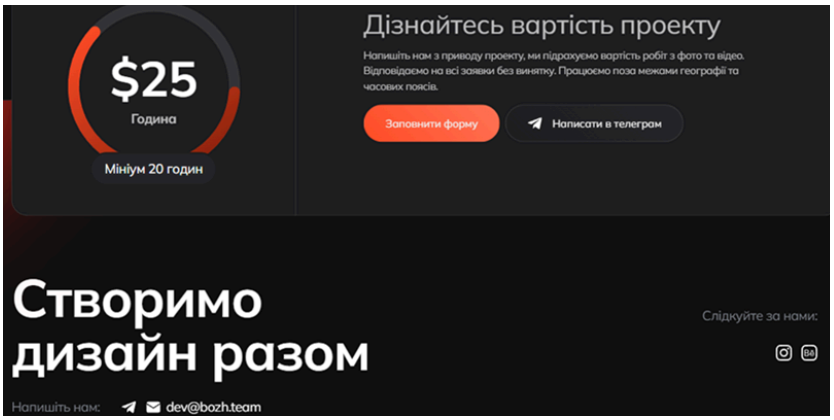


Рисунок 1.6 – Футер

І останній блок на сайті Андрія це футер в якому є форма зворотного зв’язку в якій можна дізнатися вартість проєкту, заповнивши її. А також є прямі посилання на комунікацію з автором і посилання на соціальні мережі (рис. 1.6).

Також я хочу розібрати ще один сайт, а саме сайт Мілани Мамедової. Її сайт дещо простіший за сайт Андрія, але це не робить його гіршим, навпаки він є простішим в сприйняті звичайному клієнтові.

Зайшовши на сайт Мілани, перше що ми бачимо це заголовок «UI/UX Designer», що дає нам зразу зрозуміти що це сайт дизайнера і також з низу текст що закликає до дії, а саме переглянути роботи авторки (рис. 1.7).

Рисунок 1.7 – Сайт Мілани

В хедері з лівого боку ми бачимо навігаційні посилання, з права ми бачимо посилання на соціальні мережі авторки та по центру ім’я та прізвище, яке використовується замість логотипа або є ним.

Гортаючи вниз ми бачимо роботи авторки і наводячись на них можна побачити, що це за проєкт і коли він був зроблений (рис. 1.8).

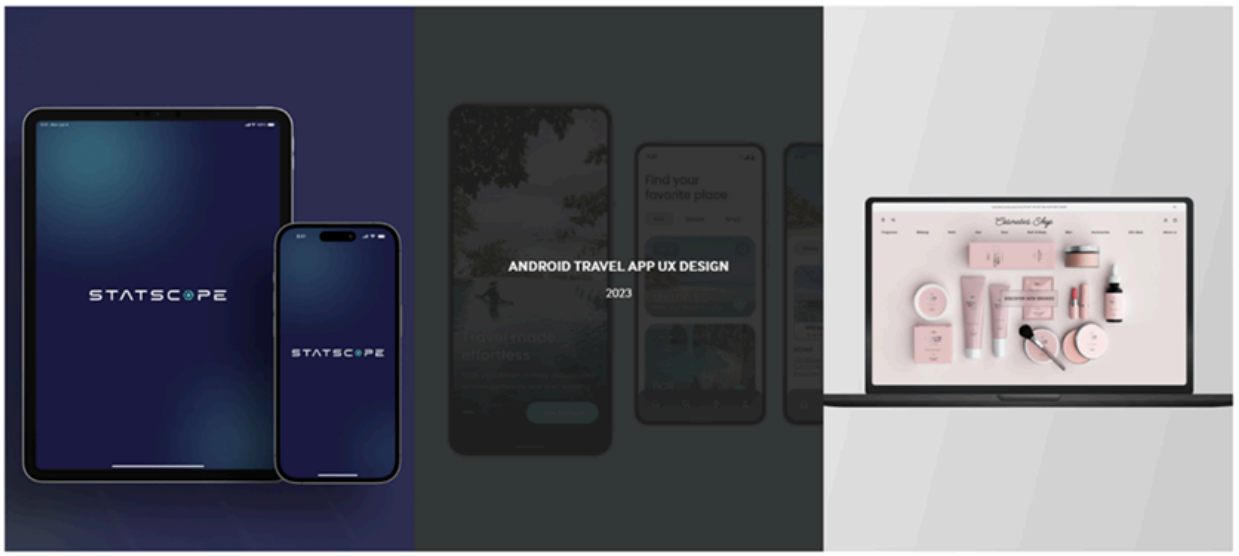


Рисунок 1.8 – Проєкти Мілани

Дата останніх проєктів це важливо, так клієнт може оцінювати навички автора роботи та також бачити його прогрес якщо автор має у своєму портфоліо старі роботи.

Горнувши нижче нас зустрічає футер в якому розміщенні посилання на соціальні мережі (рис. 1.9).

Рисунок 1.9 – Футер

Вернувшись на початок сайту і натиснувши на кнопку «About» нас перекине но нову сторінку на якій буде розповідь про авторку (рис. 1.10).

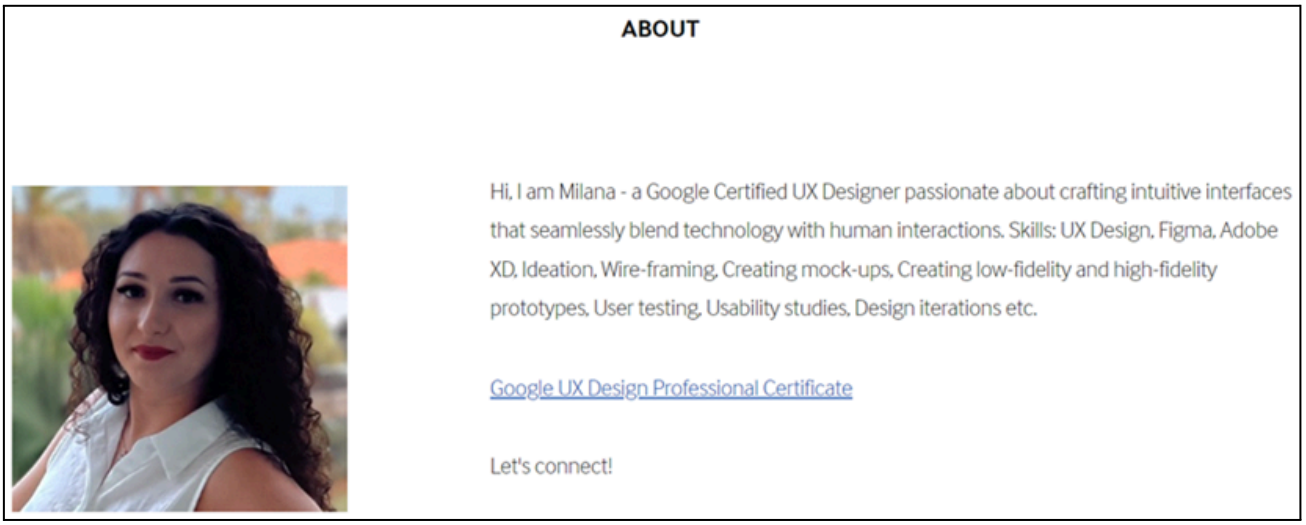


Рисунок 1.10 – Про авторку

Сподобалося, що таке є на сайті, бо в минулого автора такого не було, а було б цікаво дізнатися про його навички та досвід.

І на останок ми натиснемо на кнопку «Contact» яка нас перекине на сторінку з посиланнями на соціальні мережі та форму зворотного зв’язку з авторкою (рис. 1.11).

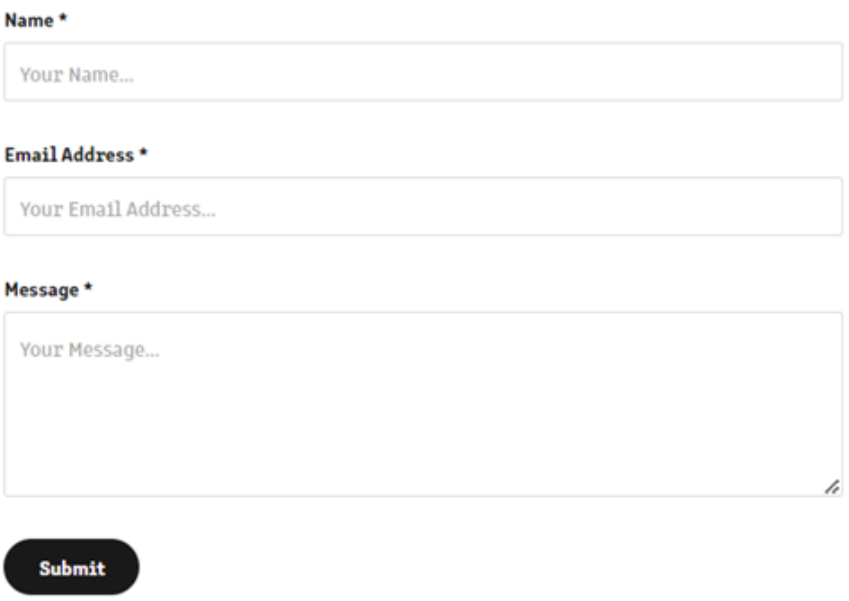


Рисунок 1.11 – Форма зворотного зв’язку

1.4 Постановка задачі

На основі аналізу сайтів Андрія та Мілани, ось кілька ключових моментів, які було враховано перед створенням сайту:

– чітке позиціонування: на сайті має бути чітко й лаконічно пояснено, чим ми займаємося та яку цінність пропонуємо своїм клієнтам. Слоган, опис діяльності та інші елементи, це ті інструменти які дадуть одразу дати зрозуміти цільовій аудиторії, хто ми та що можете їм запропонувати;

– зручна навігація: структура сайту повинна бути логічною та інтуїтивно зрозумілою. Використаємо легке меню з посиланнями на всі важливі розділи сайту.

– демонстрація робіт: портфоліо обов’язково буде на сайті, щоб клієнти змогли оцінити навички та досвід.

Висновки до розділу 1

У розділі було проведено дослідження про сайти-візитки з урахуванням усіх їхніх переваг та недоліків, а також був проведений аналіз існуючих

аналогів і конкурентів на ринку. Можна зробити висновок, що вони є ефективним інструментом для представлення компанії або особистості в Інтернеті, зокрема для привернення уваги потенційних клієнтів та партнерів. Вони досить доступні у створенні і мають простий інтерфейс для користувача. Однак важливо пам'ятати про обмежену функціональність та необхідність регулярного оновлення контенту для збереження актуальності сайту. Таким чином, сайти-візитки є ефективним інструментом, але вимагають уважного підходу до їх використання.

РОЗДІЛ 2. ПРОЄКТУВАННЯ САЙТУ

2.1 Етапи створення веб-сайту

Розробка веб-сайту складається з декількох основних процесів таких як: підготовка, проєктування, розробка дизайну, вертка, програмування, тестуванні і додаткові роботи (рис. 2.1).

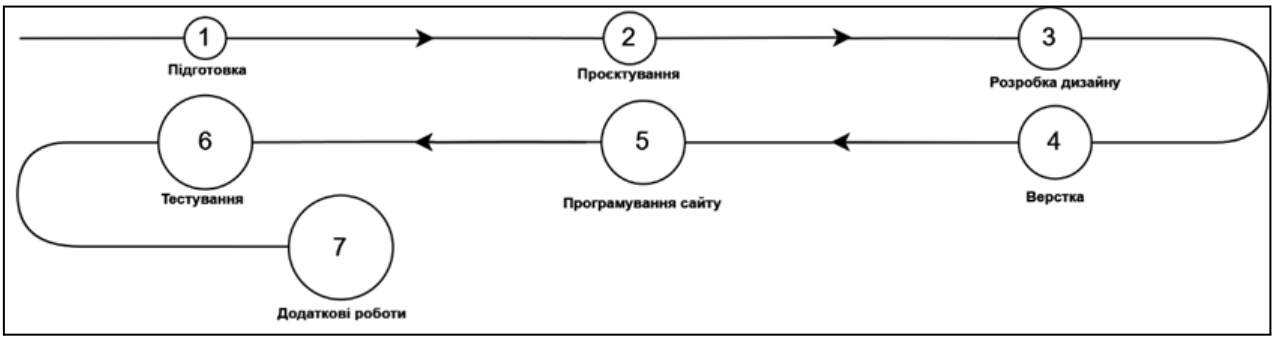


Рисунок 2.1 – Етапи створення веб-сайту

Перший етап у створенні сайту - це підготовка, яка передбачає ретельне планування та визначення основних вимог до сайту Вимог може бути багато, але потрібно визначити основні 3 вимоги, щоб легше було в подальшому. Перша вимога – це визначитися з ціллю сайту. Потрібно дати чітку відповідь для чого потрібен сайт, якщо з цим проблем немає можна переходити до другої вимоги. Друга вимога – це тип сайту. Сайтів дуже багато і у всіх різний функціонал. Це може бути інформаційний ресурс, електронний магазин, блог тощо. Тому тут слід добре подумати який сайт потрібен. І остання третя вимога – це визначитися з цільовою аудиторією.

Ми визначилися ще в попередньому розділі, з ціллю і типом нашого сайту. Наша ціль і завдання - це просування власного бренду. Ми також знаємо, що для наших вимог нам потрібен сайт-візитки, він найкраще підходить для нас і наших потреб. Наша цільова аудиторія - це бізнеси та компанії, які шукають

професійний дизайн для себе. Також наша аудиторія – це фрілансери та підприємці, які потребують створення власного бренду.

Можна тепер приступити і до проєктування сайту. При проєктуванні сайту слід приділити увагу структурі сайту. Структура – це логіка сайту. Сайт має бути логічним та простим в користуванні. Навігація має бути зрозумілою.

Також потрібно визначитися з основними розділами на сайті. На нашому сайт буде небагато інформації, але її буде достатньому щоб клієнт зрозумів чи підходимо ми йому чи ні. На сайті буде представлено основна інформація про нас, а саме: ім’я, логотип, короткий опис діяльності, контактна інформація, приклади робіт та за яким принципом ми працюємо. На мою думку цієї інформації достатньо.

Після того як структура сайту побудована, можна приступити і до розробка дизайну. Розробка дизайну – це дуже важливий етап, він є ключовим, оскільки саме візуальний вигляд веб-сторінок дуже сильно впливає на перше враження від сайту. В попередньому розділі ми розглядали сайти наших конкурентів, там ми більше приділяли увагу функціональній частині сайту, а не його вигляду. Щоб створити привабливий і найголовніше зручний для користувача дизайн, потрібно враховувати не лише функціонал, але й естетику та ергономіку. При розробці дизайну необхідно зосередитися на створенні сайту з привабливим і сучасним зовнішнім виглядом, який буде відповідати нашому бренду та стилістиці, а також буде забезпечувати зручну навігацію та легкий доступ до всього, що є на сайті.

Як тільки розробка дизайну буде завершена, вам необхідно перейти до етапу верстки. Верстка - це процес перетворення графічного дизайну код який зможе відтворити комп’ютер і побачити браузер. Є декілька етапів верстки. Перший етап – це розбір макету. Той хто буде верстати сайт, він має проаналізувати дизайн та розбити його на окремі елементи, такі як заголовки, зображення, кнопки, посилання тощо. Другий етап – це створення структури. На цьому етапі розробляється структура веб-сторінки. Кожний елемент сайту перетворюється на відповідний програмний код. Після того як побудований

каркас нашого сайту, наступає третій етап – це етап стилізації. Після того верстальник побудував каркас, стилізував його він приступить до четвертого етапу – це адаптація. Адаптивний дизайн потрібен для того, щоб веб-сторінка коректно відображалася на різних пристроях.

Все про що я тільки що розповідав - це є front end розробка сайту, тобто це все, що бачить користувач, а сайти також мають ще back end, те що користувач не бачить. Back end розробка включає в себе багато різних технологій. Back end розробка відповідає за управління серверною частиною веб-сайту. Розробка серверної частини сайту є ключовою складовою будь-якого сайту, оскільки вона забезпечує роботу всіх функцій, що необхідні для взаємодії користувача з сайтом. Є декілька основних функцій які виконує back-end. Перша функція – це управління базою даних. Друга функція – це безпека. Якщо на сайті реалізована реєстрація користувачів то база даних буде відповідати за обробку аутентифікації та авторизації користувачів. В бази даних дуже багато функцій і застосувань.

Останній і також дуже важливий пункт - це наповнення сайту контентом та тестування. Наш сайт потрібно наповнити якісним контентом для наших майбутніх клієнтів, виставити актуальні роботи та проєкти, та все протестувати. Після публікації сайту в інтернет, за ним слід буде деякий час стежити, щоб не виникло ніяких непередбачуваних проблеми які не були виявлені на етапі тестування.

2.2 Вибір мов програмування та фреймворку

Для розробки веб-сайту потрібно визначитися з мовами програмування серверної частини сайту і клієнтської сторони. Для клієнтської сторони, тобто усе, що буде бачити та з чим взаємодіятиме користувач, коли браузер завантажує сторінку ми будемо використовувати мову розмітки HTML, мову стилів CSS, а також JavaScript з використанням популярного фреймворку Vue.js. Vue.js – це фреймворк для створення користувацьких інтерфейсів. Він

забезпечує гнучкість, продуктивність та зручність розробки. Для серверної частини ми будемо використовувати мову програмування Python та його фреймворк Django. HTML або HyperText Markup Language, є основою для створення веб-сторінок. Він визначає структуру та вміст сторінок через розмітку, яка вказує браузерам, як відображати текст, зображення та інші елементи.

Рисунок 2.2 – Мова розмітки HTML

HTML не є мовою програмування, оскільки не містить логіки чи алгоритмів, але він є необхідним для побудови основи будь-якого веб-сайту. В парі разом з HTML використовується CSS або Cascading Style Sheets. CSS є мовою стилів, яка використовується для оформлення HTML-документів. CSS дозволяє веб-розробникам контролювати вигляд тексту, розмітки та інтерактивних елементів на веб-сторінці.

Рисунок 2.3 – Мова стилів CSS

Він відіграє ключову роль у створенні візуально привабливих і функціональних веб-сайтів. І на завершення до цього всього у нас є JavaScript з власним фреймворком Vue.js. Вона є потужною і гнучкою мовою програмування, яка використовується переважно для розробки клієнтської частини веб-сайтів.

Рисунок 2.4 – Мова програмування JavaScript

Хоча зараз вона також використовується у серверній розробці, мобільних додатках і навіть у програмуванні додатків для настільних комп'ютерів. Її головна мета - зробити веб-сторінку динамічною та інтерактивною. І на доповнення JS йде ще потужний і гнучкий фреймворк Vue.js. ін використовується для створення клієнтської частини веб-сайтів і виконується в браузері користувача. Однак він також знаходить застосування у серверній розробці, мобільних додатках та навіть у програмуванні додатків для настільних комп'ютерів. Мета Vue.js така сама як в JavaScript це зробити веб-сторінку динамічною та інтерактивною, дозволяючи взаємодіяти з користувачем без проблем. Вони забезпечують реактивну модель програмування, яка автоматично оновлює відображення при зміні даних, а також пропонують широкий спектр інструментів і бібліотек для розширення їх можливостей та спрощення розробки.

Python є однією з найпопулярніших мов програмування, що використовується для розробки серверної частини веб-сайтів завдяки своїй простоті, гнучкості та широкому спектру фреймворків.

Рисунок 2.5 – Мова програмування Python

Python славиться своєю легкістю читання та простотою синтаксису. Це робить мову доступною для новачків, а також зручною для досвідчених програмістів. Python має велику стандартну бібліотеку, яка включає утиліти для різних завдань у веб-розробці, таких як обробка даних, робота з датами, файлами та інтернет-протоколами. Це дозволяє розробникам використовувати готові модулі замість написання додаткового коду. Python пропонує різні фреймворки для веб-розробки, які значно спрощують процес розробки складних додатків. Найпопулярнішими серед них є Django.

Рисунок 2.6 – Фреймворк Django

Django - це високорівневий Python веб-фреймворк, який спрощує та прискорює процес створення комплексних, базованих на даних веб-сайтів. Django має в собі все необхідне для створення сайту. Має засоби взаємодії з базами даних, інструменти налагодження, адміністративний сайт, який можна використовувати для роботи зі збереженими в базі даними, а також веб-сервер. Django має одну з найкращих документацій серед відкритих інструментів розробки, що робить його доступним для новачків та корисним для досвідчених

розробників. Ці особливості роблять Django чудовим вибором для проектів різної складності, від малих сайтів до великих інтернет-платформ.

Висновки до розділу 2

Отже, на основі прочитаних двох попередніх підрозділів можна зробити наступний висновок. По-перше, перед початком роботи над веб-сайтом необхідно чітко визначити його цілі та завдання, а також його аудиторію. Це допоможе спрямувати всі зусилля на досягнення конкретних цілей і забезпечити ефективність проєкту. По-друге, структура сайту та його функціонал повинні бути максимально зрозумілими для користувача. Проста та логічна навігація сприяє зручному використанню сайту та позитивному враженню від візиту на нього.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ПРОЄКТУ

3.1 Прототип сайту

Для початку щоб написати будь-який сайт потрібно розробити його дизайн, щоб розуміти що ми хочемо бачити в кінцевому результаті і що ми маємо відтворити. В попередніх розділах ми вже обговорювали і дивилися декілька сайтів-візиток, ми підкреслили плюси та мінуси веб-сайтів, а тепер нам потрібно розробити власний дизайн який не буде схожий на інші та буде індивідуальним і підходити нас і нашому проєкту.

Для розробки дизайну нашого майбутнього сайту було обрано векторний онлайн-сервіс розробки інтерфейсів – Figma[19].

Рисунок 3.1 – Логотип Figma

Figma є одним з найпопулярніших векторних онлайн-сервісів для дизайну інтерфейсів та прототипування з багатьма перевагами(рис. 3.1).

Перша перевага – це не потрібно встановлювати ніяких програм на комп’ютер, можна працювати прямо в браузері. Це робить Figma легкою та доступною для користувачів з будь-якого пристрою. Друге перевага – це що вона дозволяє користувачам спільно працювати над один проектом в реальному

часі. Це корисно для команд, які віддалено. Усі зміни вносяться миттєво, що сприяє ефективній спільній роботі.

Третя перевага – це те що вона безкоштовна. Це безкоштовний онлайн-сервіс з великим функціонал та простим в освоєнні для новачків у веб-дизайні.

Щоб почати працювати з Figma потрібно зайти на їхній сайт, пройти реєстрацію і після чого Ви зможете працювати в редакторі. Після реєстрації було створено новий дизайн і почав створювати робочу область або Frame.

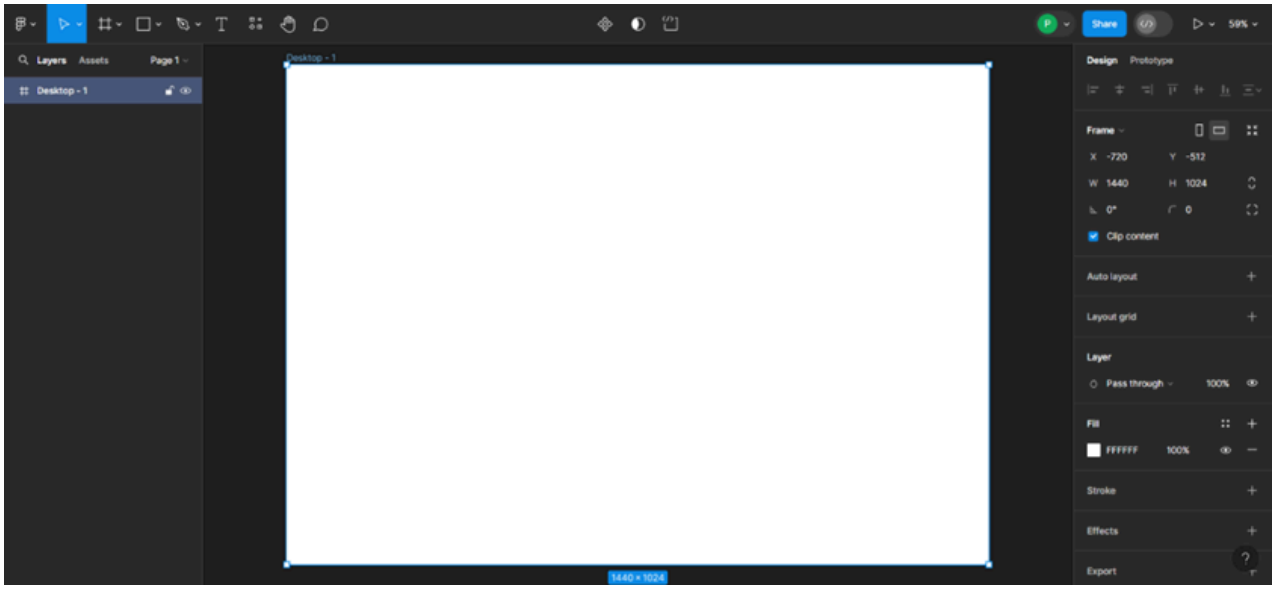


Рисунок 3.2 – Desktop Figma

Frame з англійської перекладається як рамка, каркас, скелет. Якщо спростити, то фрейми можна порівняти з полотном художника або робочою областю. На рисунку 3.2 ми бачимо біле полотно з розмірами 1440x1024. Цей розмір полотна є стандартний розмір для десктопних пристроїв. Висоту під час дизайну сайту ми будемо змінювати для того щоб помістити ввесь контент який має бути на сайті, це робиться тому, що наш сайт буде односторінковий.

Після створення створення екрану в якому ми будемо працювати, я приступив до дизайну блоку у верхній частині нашого сайту, а саме header. В header я хочу бачити блок навігації, посилання на соціальні мережі та логотип.

Рисунок 3.3 – Header

На рисунку 3.3 ми бачимо, що логотип розміщений з крайнього лівого боку, навігаційні кнопки які будуть допомагати користувачам у навігації на сайті розміщені по центру, а також посилання на соціальні мережі розміщенні з крайнього правого боку.

Після хедера на сайті буде блок з привітанням і інформацією про автора.

Рисунок 3.4 – Page content

На рисунку 3.4 ми бачимо привітання та розповідь інформацію про автора сайту, також є список послуг який надає автор та його фото.

Після знайомства з автором добре би було ознайомитися з його роботами, щоб оцінити який він спеціаліст. Тому одразу після блоку з привітання, ми робимо блок з роботами автора.

На рисунку 3.5 ми можемо побачити блок робіт який містить в собі останні актуальні роботи автора які зможуть показати його актуальні навички. В цьому блоці 4 секції 3 з яких мають містити фото роботі, назви, кнопку з посиланням на роботу та для кого та робота була виконана. В 4 секції буде кнопка для зв’язку з автором.

Рисунок 3.5 – Роботи

Після портфоліо автора буде блок з послугами. Для того щоб клієнт зайвий раз не запитував чи автор зможе виконати ту чи іншу роботу (рис.3.6).

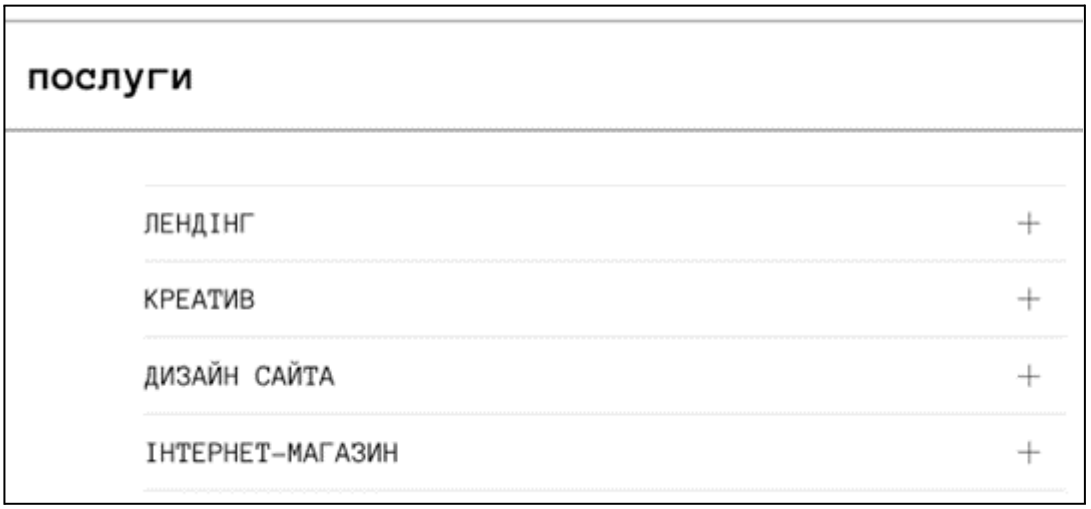


Рисунок 3.6 – Послуги

Цей блок розбитий на 4 секції, ці секції будуть інтерактивні, після натискання на кнопку буде відбуватися анімація по типу “Drop down menu button” і буде можливість більш детально прочитати про послугу.

Передостанній блок буде про етапи роботи. Клієнт зможе ознайомитися за яким принципом працює автор (рис. 3.7).

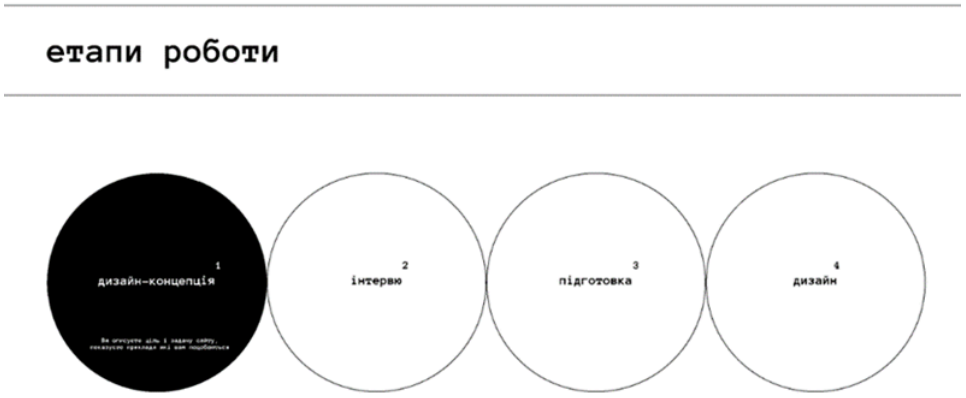


Рисунок 3.7 – Етапи роботи

В цьому блоці в нас є 4 кружечки ці кружечки будуть інтерактивні, тобто коли курсор не буде на кружечку то він буде мати вигляд як 2,3 і 4 кружечки, чорний текст на білому полотні, після наведення на кружечок він буде змінювати заливку з білої на чорну, текст також буде змінювати з чорного на білий для контрасту і також появиться додатковий текст.

І останній блок це footer в якому будуть розміщені контактні дані автора, такі як номер телефону, пошта та посилання на соціальні мережі.

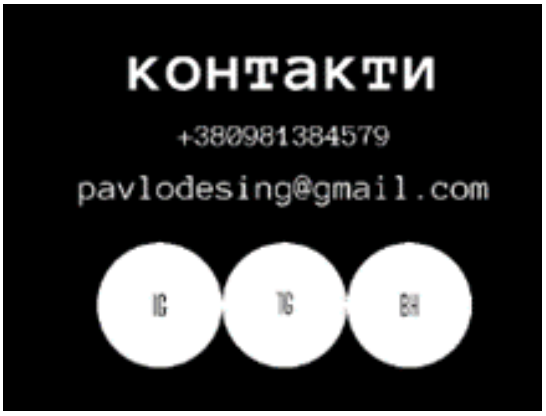


Рисунок 3.8 – Footer

Як ми бачимо на рисунку 3.8, що задній фон в нас темного кольору, текст білого і контактні дані розміщені по центру. Кнопки соціальних мереж я також планую зробити інтерактивними як в блоці “Етапи роботи”.

3.2 Структура проєкту

Проєкт складається з 3 головних папок.

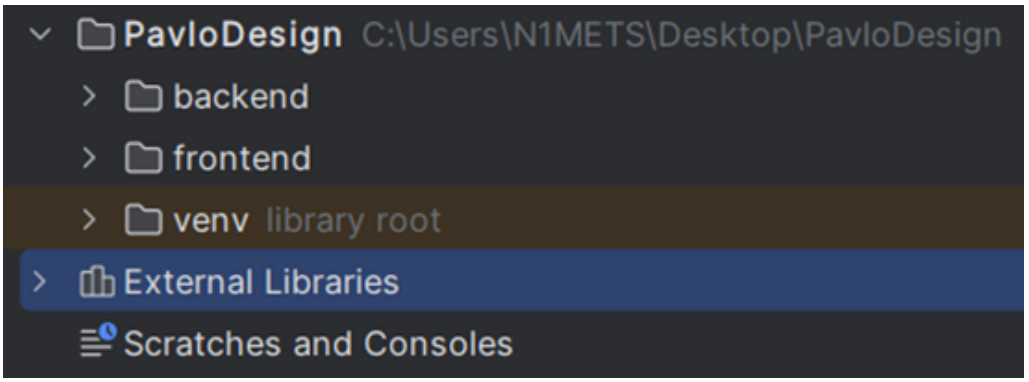


Рисунок 3.9 – Структура проєкту

На рисунку 3.9 продемонстровані ці 3 папки. Перша папка під назвою "backend" містить в собі наступні файли та каталоги:

1) application: основна директорія, в якій знаходиться код самого Django додатку;
2) database: ця директорія містить файли, пов'язані з базою даних;
3) project_images: директорія, яка зберігає зображення та інші мультимедійні файли, що використовуються в проєкті;
4) settings.py: цей файл містить налаштування Django проєкту, такі як параметри підключення до бази даних, встановлені додатки, конфігурація безпеки тощо;
5) initialization.py: файл ініціалізації проєкту;
6) manage.py: спеціальний скрипт Django який використовується для адміністрування проєкту, запуску веб-сервера тощо;
7) requirements.txt: цей файл містить список пакетів Python та їх версій, які необхідні для роботи проєкту.

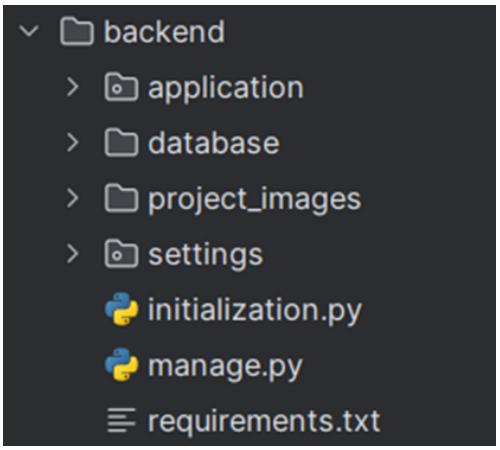


Рисунок 3.10 – Структура паки "backend"

Давайте тепер перейдемо до структури папки "frontend". На рисунку 3.11, ви можете побачити структуру директорій та файлів для частини веб-сайту яку буде бачити користувач.

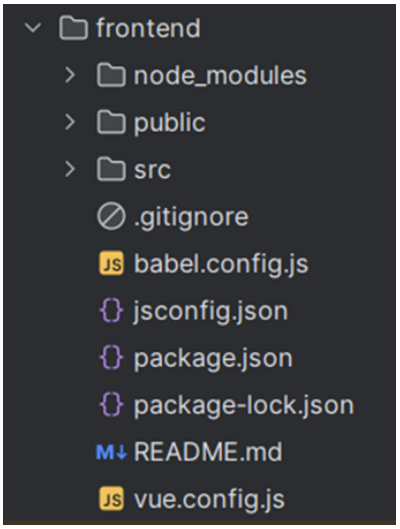


Рисунок 3.11 – Структура паки "frontend"

1) node_modules - ця директорія зберігає всі встановлені Node.js пакети;

2) public - тут зберігається заготовка для веб-сторінки та фавікон;

3) src - в цій директорії знаходиться вихідний код Vue.js;

4) .gitignore - цей файл містить список каталогів та файлів які мають бути проігноровані системою контролю версій Git;

5) babel.config.js - конфігураційний файл для Babel. Він перетворює новітній код JavaScript у версію сумісну з більшістю браузерів;

6) jsconfig.json - файл конфігурації для середовища розробки JavaScript в Visual Studio Code;

7) package.json - цей файл містить метадані проекту;

8) package-lock.json - створюваний менеджером пакетів npm для точного відтворення залежностей проєкту;

9) README.md - файл з інструкціями та іншою корисною інформацією про проєкт;

10) vue.config.js - конфігураційний файл для налаштування збірки Vue.js за допомогою webpack.

І остання директорія це є "venv" або "virtual environments" вона є вірутальним середовищем Python. Віртуальні середовища дозволяють ізолювати залежності та пакети Python для конкретного проекту, щоб уникнути конфліктів пакетів між різними проектами на одній машині.

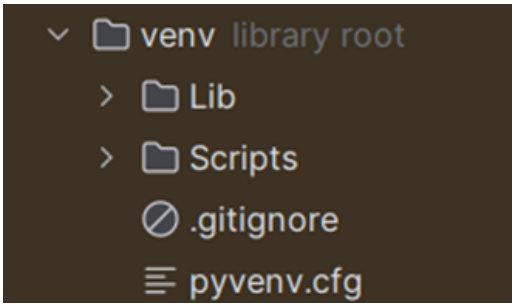


Рисунок 3.12 – Структура паки "venv"

На рисунку 3.12 ви бачите директорію "venv".

1) Lib: Директорія всередині venv, що містить встановлені Python-пакети та їх бібліотеки для даного віртуального середовища;

2) Scripts: Директорія всередині venv, що містить допоміжні скрипти та виконувані файли, пов'язані з управлінням віртуальним середовищем та встановленими пакетами;

3) .gitignore - Цей файл містить список каталогів та файлів які мають бути проігноровані системою контролю версій Git;

4) pyvenv.cfg: Конфігураційний файл, який зберігає налаштування віртуального середовища Python.

3.3 Головна сторінка сайту та навігація

Ця кваліфікаційна робота присвячена створенню сайту-візитки для веб-дизайнера. Мета полягала в розробці простого, але функціонального та естетично привабливого веб-ресурсу, який би демонстрував портфоліо робіт, надавав інформацію про послуги та забезпечував зручні способи зв'язку з веб-дизайнером. Для досягнення поставленої мети я обрав фреймворк Vue.js, оскільки він надає потужні інструменти для створення інтерактивних та швидких веб-інтерфейсів. Код який буде представлений нижче, є основним шаблоном головної сторінки сайту, створеним за допомогою Vue.js.

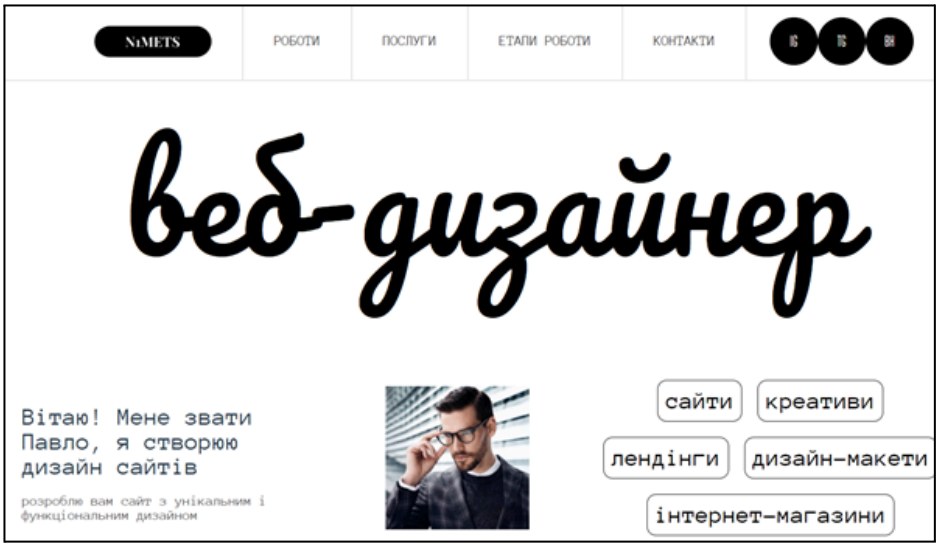


Рисунок 3.13 – Головна сторінка сайту

Зайшовши на сайт перше, що бачить відвідувач це великий надпис "веб-дизайнер" який вже сам за себе говорить чий це сайт, також бачить фото дизайнера, а також хедер.

Почнемо з того як створити такий хедер. На рисунку 3.14 ми бачимо фрагмент коду HTML шаблону компонента Vue.js, який відповідає за створення верхньої панелі навігації сайту.

```
1   <template>
2     <div>
3       <header>
4         <nav>
5           <div class="left-side">
6             <ul class="right-lines">
7               <li><a href="/"><img src="./assets/logo.png" alt="Логотип"></a></li>
8               <li><a @click="scrollToSection('works')">Роботи</a></li>
9               <li><a @click="scrollToSection('services')">Послуги</a></li>
10              <li><a @click="scrollToSection('work-stages')">Етапи роботи</a></li>
11              <li><a @click="scrollToSection('contacts')">Контакти</a></li>
12            </ul>
```

Рисунок 3.14 – Код хедера сайту

У рядках 1-2 ми бачимо відкриваючі теги "template" та "div", які позначають початок шаблону. У рядках 3-5 створюється секція "header" з навігаційним меню "nav", яке міститься всередині елемента "div" з класом "left-side". У рядках 6-12 знаходиться невпорядкований список "ul" з класом "right-lines", який містить елементи навігаційного меню. В рядку 7 створюється елемент списку "li" з посиланням на головну сторінку всередині якого міститься зображення логотипу. Рядки 8-11 створюють елементи списку "li" з посиланнями які при натисканні викликають метод "scrollToSection" з відповідним аргументом ("works", "services","work-stages","contacts").

Ці посилання прокручують сторінку до відповідних розділів сайту. Цей фрагмент коду створює верхню панель навігації з логотипом та посиланнями на різні розділи сайту. При натисканні на посилання викликається метод "scrollToSection", який плавно прокручує сторінку до потрібного розділу. Це забезпечує зручну навігацію та безперервний перегляд сторінки. Код також використовує директиву "@click", що дозволяє викликати методи та обробляти події натискання безпосередньо в шаблоні компонента.

В наступних рядках коду можна буде побачити як добавляються посилання на соціальні мережі.

```
13            <ul class="links-right">
14              <li><a href="https://www.instagram.com/n1metspavlodesing/" target="_blank">IG</a></li>
15              <li><a href="https://t.me/n1metspavlo" target="_blank">TG</a></li>
16              <li><a href="https://www.behance.net/n1metspavlo" target="_blank">BH</a></li>
17            </ul>
18          </div>
19        </nav>
20      </header>
21      <div id="prof" class="prof">
22        <profComp />
23      </div>
```

Рисунок 3.15 – Посилання на соціальні мережі

На рисунку 3.15 ми бачимо, що рядках 13-17 невпорядкований список "ul" з класом "links-right", який містить посилання на соціальні мережі. Рядок 14 сворює елемент списку "li" з посиланням на сторінку в Instagram. Атрибут "target" зі значенням "_blank" що означає, що посилання має відкриватися в новій вкладці. В рядку 15 та 16 робиться така сама процедура як в попередньому рядку тільки там посилання вже не на Instagram, а на Telegram та Behance. В рядку 18 закривається елемент "div" з класом "left-side", який містить навігаційне меню. В рядку 19 закривається тег "nav", який позначає кінець навігаційного меню. А в рядку 20 закривається тег "header", який позначає кінець секції header. У рядках 21-23 ми бачимо створення елемента "div" з id "prof" та класом "prof". Всередині цього елемента "div" знаходиться компонент який відповідає за відображення інформації про веб-дизайнера.

```
27        <div id="services" class="section">
28          <serviceComp />
29        </div>
30        <div id="work-stages" class="work-stages">
31          <stageComp />
32        </div>
33        <div id="contacts" class="footer">
34          <footerComp />
35        </div>
36      </div>
37    </template>
```

Рисунок 3.16 – Створення контейнерів

Ми вже створили верхню панель навігації, посилання на соціальні мережі, ще залишилося створити контейнери для розділів які будуть на сайті.

В рядках 27-29, що зображені на рисунку 3.16 ми бачимо створення елемента "div" з id "services" та класом "section". Всередині цього елемента "div" знаходиться компонент "serviceComp", який, відповідає за відображення розділу "Послуги" на сайті. У рядках 30-32 створюється елемента "div" з id "work-stages" з класом "work-stages". Всередині цього елемента "div" міститься компонент "stageComp", який, відображає інформацію про етапи роботи веб-дизайнера. У рядках 33-35 ми бачимо створення елемента "div" з id "contacts" та класом "footer". Всередині цього елемента "div" знаходиться компонент "footerComp", який відповідає за відображення інформації у футері сайту, такої як контактні дані та інша додаткова інформація. У рядку 36 закривається кореневий елемент "div", який обгортав усю структуру сайту. І нарешті, у рядку 37 закривається тег "/template", який позначає кінець шаблону компонента Vue.js.

Також нам потрібно ще прописати метод який буде плавно переміщати нас по сайту. Щоб користувач мав змогу натиснути на потрібну йому секцію і перемістится на неї без зайвих скролів колесом миші. Для початку нам потрібно імпортувати потрібні компоненти в код щоб використовувати у шаблоні Vue.js. Після чого потрібно створити об'єкт Vue.js додатка та зареєструвати ці компоненти.

```
46  export default {
47    name: 'App',
48    components: {
49      workComp,
50      serviceComp,
51      stageComp,
52      profComp,
53      footerComp
54    },
```

Рисунок 3.17 – Імпорт компонентів

На рисунку 3.17, ви можете побачити імпорт компонентів та реєстрацію додатку. В рядку 47 встановлюється назва додатка на рівні Vue.js, в рядку 48 реєструються компоненти Vue.js для використання у шаблоні додатка. Кожен компонент із списку компонентів буде доступний для використання у шаблоні під відповідною назвою. Дальше потрібно прописати метод який буде переміщати користувача по сайту.

```
55      methods: {
56        scrollToSection(sectionId) {
57          const element = document.getElementById(sectionId);
58          if (element) {
59            const offset = 110;
60            const offsetPosition = element.offsetTop - offset;
61            window.scrollTo({
62              top: offsetPosition,
63              behavior: 'smooth'
64            });
65          }
66        }
67      }
68    }
69    </script>
```

Рисунок 3.18 – Метод переміщення

На рисунку 3.18, ви можете побачити метод "scrollToSection(sectionId)" він використовується для прокрутки до певної секції сторінки за її ідентифікатором. Спочатку він отримує елемент секції за її ідентифікатором потім встановлює зміщення від верхнього краю сторінки для цієї секції з урахуванням певного зміщення, в даному випадку 110 пікселів. І в кінці викликає метод "window.scrollTo()" для прокрутки до певної позиції на сторінці з плавною анімацією.

Тепер можемо приступити до стилізації нашого хедера, а також для для кореневого компоненту додатку з ідентифікатором "app".

```
71    <style>
72    body {
73      margin: 100px 0 0 0;
74    }
75    #app {
76      font-family: Avenir, Helvetica, Arial, sans-serif;
77      -webkit-font-smoothing: antialiased;
78      -moz-osx-font-smoothing: grayscale;
79      text-align: center;
80      color: #2c3e50;
81      margin-top: 60px;
82    }
```

Рисунок 3.19 – Стилізація "app"

На рисунку 3.19, ви можете бачити, що в рядку 72 ми звертаємося до "body" і встановлюємо відступ "margin: 100px 0 0 0;" це означає, що верхній відступ буде 100 пікселів, а решта відступів будуть нульовими. Дальше ми звертаємося до селектора "#app" і задаємо йому шрифти. Перший шрифт який ми задали це є бажаний шрифт який має бути по стандарту, якщо цей шрифт недоступний, буде використано наступний доступний шрифт з переліку. В рядках 77 та 78 ми прописуємо згладження шрифтів. Дальше в най йде вирівнювання по центру, задавання кольору тексту та верхній відступ.

Дальше в нас йде написання головної сторінки нашого сайту, де в нас буде інформація про автора і його послуги.

```
1    <template>
2      <div class="web-designer">
3        <h1>веб-дизайнер</h1>
4        <div class="content">
5          <div class="left-side">
6            <p class="title">Вітаю! Мене звати<br>Павло, я створюю дизайн сайтів</p>
7            <p class="text">розроблю вам сайт з унікальним і функціональним дизайном</p>
8          </div>
```

Рисунок 3.20 – Блок з інформацією про веб-дизайнера

На рисунку 3.20 ми бачимо, що в рядку 1 відкривається контейнер або шаблон під назвою "template" в якому буде зберігатися код для зберігання HTML коду для подальшого використання в JavaScript. В рядку 2 створюється

"div" елемент з класом "web-designer" для стилізації всіх елементів з цим класом. В рядках 3 створюється створює заголовок першого рівня з текстом "веб-дизайнер". Це заголовок сторінки. В наступному рядку ще раз створюється "div" елемент з класом "content" але він вже буде стилізувати основний вміст сторінки сайту. І в наступному коді також створюється "div" елемент з класом "left-side" який буде стилізувати вміст розташований зліва. В рядках 6 та 7 використовується тег "p" з різними класами, в рядку 6 використовується клас "title". Він містить привітання та інформацію про веб-дизайнера, а в рядку 7 використовується "text". Він містить текст, що описує послуги веб-дизайнера. Ще в рядку 6 є тег "br" він використовується для вставлення переносу рядка. Коли браузер стикається з тегом "br", він переходить на новий рядок, тобто текст, що йде після тегу "br", розміщується на новому рядку. І в рядку 8 в нас йде закриття тегу "div".

Після текстової інформації про автора в нас ще має бути його фото та список послуг. На рисунку 3.21 ми можемо побачити код який додає фото та список послуг.

```
9         <div class="photo">
10          <img src="../assets/prof.png" alt="Веб-дизайнер Павло">
11        </div>
12        <div class="right-side">
13          <div>
14            <div>сайти</div>
15            <div>креативи</div>
16            <div>лендінги</div>
17            <div>дизайн-макети</div>
18            <div>інтернет-магазини</div>
19          </div>
20        </div>
21      </div>
22    </div>
23  </template>
```

Рисунок 3.21 – Додавання фото і послуг

В рядку 9 ми бачимо тег "div" який має клас "photo" в якому міститься тег "img" який відображає фотографію веб-дизайнера з файлу "prof.png",

розташованого у папці "assets". Атрибут "alt" вказує альтернативний текст для зображення у випадку, якщо воно не може бути відображене. В рядку 12 ми маємо тег "div" який має клас "right-side" в якому міститься блок з п'ятьма внутрішніми "div" елементами, кожен з яких містить текст, що представляє певну категорію, наприклад, "сайти", "креативи" тощо. І в кінці в нас закриваються всі відкриті теги "div" та шаблон "template".

Після написання головного блоку дальше йде оголошення JavaScript скрита. Це ви можете побачити на рисунку 3.22

```
25  <script>
26  export default {
27    name: 'WebDesigner',
28  }
29  </script>
```

Рисунок 3.22 – Скрипт

У цьому JavaScript скрипті використовується Vue.js для оголошення компонента з іменем "WebDesigner". Vue компоненти дозволяють організувати веб-сторінку в окремі частини, які можна повторно використовувати, а також керувати їх поведінкою.

Нам залишилося стилізувати весь код, що ми писали вище щоб він мав привабливий вигляд.

```
31  <style scoped>
32  .web-designer h1 {
33    font-family: 'Pacifico';
34    font-style: normal;
35    font-weight: 400;
36    font-size: 175px;
37    line-height: 307px;
38    color: black;
39    margin: 50px 0 70px;
40  }
```

Рисунок 3.23 – Стилізація надпису "веб-дизайнер"

На рисунку 3.23 ми бачимо тег "style scoped" - це спеціальний тег у Vue.js, який використовується для визначення стилів, які будуть застосовані тільки до компонентів, до яких вони відносяться. Це означає, що стилі, визначені всередині тега "style scoped", будуть діяти тільки на елементи внутрішнього шаблону цього конкретного компонента Vue, і вони не будуть впливати на стилі зовнішніх елементів або інших компонентів. В рядку 32 ми стилізуємо всі елементи тегу "h1" які знаходяться знаходяться всередині елементів з класом ".web-designer". Після чого встановлюється шрифт "Pacifico", нормальний стиль тексту, товщина шрифту 400, розмір шрифту 175 пікселів, висоту рядка 307 пікселів, чорний колір тексту та відступи від верхнього та нижнього краю 50 і 70 пікселів. Дальше ми в коді звертаємося до елементів з класом "content", прописуємо їм гнучкість щоб можна було легко керувати розташуванням елементів всередині них, також розміщуємо елементи всередині гнучкого контейнера рівномірно з рівними проміжками між ними, вирівнюємо їх по середині, встановлюємо ширину 1200 пікселів та встановлюємо зовнішні відступи по боках. Ці всі команди ви можете побачити на рисунку 3.24

```
41  .content {
42    display: flex;
43    justify-content: space-between;
44    align-items: center;
45    width: 1200px;
46    margin: 0 auto;
47  }
```

Рисунок 3.24 – Стилізація класу "content"

Тепер ми будемо стилізувати ліву частину сайту, а саме елементи з класами ".left-side" і також його дочірній елемент з класом ".title"

```
.left-side {
  width: 395px;
  display: flex;
  flex-direction: column;
  text-align: start;
}
.left-side .title {
  font-family: 'Anonymous Pro';
  font-style: normal;
  font-weight: 400;
  font-size: 32px;
  line-height: 32px;
  margin-bottom: 10px;
}
```

Рисунок 3.25 – Стилізація класу "left-side" та "left-side .title"

Для елементів з класом "left-side" встановлена ширина 395 пікселів, встановлена гнучкість, що дозволяє легко керувати розташуванням дочірніх елементів всередині них також прописано напрямок гнучкого контейнера по вертикалі, тобто дочірні елементи розташовуються один за одним по вертикалі і вирівнювання тексту всередині елементів. А для елементів з класом "left-side .title" задається шрифт "Anonymous Pro", нормальний стиль тексту, товщина шрифту 400, розмір шрифту 32 пікселі, висоту рядка 32 пікселі і задається зовнішній відступ знизу елементів на 10 пікселів. В нас ще залишився один елемент з класом "left-side" в середині якого є клас "text", стилізацію цього класу ви можете побачити на рисунку 3.26

```
.left-side .text {
  font-family: 'Anonymous Pro';
  font-style: normal;
  font-weight: 400;
  font-size: 18px;
  line-height: 18px;
  color: #5B5B5B;
  margin-top: 10px;
}
```

Рисунок 3.26 – Стилізація класу "left-side .text"

Тут так само як в попередньому класі "left-side .title" задається шрифт "Anonymous Pro", нормальний стиль тексту, товщина шрифту 400, але розмір шрифту вже 18 пікселів, висота рядка також 18 пікселів, колір тексту сірий і зовнішній відступ зверху для елементів 10 пікселів.

Стилізуємо також фото яке ми добавили. Його ми розмістимо по центру між лівим та правим блоками тексту.

```
71  .photo img {
72    max-width: 100%;
73    display: block;
74    margin: 0 auto;
75  }
```

Рисунок 3.27 – Стилізація зображення автора

Щоб помістити наше фото по центру спершу потрібно до нього звернутися через клас "photo img", дальше ми встановлюємо максимальну ширину зображення на 100%. Це потрібно для того щоб він не перевищував ширину його батьківського елемента. Після чого встановлює зображення як блочний елемент. Це означає, що інші елементи будуть відображатися над та під зображенням, а не поруч з ним і на завершення встановлюємо автоматичні зовнішні відступи зверху і знизу їх ми позначили "0" та автоматичний відступ зліва і справа, а їх ми позначили "auto". Це центрує зображення горизонтально всередині свого батьківського контейнера.

Ліву та центральну частину сайту ми вже стилізували, залишилося праву.

```
76  .right-side {
77    width: 475px;
78    display: flex;
79    flex-direction: column;
80  }
81  .right-side > div {
82    display: flex;
83    flex-wrap: wrap;
84    justify-content: center;
85
86  }
```

Рисунок 3.28 – Стилізація класу "right-side" та "right-side > div"

На риску 3.28 ми бачимо наступне, що в рядку 76 йде звернення до класу ".right-side", в рядку 77 встановлюється ширина елементів на 475 пікселів, також в рядку 78 вказується, що вони мають відображатися як гнучкий контейнер, щоб ними було легко керувати і в останньому 79 рядку прописується напрямок розміщення дочірніх елементів всередині гнучкого контейнера по вертикалі. З цим параметром, елементи будуть розташовані вертикально один після одного, починаючи зверху і закінчуючи знизу. В класі "right-side > div" також прописаний "display: flex" для ефективного заповнення простору, властивість "flex-wrap" дає можливість виводити "flex" елементи в один або кілька рядів з перенесенням блоків і останній атрибут "justify-content: center" вирівнює елементи в контейнері коли вони не використовують ввесь простір, що їм надається. І останнє, що ми зробимо ми стилізуємо в цій правій частині сайту, це текст і рамки які ми бачили на рисунку 3.4

```
87  .right-side div div {
88    flex-grow: 0;
89    padding: 10px;
90    margin: 10px;
91    text-align: center;
92    font-family: 'Anonymous Pro';
93    font-style: normal;
94    font-weight: 400;
95    font-size: 32px;
96    line-height: 32px;
97    color: #000000;
98    border: 1px solid #000000;
99    border-radius: 12px;
100 }
101 </style>
```

Рисунок 3.29 – Стилізація рамок

На рисунку 3.29 ми бачимо, що в рядку 88 вимикається можливість гнучким елементам розширюватися відносно інших гнучких елементів у контейнері. Тобто, елементи не будуть змінювати свою ширину, щоб зайняти весь доступний простір. В рядку 89 та 90 встановлюється внутрішній та зовнішній відступи по 10 пікселів. "text-align: center" вирівнює текст по

середині, встановлюється шрифт "Anonymous Pro", прописується нормальний стиль тексту, товщина шрифту 400, розмір шрифту та висота рядка 32 пікселі, колір тексту чорний і в рядку 98 та 99 прописується рамка товщиною 1 піксель чорного кольору із заокругленням 12 пікселів.

3.3.1 Портфоліо

Після того як ми написали привітальну частину, дальше ми приступаємо до написання блоку для портфоліо. На рисунку 3.20, ви можете побачити як створюється блок з портфоліо. Як і в попередньому коді 1 рядок коду в нас починається з "template", в рядках 2-3 створюються "div" контейнери з класами "work-item" та "work-title" для того щоб можна було в майбутньому їх стилізувати. В рядку 4 створюється заголовок для цього контейнера.

```
1  <template>
2    <div class="work-item">
3      <div class="work-title">
4        <h2>роботи</h2>
5      </div>
6      <div class="work-content">
7        <div v-for="(work, index) in works" :key="index" class="work-entry">
8          <div class="work-width">
9            <img :src="getPhoto(work)" :alt="work.project_name">
10           <div class="right-side">
11               <div class="work-details">
12                   <h3>{{ work.project_name }}</h3>
13                   <p>{{ work.project_for_whom }}</p>
14               </div>
15               <a :href="work.project_link" class="view-button" target="_blank">ДИВИТИСЯ</a>
16           </div>
17         </div>
18       </div>
```

Рисунок 3.30 – Створення блоку портфоліо

В рядку 6 створюється контейнер "div" з класом "work-content" для відображення списку робіт. В рядку 7 використовується директива "v-for" для циклічного проходження по масиву "works" і створення конейнера "div" з класом "work-entry" для кожної роботи. В рядку 8 створюється контейнер "div" з класом "work-width" для управління шириною елемента роботи. В 9 рядку

відображається зображення для кожної роботи. В рядку 10 створюється контейнер "div" з класом "right-side" для розміщення деталей роботи праворуч від зображення. В рядку 11 створюється контейнер "div" з класом "work-details" для відображення деталей роботи. В рядку 12 відображається назва проекту як елемент "h3". В рядку 13 відображається інформація про того, для кого виконувалася робота, як елемент параграфа. В рядку 15 створюється посилання з класом "view-button" для перегляду деталей роботи. ":href="work.project_link"" зв'язує "href" посилання з властивістю "project_link" з масиву і відкриває посилання в новій вкладці. Ми закінчили написання 3 контейнерів які демонструють роботи, а в нас ще є контейнер для зв’язку з автором. На рисунку 3.31 ми бачимо аналогічне створення "div" контейнерів для того щоб можна було в майбутньому їх стилізувати. В рядку 21 ми прописуємо тег для відображення зображення, завдаємо шлях та альтернативний текст.

```
19          <div class="work-entry">
20            <div class="work-widths">
21              <img src="../assets/placement.png" alt="work.project_name">
22              <div class="right-side connect">
23                  <div class="work-details">
24                      <h3>ЗВЯЖІТЬСЯ ЗІ МНОЮ</h3>
25                  </div>
26                  <a href="https://linktr.ee/n1metsdesing" class="view-button" targ
27              </div>
28            </div>
29          </div>
30        </div>
31      </div>
32    </template>
```

Рисунок 3.31 – Створення блоку зв’язку

В рядку 22 ми створюємо контейнер "div" з класом "right-side connect" який створює внутрішній блок HTML, який буде розташований праворуч від зображення і який буде містити інформацію для зв'язку. В рядку 23 ми ще раз створюємо контейнер "div" з класом "work-details" який буде використовуватися для розміщення тексту. В рядку 24 ми використовує тег "h3", а в рядку 25 ми

закриваємо контейнер "div". В рядку 26 створюється текст з посиланням на спеціальний сервіс який надає контакти зв’язку з автором.

Після написання візуальної частими ми оголошуємо скрипт та імпортуємо бібліотеку Axios для здійснення запитів HTTP. Це ви можете побачити на рисунку 3.32

```
34  <script>
35  import axios from 'axios';
36
37  export default {
38    data() {
39      return {
40        works: {},
41      };
42    },
```

Рисунок 3.32 – Імпорт бібліотеки "Axios"

В рядку 37 об'єкт експортується як основний об'єкт модуля. В рядку 38 метод "data()" визначає початкові дані компонента. У даному випадку, створюється об'єкт "works", який починається як порожній об'єкт. Наступний метод метод викликається після того, як компонент було відображено на сторінці, метод представлено на рисунку 3.33

```
43  mounted() {
44    axios.get('http://127.0.0.1:8000/main')
45      .then(response => {
46        this.works = response.data[0];
47        console.log(this.works)
48      })
49      .catch(error => {
50        console.error('Error fetching data:', error);
51      });
52  },
```

Рисунок 3.33 – Отримання даних з сервера

У цьому методі відбувається отримання даних з сервера за допомогою HTTP запиту до URL. Отримані дані присвоюються змінній "works". Якщо запит відбувся успішно, дані зберігаються у змінній, в іншому випадку виводиться помилка у консоль. І останній функція в цьому коді буде методом який буде отримувати зображення. На рисунку 3.34, ви можете побачити метод "getPhoto", який отримує фотографію роботи. Цей метод приймає один аргумент "works", який представляє об'єкт роботи. Метод спробує розібрати URL фотографії з об'єкта роботи та поверне шлях до фотографії. Якщо спроба розібрати URL не вдасться, метод поверне порожній рядок та виведе помилку.

```
53    methods: {
54      getPhoto(works) {
55        try {
56          const photoPath = works.project_image.split('/project_images/')[1];
57          const fileName = photoPath.split('/').pop();
58          return require(`@/../../backend/project_images/${fileName}`);
59        } catch (error) {
60          console.error('Error loading photo:', error);
61          return '';
62        }
63      },
64    },
```

Рисунок 3.34 – Метод отримання зображень

Після написання візуальної частини сайту та імпорту скриптів з бібліотекою можна приступити до стилізації коду. На рисунку 3.35, ви можете побачити фрагмент коду який відповідає за стилізацію елементів з класами: "connect", "work-item", "work-title", та "work-title h2".

```
68    <style>
69    .connect {
70      margin-left: 120px;
71    }
72    .work-item {
73      margin-top: 50px;
74      border-top: 1px solid black;
75    }
```

Рисунок 3.35 – Стилізація заголовку

В класі "connect" прописаний стиль який встановлює лівий відступ для елементів з класом "connect", роблячи їх відступ від лівого краю контейнера на

120px. В класі "work-item" встановлюється верхній відступ для елементів на 50px та встановлює верхню горизонтальну межу, яка представлена чорною лінією товщиною 1 піксель.

```
76  .work-title {
77    border-bottom: 1px solid black;
78    padding: 10px 0;
79  }
80  .work-title h2 {
81    width: 1200px;
82    margin: 20px auto;
83    text-align: justify;
84    font-family: 'Anonymous Pro';
85    font-style: normal;
86    font-weight: 700;
87    font-size: 48px;
88    line-height: 48px;
```

Рисунок 3.36 – Стилізація заголовку 2

В класі "work-title" стилі встановлюють нижню горизонтальну межу для елементів, що представлена чорною лінією товщиною 1 піксель. Також встановлюється внутрішній відступ 10 пікселів по вертикалі та 0 по горизонталі для зручності розміщення вмісту в межах цього елемента. В нас також є ще одне згадування класу "work-title" тільки вже із заголовком "h2". В стилізації цього заголовку використовується стиль "width: 1200px" який встановлює ширину заголовків на 1200 пікселів, щоб контент займав фіксовану ширину на сторінці. Також задається внутрішній відступ по вертикалі 20px і автоматичне вирівнювання по горизонталі, щоб заголовок був центрованим відносно батьківського контейнера. Це ви можете побачити в рядку 82. В рядку 83 стиль вирівнює текст заголовка по ширині, рівномірно розподіляючи пробіли між словами і вирівнюючи текст відносно обох країв блоку. В рядку 84 встановлює шрифт заголовка на шрифт з назвою "Anonymous Pro". Також в наступному рядку встановлює нормальний стиль шрифту, без нахилу або курсиву. Дальше встановлюється жирний стиль шрифту. І в рядках 87 та 88 встановлюється розмір шрифту який становить 48 пікселів та стиль який відповідає за висоту рядка який також має 48 пікселів, забезпечуючи відповідний проміжок між рядками тексту.

Це була стилізація заголовку для нашого контейнера, дальше будуть рядки коду які стилізують сам контейнер, все що в ньому буде знаходититися, а саме: зображення, текст, кнопки. На рисунку ви можете побачити як стилізується сам контейнер для робіт.

```
90  .work-width {
91      width: 1200px;
92      display: flex;
93      justify-content: space-between;
94      margin: 0 auto;
95  }
96  .work-widths {
97    width: 1200px;
98    display: flex;
99    margin: 0 auto;
100
101 }
102 .work-details {
103     display: flex;
104     flex-direction: row;
105     justify-content: center;
106     gap: 120px
107 }
```

Рисунок 3.37 – Стилізація контейнера

В рядку 90 ми звертаємося до класу "work-width" який має ширину контейнера 1200 пікселів, також цей контейнер має властивість розміщувати елементи в рядок або стовпчик, атрибут "justify-content: space-between" розподіляє простір навколо флекс-елементів вздовж головної осі контейнера. Це робиться після того, як застосовуються розміри та автоматичні відступи і останній атрибут "margin: 0 auto;" це означає, що елемент буде вирівняно по горизонталі всередині свого батьківського контейнера.

В рядку 96 ми звертаємося до класу "work-widths" але код в ньому майже аналогічний попередньому тільки є одна відмінність, а саме відсутність атрибуту "justify-content: space-between". Це пов'язано з тим, що для цього блоку не потрібно рівномірно розподіляти дочірні елементи по горизонталі з проміжками між ними. Натомість, вони вирівнюються зліва на направо за

замовчуванням. В класі "work-details" є код який перетворює його на контейнер "flexbox", розміщує флекс-елементи в рядку та один поруч з одним, вирівнює флекс-елементи по центру контейнера та додає проміжок між флекс-елементами в 120 пікселів.

Після стилізації контейнерів можна перейти до стилізації заголовків дочірніх елементів класів які ми вже стилізували. На рисунку 3.37, ви можете побачити як вони стилізуються.

```
108  .work-details h3 {
109      width: 372px;
110      text-align: justify;
111      margin: 0;
112  }
113  .work-details h3 {
114    font-family: 'Anonymous Pro';
115    font-style: normal;
116    font-weight: 400;
117    font-size: 48px;
118    line-height: 48px;
119  }
```

Рисунок 3.38 – Стилізація заголовків класу "work-details"

В першому звернені до заголовку "h3" класу "work-details" встановлюється ширина заголовків "h3" на 372 пікселі. Робиться вирівнювання тексту по ширині, також видаляються будь-які поля, які можуть бути по замовчуванню застосовані до заголовків "h3". За це відповідає властивість "margin: 0;" вона встановлює поле на 0 у всіх напрямках. В другому звернені встановлює шрифт "Anonymous Pro", прописується звичайний стиль шрифту, встановлюється жирність шрифту на нормальну, а також встановлюється розмір шрифту який становить 48 пікселів та стиль який відповідає за висоту рядка який також має 48 пікселів, забезпечуючи відповідний проміжок між рядками тексту.

Клас "work-details" крім тегу "h3" які відповідають за заголовки, має також ще тег "p" який відповідає за абзац який містить ім’я замовника або організації, для якої був виконаний проєкт.

```
120  .work-details p {
121    font-family: 'Anonymous Pro';
122    font-style: normal;
123    font-weight: 400;
124    font-size: 20px;
125    line-height: 20px;
126    text-align: justify;
127    margin: 0;
128    display: flex;
129    align-items: center;
130    text-transform: uppercase;
131  }
```

Рисунок 3.39 – Стилізація параграфу класу "work-details"

Стилізація параграфу класу "work-details" починається з присвоєння йому шрифта "Anonymous Pro", дальше встановлюється нормальний шрифт стилю, встановлюється жирність 400 тобто нормальна, прописується розмір тексту 20 пікселів, встановлюється також висота 20 пікселів, прописується вирівнювання тексту по ширині, видаляються будь-які поля, які можуть бути по замовчуванню застосовані, встановлює режим відображення для абзаців на "flex", в рядку 129 прописано вирівнювання всіх елементів по центру і в кінці властивість "text-transform: uppercase;" переводить весь текст в верхній регістр.

Після стилізації тексту можна приступити і до оформлення вигляду записів робі. Задати їм кордони, відступи, стилізувати зображення, масштабувати. Почнемо з класу "work-entry", на рисунку 3.8, ви можете, що в класі "work-entry" прописаний чорний нижній кордон товщиною 1 піксель, це використовується для візуального відокремлення кожного запису роботи один від одного. Також прописаний внутрішній відступ зверху та знизу елементів з цим класом на 40 пікселів, але не додає відступу зліва та справа.

В класі "work-entry img" встановлюється ширина зображення 457 пікселів та висота зображення 260 пікселів.

В рядку 139 прописана властивість "object-fit: cover;" яка буде масштабувати зображення, щоб повністю заповнити контейнер, зберігаючи співвідношення сторін. Якщо зображення має інше співвідношення сторін, ніж контейнер, частини зображення вони бути обрізані.

```
132  .work-entry {
133      border-bottom: 1px solid black;
134      padding:  40px 0;
135  }
136  .work-entry img {
137    width: 457px;
138    height: 260px;
139    object-fit: cover;
140    margin-right: 40px;
141  }
142  .work-details h3 {
143    margin: 0;
```

Рисунок 3.40 – Стилізація запису робіт

І остання властивість в цьому класі це "margin-right: 40px;" вона додає зовнішній відступ праворуч від зображення на 40 пікселів. Це створює простір між зображенням та іншим вмістом елемента.

Для останнього контейнера в якому в нас немає демонстрації роботи, а тільки заклик до комунікації з автором використано окремий клас. На рисунку 3.29, ви можете побачити стилізацію цього класу.

```
145  .right-side {
146      display: flex;
147      flex-direction: column;
148      justify-content: space-between;
149  }
```

Рисунок 3.41 – Стилізація класу "right-side"

В цьому класі використано "display: flex;" для ефективного заповнення простору, також використано "flex-direction: column;" ця властивість визначає напрямок, вздовж якої будуть розташовані флекс-елементи всередині контейнера. "colum" вказує, що елементи будуть складені один над одним і

остання властивість це "justify-content: space-between; " вона рівномірно розподіляє флекс-елементи по всьому рядку, що перший і останній флекс прижимаються до країв контейнера.

І останнє що ще не було стилізовано в нашому контейнері це кнопка. На рисунку 3.41, ви можете побачити код яким вона стилізувалася.

```
150  .view-button {
151      margin-top: auto;
152      text-align: justify;
153      border-radius: 8px;
154      border: 1px solid black;
155      width: 177px;
156      display: flex;
157      justify-content: center;
158      font-family: 'Anonymous Pro';
159      font-style: normal;
160      font-weight: 400;
161      font-size: 32px;
162      line-height: 32px;
163      transition: 0.3s ease-in-out;
164  }
165  .view-button:hover {
166    background-color: black;
167    color: white;
168  }
```

Рисунок 3.42 – Стилізація кнопки

В класі "view-button" ми надаємо вигляд самій кнопці. Спочатку ми встановлюємо верхній відступ кнопки. У флекс-контейнері це відштовхує кнопку до нижньої частини доступного простору в її контейнері, дальше в нас йде вирівнювання тексту всередині кнопки по ширині, після ми вже прописуємо закруглені кути кнопці з радіусом 8 пікселів, створюючи більш плавний вигляд і також додаємо чорну суцільну рамку товщиною 1 піксель до кнопки. Дальше ми розміщуємо нашу кнопку, ми прописуємо ширину кнопки 177 пікселів, задаємо їй властивість "display: flex;". Оскільки кнопка тепер є флекс-контейнером, ця властивість вирівнює текст кнопки по горизонталі всередині кнопки. Тому в рядку 157 в нас є команда "justify-content: center; ". Також задаємо шрифт, його стиль шрифту, товщину, розмір, висоту та плавний перехід. "transition: 0.3s ease-in-out;" це якраз властивість яка робить плавний перехід коли користувач буде наводити на кнопку курсором миші. В класі

"view-button:hover" якраз прописано як має змінюватися кнопка при наведені на неї. По стандарту кнопка з тексом в нас має чорну ободку, чорний текст та білий фон, при наведені вона змінює фон та колір тексту. Це ви можете побачити на рисунку 3.41 в рядку 166 прописана команда "background-color: black;" ця властивість встановлює фоновий колір кнопки на чорний, коли користувач наводить на неї курсор миші завдяки класу "hover", а також в рядку 167 прописана властивість яка встановлює колір тексту кнопки на білий, коли користувач наводить на неї курсор миші.

Додавання та редагування інформації для портфоліо здійснюється через адмін панель Django. Це ви можете побачити на рисунку 3.42

Рисунок 3.43 – Адмін панель Django

Щоб наприклад змінити дані про якусь існуючу роботу потрібно просто натиснути на неї, вам відкриється вікно з редагуванням назви проекту, зміною зображення, з редагуванням тексту для кого цей проект був зроблений та посилання нам сам проект це ви можете побачити на рисунку 3.43

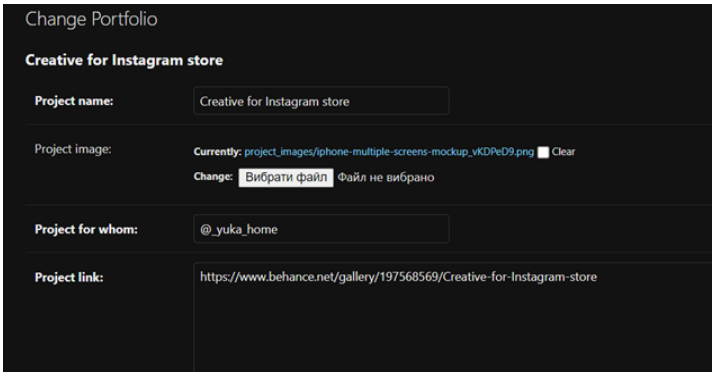


Рисунок 3.44 – Редагування портфоліо

3.3.2 Послуги

Розділ "Послуги" на сайті є дуже важливим, оскільки саме він дозволяє зрозуміти відвідувачам сайту, чи зможе автор задовольнити їхні потреби. На головній сторінці сайту з правого боку від фотографії були вже згадані послуги які надає автор, але там вони були написані без конкретного опису, та без додаткової інформації по них. В цьому розділі буде детальний опис усіх послуг які надає автор. На рисунку 3.44, ви можете побачити як виглядає розділ послуги.

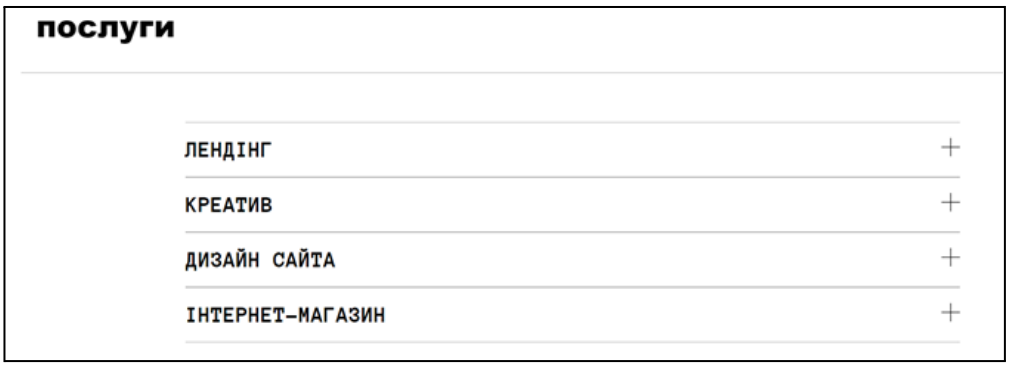


Рисунок 3.45 – Послуги

Щоб створити такий розділ спочатку потрібно створити для нього контейнер який буде вміщати в собі всі дані. Дальше ми створюємо заголовок розділу "Послуги". Це ви можете побачити на рисунку 3.45 в 3 та 4 рядку.

```
1  <template>
2    <div class="services">
3      <div class="work-title services-title">
4        <h2>послуги</h2>
5      </div>
6      <div class="service" v-for="(service, index) in services" :key="index">
7        <div class="service-header" @click="toggleService(index)">
8          <span class="service-name">{{ service.name }}</span>
9          <span class="service-toggle">{{ service.open ? '-' : '+' }}</span>
10       </div>
11       <div class="service-description" :class="{ 'open': service.open, 'closed': !service.open }">
12         {{ service.description }}
13       </div>
14     </div>
15     <div class="serv"></div>
16   </div>
17 </template>
```

Рисунок 3.46 – Створення розділу послуги

В 6 рядку ми створюємо контейнер для кожної послуги яка в нас буде, також використовується директива "v-for" для відтворення цього контейнера для кожного об'єкта "service" в масиві "services. :key" вказує на унікальний ідентифікатор кожного елемента. Під час виконання цього коду перебирається масив і для кожного елемента створює окремий контейнер. В рядку 7 в нас код який містить назву кожної послуги та знак "+" або "-". Ці знаки вказують, чи розгорнута вже ця послуга чи ні. При натисканні на заголовок викликається метод "toggleService(index)".

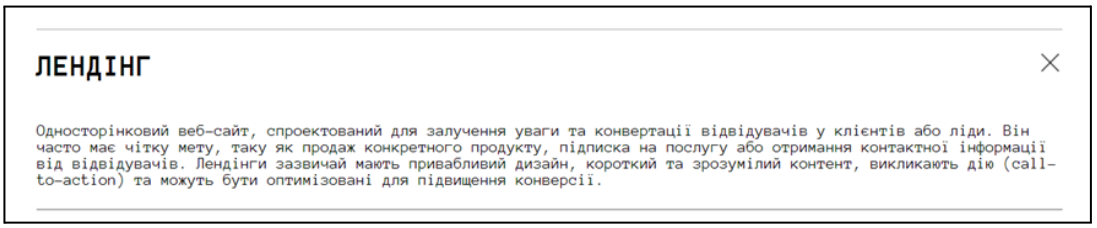


Рисунок 3.47 – Розгорнута послуга

В рядку 11 ми маємо блок опису послуги. Він має два класи "open" або "closed" в залежності від того, чи розгорнута послуга чи відкрита. Вміст цього блоку відображається або приховується в залежності від стану "service.open".

Перейдемо до написання скрипта який буде керувати нашим списком послуг. На рисунку 3.47, ви можете бачити код скрипта.

```
19  <script>
20  export default {
21    data() {
22      return {
23        services: [
24          { name: 'Лендінг', description: 'Односторінковий веб-са
25          { name: 'Креатив', description: 'Розробка унікальних та
26          { name: 'Дизайн сайта', description: 'Створення естетич
27          { name: 'Інтернет-магазин', description: 'Створення онл
28        ]
29      };
30    },
31    methods: {
32      toggleService(index) {
33        this.services[index].open = !this.services[index].open;
34      }
35    }
36  }
37  </script>
```

Рисунок 3.48 – Скрипт послуг

У функції "data()" ми повертаємо об'єкт який містить властивість "services". Це масив об'єктів де кожен об'єкт представляє окрему послугу. Кожен об'єкт має три властивості, а саме: "name" це назва послуги, "description" це опис послуги та "open" це прапорець який показує чи розгорнутий опис цієї послуги чи ні. По стандарту всі описи згорнуті.

Далі визначається метод "toggleService". Цей метод змінює значення властивості "open" для обраної послуги на протилежне. Якщо послуга була згорнута "open: false", то метод розгорне її "open: true" і навпаки.

Після написання скрипта можна приступити до стилізації нашого блоку послуг. Перше що потрібно зробити це зробити відступ, щоб забезпечити достатній простір між цим розділом та контентом яким ми в подальшому будемо стилізувати. На рисунку 3.48, ви можете побачити, що в рядку 41 робиться відступ на 65 пікселів.

```
39  <style>
40  .services {
41    margin-bottom: 65px;
42  }
43  .services-title {
44    margin-bottom: 65px;
45  }
46  .service {
47    max-width: 1000px;
48    margin: 0 auto;
49  }
50  .serv {
51    border-top: 1px solid #D4D4D4;
52    max-width: 1000px;
53    margin: 0 auto;
54  }
```

Рисунок 3.49 – Стилізація контейнера послуг

В рядку 43 встановлюється відступ внизу заголовка на 65 пікселів, щоб створити відстань між заголовком і першою послугою, в рядку 46 ми звертаємося до класу "service" та задаємо максимальну ширину контейнера 1000 пікселів і автоматичне вирівнювання по центру. В рядку 50 ми звертаємося

до класу "serv" та встановлюємо лінію розміром 1 піксель, щоб розділити послуги, також задаємо максимальну ширину контейнера 1000 пікселів і автоматичне вирівнювання по центру. Дальше нам потрібно стилізувати заголовок кожної послуги, тексту який розміщений всередині послуги та до знаку плюс і мінус всередині заголовка. На рисунку 3.49, ви можете побачити код який це все стилізує.

```
.service-header {
  display: flex;
  justify-content: space-between;
  cursor: pointer;
  border-top: 1px solid #D4D4D4;
  box-sizing: border-box;
}
.service-name {
  padding: 20px;
  font-family: 'Anonymous Pro';
  font-style: normal;
  font-weight: 400;
  font-size: 32px;
  line-height: 32px;
}
.service-toggle {
  margin-left: 10px;
  font-size: 28px;
  padding: 5px;
  display: flex;
  justify-content: center;
  align-items: center;
}
```

Рисунок 3.50 – Стилізація елементів всередині контейнера

Спочатку ми звертаємося до "service-header " встановлюється властивість "display: flex ", щоб розмістити елементи всередині заголовка. "justify-content: space-between " розмішає назву послуги та знак плюс та мінус від краю до краю контейнера, щоб створити проміжок між ними. Вказується курсор "pointer" для вказівки на можливість клікнути на заголовок. Додається тонка лінія зверху, "box-sizing: border-box" задає щоб розмір елемента враховував його рамку та відступи всередині нього. В класі "service-name" задаються параметри шрифту. Задається шрифт "Anonymous Pro" для тексту назви послуги. Прописується звичайний стиль шрифту, товщина шрифту 400, розмір шрифту 32 пікселі та

встановлюється висота рядка 32 пікселі. Це допоможе створити зручне читання тексту.

Дальше ми будемо стилізувати елементи згортання та розгортання. На рисунку 3.49, ви можете бачити фрагмент коду який стилізує дані елементи. В рядку 70 ми звертаємося до класу "service-toggle", спочатку ми додаємо відступ зліва на 10 пікселів, щоб відсунути елемент від назви послуги, потім встановлюємо розмір шрифту 28 пікселів, додаємо внутрішній відступ 5 пікселів навколо вмісту елемента, перетворює елемент на flex-контейнер, що дозволяє використовувати властивості "flexbox" і в кінці вирівнюємо вміст по горизонталі та по вертикалі.

```
70  .service-toggle {
71    margin-left: 10px;
72    font-size: 28px;
73    padding: 5px;
74    display: flex;
75    justify-content: center;
76    align-items: center;
77  }
78  .service-description {
79    max-height: 0;
80    opacity: 0;
81    font-family: 'Anonymous Pro';
82    font-style: normal;
83    font-weight: 400;
84    font-size: 28px;
85    line-height: 32px;
86    overflow: hidden;
87    transition: max-height 0.7s ease, opacity 0.7s ease, padding 0.7s ease;
88  }
```

Рисунок 3.51 – Стилізація елементів згортання та розгортання

І останнє що ми стилізуємо, це будуть елементи які використовуються для керування станом розгортання і згортання.

```
89  .service-description.open {
90    max-height: 1000px;
91    opacity: 1;
92    border-top: 1px solid #D4D4D4;
93    padding: 15px;
94  }
95  .service-description.closed {
96    max-height: 0;
97    opacity: 0;
98    padding: 0;
99    transition: max-height 0.5s ease, opacity 0.5s ease, padding 0.5s ease;
100 }
```

Рисунок 3.52 – Стилізація елементів керування стану розгортання та згортання

На рисунку 3.51, ви можете побачити стилізацію класу "service-description" у закритому та відкритому стані. У відкритому стані в нас встановлюється максимальна висота елемента 1000 пікселів, встановлюється максимальна прозорість, додається верхня межа товщиною 1 піксель та внутрішній відступ 15 пікселів навколо вмісту елемента.

В закритому стані в нас встановлюється максимальна висота елемента 0 пікселів, встановлюється прозорість елемента на 0, роблячи його повністю невидимим, видаляється внутрішній відступ та прописується плавних перехід який анімує зміну максимальної висоти, прозорість та відступ від елементів.

3.3.3 Етапи роботи

Розділ "Етапи роботи" є також дуже важливим на сайті, він дає змогу ознайомитися клієнту з етапами роботи над проєктом від самого початку і до кінця. Чітке усвідомлення кожного етапу роботи дозволяє краще планувати свої побажання по проєкту та розуміти чого очікувати в подальшому в розробці проєкту.

Як завжди написання кожного нашого блоку починається із створення контейнера для розділу та написання заголовку. На рисунку 3.52, ви можете побачити код який стилізує головний контейнер і заголовок.

```
1  <template>
2    <div class="work-stages">
3      <div class="work-title">
4        <h2>етапи роботи</h2>
5      </div>
```

Рисунок 3.53 – Створення контейнера та заголовку в "Етапи роботи"

Дальше в нас йде створення контейнера для всіх етапів робіт який має круглясту форму та створення контейнера для кожного етапу роботи (рис.3.53).

```
<div class="stage-circles">
  <div class="circle">
    <div class="center">
        <div class="circle-number">1</div>
        <div class="circle-text">Дизайн-концепція</div>
    </div>
    <p>Ви описуєте ціль і задачу роботи, показуєте приклади які вам подобаються.</p>
  </div>
```

Рисунок 3.54 – Створення круглястого контейнера

Також на рисунку можна побачити тег "div" з класом "circle-number" він відповідає за номер етапу, клас "circle-text" відповідає за назву етапу та тег "p" в якому написаний опис етапу. Таким самим чином створюються ще 3 круги, після чого ми приступаємо до стилізації нашого блоку та кругів.

```
.work-stages {
  margin: 50px 0;
  text-align: center;
  border-top: 1px solid #000;
}
.stage-title {
  border-bottom: 1px solid #000;
  padding-bottom: 10px;
  margin-bottom: 20px;
}
.stage-circles {
  display: flex;
  justify-content: center;
  margin: 120px 0 80px;
```

Рисунок 3.55 – Стилізація блок "Етапи роботи"

На рисунку 3.54, ви можете побачити стилізацію класів: work-stages, stage-title та stage-circles. В класі "work-stages" застосовує верхній і нижній відступи на 50 пікселів, вирівнюється текс по цетру та додається верхній кордон шириною 1 піксель.

В класі "stage-title" додається нижня межа шириною 1 піксель, внутрішній відступ в 10 пікселів та зовнішній відступ в 20 пікселів.

В класі "stage-circles" встановлюється флекс-контейнер, вирівнюються всі круги по цетру та застосовуються зовнішні відступи.

Після стилізації секції "Етапи роботи" можна приступити до стилізації кружків. Стилізація кружків розпочнеться з того, що ми змінюємо їх позиціонування за допомогою властивості "position: relative;" (рис. 3.55).

```
.circle {
  position: relative;
  width: 295px;
  height: 295px;
  color: #000;
  background-color: #fff;
  border-radius: 50%;
  border: 1px solid black;
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: center;
  transition: 0.5s ease;
}
```

Рисунок 3.56 – Стилізація кружка

Також задається висота і широта кружка по 295 пікселів, що робить його квадратом, встановлюється чорний колір текст, та білий колір фону. Щоб круг в нас не був квадратом ми застосовуємо до нього "border-radius: 50%" щоб заокруглити, додаємо рамку шириною 1 піксель. Застосовуємо флекс-контейнер, задаємо напрямок розміщення, вирівнюємо вміст всередині флекс-контейнера та додаємо плавний перехід при наведені.

Стилізуємо тег "p" в якому в нас розміщено текст який описує етап робіт. На рисунку 3.56, ви можете побачити фрагмент коду для стилізації тегу.

```
.circle p {
  color: #fff;
  width: 200px;
  padding: 25px 5px;
  font-family: 'Anonymous Pro';
  font-style: normal;
  font-weight: 700;
  font-size: 9px;
  line-height: 9px;
}
```

Рисунок 3.57 – Стилізація параграфа

В рядку 70 встановлюється білий колір тексту, задається ширина елемента 200 пікселів, додається внутрішній відступ зверху та знизу 25 пікселів та з ліва та справа по 5 пікселів. Встановлюється шрифт, задається нормальний стиль шрифту, встановлюється жирність шрифту, встановлюється розмір шрифту 9 пікселів та встановлюємо висоту рядка.

Наш круг є інтерактивним тобто коли на нього наводять курсор він має змінювати свій стиль. На рисунку 5.57, ви можете побачити код який робить наш круг інтерактивним.

```
79  .circle:hover {
80    background-color: #000;
81    color: #fff;
82  }
```

Рисунок 3.58 – Стилізація інтерактивності круга

В рядку 79 ми бачимо звернення до класу "circle" з ефектом "hover" це ефект який дозволяє при наведені на ньому змінювати свій вигляд. При наведені на нього в нас буде змінювати колір фону круга на чорний, а текст в середині круга буде ставати білим.

В середині нашого круга також є число яке відповідає послідовність кожного етапу. Це число також потребує стилізації. На рисунку 3.58, ви можете побачити як стилізується клас "circle-number".

```
83  .circle-number {
84    font-family: 'Anonymous Pro';
85    font-style: normal;
86    font-weight: 700;
87    font-size: 18px;
88    line-height: 15px;
89  }
```

Рисунок 3.59 – Стилізація числа послідовності

Встановлюється шрифт "Anonymous Pro", задається нормальний стиль шрифту, встановлюється жирність шрифту 700 або bold, розмір шрифту 18 пікселів, а висота рядка 15 пікселів.

І на кінець нам залишилося стилізувати 3 класи які відповідають за вирівнювання елементів всередині контейнера, шрифт, розмір та інші властивості тексту (рис. 5.59).

```
.center{
  display: flex;
  flex-direction: column;
  justify-content: end;
  align-items: end;
  margin-top: 65px;
}
.circle-text {
  margin-top: 15px;
  font-family: 'Anonymous Pro';
  font-style: normal;
  font-weight: 700;
  font-size: 18px;
  line-height: 18px;
}
.stage-info {
  margin-top: 20px;
}
```

Рисунок 5.60 – Стилізація класів для вирівнювання елементів всередині контейнера

В класі "center" встановлюються стилі для центрування вмісту всередині. В ньому використовуються властивості флекс-бокса для вирівнювання елементів. В класі "circle-text" встановлюються стилі для тексту які розміщені поруч з кружками. І в останньому класі "stage-info" прописані стилі для додаткової інформації про кожен етап роботи.

3.3.4 Футер

Останній блок на сайті це є футер. В футері в нас розміщена контактна інформація, така як: номер телефону, посилання на соціальні мережі та

електронна пошта. На рисунку 5.60, ви можете побачити код який створює футер веб-сайту.

```
<template>
  <footer class="footer">
    <div class="container">
      <h2>контакти</h2>
      <ul>
        <li class="number"><a href="tel:+380981384579">+380981384579</a></li>
        <li class="email"><a href="mailto:pavlodesign@gmail.com">pavlodesign@gmail.com</a></li>
      </ul>
      <ul class="links-footer">
        <li><a href="https://www.instagram.com/n1metspavlodesing/" target="_blank">IG</a></li>
        <li><a href="https://t.me/n1metspavlo" target="_blank">TG</a></li>
        <li><a href="https://www.behance.net/n1metspavlo" target="_blank">BH</a></li>
      </ul>
    </div>
  </footer>
</template>

```

Рисунок 5.61 – Створення футеру

В 2 рядку ми створюємо футер з класом футер, дальше створюємо контейнер, створюємо заголовок з назвою "контакти", створюємо нумерований список і в нього добавляємо контакті дані.

Вся потрібна інформація вже внесена, залишилося тільки все стилізувати. Для початку стилізуємо футер та контейнер.

```
.footer {
  background-color: #000;
  color: #fff;
  padding: 20px 0;
}
.container {
  max-width: 960px;
  margin: 0 auto;
  padding: 0 15px;
}
```

Рисунок 5.62 – Стилізація футеру

В рядку 19 ми звертаємося до класу "footer" та задаємо йому чорний колір фону, колір тексту білий та внутрішній відступ в 20 пікселів зверху та знизу, з ліва та справа по 0 пікселів. В класі "container" задається максимальна ширина

контейнера, в рядку 26 встановлюємо поля на 0 з усіх боків, прописуємо внутрішній відступ з лівого та правого боку на 15 пікселів, зверху та знизу не даємо відступ.

Висновки до розділу 3

В цьому розділі ми детально розглянули процес розробки дизайну та створення самого сайту-візитку. Дізналися як створювався кожний розділ на сайті та як він стилізувався.

Розроблений сайт відповідає сучасним стандартам дизайну та стандартам розробки веб-сайтів.

ВИСНОВОК

В ході виконання кваліфікаційної роботи був розроблений веб-сайт, а саме сайт-візитка для веб-дизайнера. Сайт виконаний за допомогою фреймворка Django та таких мов програмування як: Python, HTML, CSS, JavaScript, Vue.js та SQL. Даний сайт орієнтований на потенційних клієнтів, людей, які шукають професійні послуги з веб-дизайну. Сайт простий у використанні та надає всю корисну інформацію про веб-дизайнера.

Сайт містить такі розділи як портфоліо, послуги, відгуки клієнтів та контактну інформацію. Користувачі можуть легко знайти потрібну інформацію завдяки зручній навігації та сучасному дизайну. Інтерактивні елементи, реалізовані за допомогою Vue.js, покращують взаємодію користувачів із сайтом. Застосування SQL забезпечує надійне збереження даних та ефективний доступ до них. Загалом, створений сайт відображає професіоналізм веб-дизайнера та сприяє залученню нових клієнтів.

Крім того, у процесі розробки були застосовані сучасні підходи до веб-дизайну та програмування, що забезпечило високий рівень продуктивності та безпеки сайту. Сайт адаптований для різних пристроїв, що робить його доступним для широкого кола користувачів. Тестування сайту включало перевірку на різних браузерах і платформах, щоб забезпечити його коректну роботу в будь-яких умовах. Реалізація проекту дозволила отримати цінний досвід у розробці веб-додатків та підтвердити знання в сфері сучасних технологій веб-розробки.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Житомир.info: веб-сайт. URL: http://surl.li/ttuve (дата звернення: 20.03.2024)

2. Cursor.education: веб-сайт. URL: http://surl.li/sjinf (дата звернення: 20.03.2024)

3. Frusia.pro: веб-сайт. URL: https://frusia.pro/p/12 (дата звернення: 25.04.2024)

4. CSS.in.ua: веб-сайт. URL: http://surl.li/ttuvp (дата звернення: 15.05.2024)

5. ITWiki: веб-сайт. URL: https://itwiki.dev/front-end/css-reference (дата звернення: 25.04.2024)

6. Vue.js: веб-сайт. URL: https://ua.vuejs.org/guide/introduction.html (дата звернення: 26.04.2024)

7. Foxminded: веб-сайт. URL: https://foxminded.ua/vue-js/ (дата звернення: 27.04.2024)

8. Cases: веб-сайт. URL: http://surl.li/tzwwj (дата звернення: 29.04.2024)

9. GOIT: веб-сайт. URL: http://surl.li/tqjlo (дата звернення: 29.04.2024)

10. Jstify Community: веб-сайт. URL: http://surl.li/ttuxz (дата звернення: 05.05.2024)

11. Мій клас: веб-сайт. URL: http://surl.li/oaiwq (дата звернення: 05.05.2024)

12. CSS.in.ua: веб-сайт. URL: http://surl.li/tzwwk (дата звернення: 08.05.2024)

13. CSS.in.ua: веб-сайт. URL: http://surl.li/tzwwm (дата звернення: 08.05.2024)

14. IT Step: веб-сайт. URL: http://surl.li/tzwwo (дата звернення: 10.05.2024)

15. Amazon Web Services: веб-сайт. URL: http://surl.li/tzwwq (дата звернення: 12.05.2024)

16. Django Girls: веб-сайт. URL: http://surl.li/ttuyk (дата звернення: 13.05.2024)

17. MDN Web Docs: веб-сайт. URL: http://surl.li/ttuyg (дата звернення: 18.05.2024)

18. Wezom Academy: веб-сайт. URL: http://surl.li/tzwwt (дата звернення: 19.05.2024)

19. Muraha.eu: веб-сайт. URL: http://surl.li/tzwwv (дата звернення: 22.05.2024)

20. Lemon School: веб-сайт. URL: http://surl.li/tzwxc (дата звернення: 23.05.2024)

21. Вікіпедія: веб-сайт. URL: https://uk.wikipedia.org/wiki/Figma (дата звернення: 24.05.2024)

22. MDN Web Docs: веб-сайт. URL: http://surl.li/ttuzb (дата звернення: 24.05.2024)

ДОДАТКИ

Додаток А

Код App.vue

```
<template>
  <div>
    <header>
      <nav>
        <div class="left-side">
          <ul class="right-lines">
                    <li><a  href="/"><img  src="./assets/logo.png"
alt="Логотип"></a></li>
                                                                <li><a
@click="scrollToSection('works')">Роботи</a></li>
                                                                <li><a
@click="scrollToSection('services')">Послуги</a></li>
                  <li><a @click="scrollToSection('work-stages')">Етапи
роботи</a></li>
                                                                <li><a
@click="scrollToSection('contacts')">Контакти</a></li>
            </ul>
            <ul class="links-right">
                                                                <li><a
href="https://www.instagram.com/n1metspavlodesing/"
target="_blank">IG</a></li>
                                  <li><a  href="https://t.me/n1metspavlo"
target="_blank">TG</a></li>
                      <li><a  href="https://www.behance.net/n1metspavlo"
target="_blank">BH</a></li>
            </ul>
          </div>
        </nav>
      </header>
      <div id="prof" class="prof">
        <profComp />
```

```
    </div>
    <div id="works" class="section">
      <workComp />
    </div>
    <div id="services" class="section">
      <serviceComp />
    </div>
    <div id="work-stages" class="work-stages">
      <stageComp />
    </div>
    <div id="contacts" class="footer">
      <footerComp />
    </div>
  </div>
</template>

<script>
import workComp from './components/worksComp.vue';
import serviceComp from './components/serviceComp.vue';
import stageComp from './components/stageComp.vue';
import profComp from './components/mainComp.vue';
import footerComp from './components/footerComp.vue';

export default {
  name: 'App',
  components: {
    workComp,
    serviceComp,
    stageComp,
    profComp,
    footerComp
  },
  methods: {
    scrollToSection(sectionId) {
      const element = document.getElementById(sectionId);
      if (element) {
        const offset = 110;
        const offsetPosition = element.offsetTop - offset;
        window.scrollTo({
```

```
          top: offsetPosition,
          behavior: 'smooth'
        });
      }
    }
  }
}
</script>

<style>
body {
  margin: 100px 0 0 0;
}
#app {
  font-family: Avenir, Helvetica, Arial, sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  text-align: center;
  color: #2c3e50;
  margin-top: 60px;
}
header {
  background-color: #fff;
  margin: 0 auto;
  border-bottom: 1px solid #D4D4D4;
  position: fixed;
  top: 0;
  width: 100%;
  z-index: 1000;
}
nav {
  display: flex;
  align-items: stretch;
  height: 100px;
  margin: 0 auto;
  justify-content: center;
}
.left-side {
  display: flex;
```

```
  align-items: center;
}
.right-lines {
  display: flex;
  justify-content: center;
  align-items: center;
}
.right-lines li {
  border-right: 1px solid #D4D4D4;
  padding: 0 40px;
  height: 100px;
  display: flex;
  justify-content: center;
  align-items: center;
  font-family: 'Anonymous Pro';
  font-style: normal;
  font-weight: 400;
  font-size: 18px;
  line-height: 25px;
}
.right-lines li a {
  cursor: pointer;
  text-transform: uppercase;
}
.left-side ul, .right-side ul {
  list-style-type: none;
  padding: 0;
}
.left-side a, .right-side a {
  text-decoration: none;
  color: #444;
  transition: all 0.5s ease;
}
.links-right {
  display: flex;
  justify-content: center;
  align-items: center;
  margin-left: 30px;
  font-family: 'Big Shoulders Display';
```

```
  font-style: normal;
  font-weight: 400;
  font-size: 18px;
  line-height: 22px;
}
.links-right li {
  width: 60px;
  height: 60px;
  border-radius: 50%;
  background-color: black;
  border: 1px solid black;
}
.links-right li a {
  display: flex;
  justify-content: center;
  align-items: center;
  text-decoration: none;
  color: white;
  line-height: 60px;
  transition: 0.5s ease-in-out;
}
.links-right li:hover {
  background-color: white;
}
.links-right li a:hover {
  color: black;
}
ul li img {
  object-fit: cover;
  vertical-align: middle;
}
</style>
```

Код mainComp.vue:

```
<template>
  <div class="web-designer">
    <h1>веб-дизайнер</h1>
```

```
        <div class="content">
          <div class="left-side">
               <p class="title">Вітаю! Мене звати<br>Павло, я створюю
дизайн сайтів</p>
                  <p class="text">розроблю вам сайт з унікальним і
функціональним дизайном</p>
          </div>
          <div class="photo">
            <img src="../assets/prof.png" alt="Веб-дизайнер Павло">
          </div>
          <div class="right-side">
            <div>
              <div>сайти</div>
              <div>креативи</div>
              <div>лендінги</div>
              <div>дизайн-макети</div>
              <div>інтернет-магазини</div>
            </div>
          </div>
        </div>
      </template>

      <script>
      export default {
        name: 'WebDesigner',
      }
      </script>

      <style scoped>
      .web-designer h1 {
        font-family: 'Pacifico';
        font-style: normal;
        font-weight: 400;
        font-size: 175px;
        line-height: 307px;
        color: black;
        margin: 50px 0 70px;
      }
```

```
.content {
  display: flex;
  justify-content: space-between;
  align-items: center;
  width: 1200px;
  margin: 0 auto;
}
.left-side {
  width: 395px;
  display: flex;
  flex-direction: column;
  text-align: start;
}
.left-side .title {
  font-family: 'Anonymous Pro';
  font-style: normal;
  font-weight: 400;
  font-size: 32px;
  line-height: 32px;
  margin-bottom: 10px;
}
.left-side .text {
  font-family: 'Anonymous Pro';
  font-style: normal;
  font-weight: 400;
  font-size: 18px;
  line-height: 18px;
  color: #5B5B5B;
  margin-top: 10px;
}
.photo img {
  max-width: 100%;
  display: block;
  margin: 0 auto;
}
.right-side {
  width: 475px;
  display: flex;
  flex-direction: column;
```

```
}
.right-side > div {
  display: flex;
  flex-wrap: wrap;
  justify-content: center;

}
.right-side div div {
  flex-grow: 0;
  padding: 10px;
  margin: 10px;
  text-align: center;
  font-family: 'Anonymous Pro';
  font-style: normal;
  font-weight: 400;
  font-size: 32px;
  line-height: 32px;
  color: #000000;
  border: 1px solid #000000;
  border-radius: 12px;
}
</style>
```

Код serviceComp.vue

```
<template>
  <div class="web-designer">
    <h1>веб-дизайнер</h1>
    <div class="content">
      <div class="left-side">
           <p class="title">Вітаю! Мене звати<br>Павло, я створюю
дизайн сайтів</p>
             <p class="text">розроблю вам сайт з унікальним і
функціональним дизайном</p>
      </div>
      <div class="photo">
        <img src="../assets/prof.png" alt="Веб-дизайнер Павло">
      </div>
```

```
      <div class="right-side">
        <div>
          <div>сайти</div>
          <div>креативи</div>
          <div>лендінги</div>
          <div>дизайн-макети</div>
          <div>інтернет-магазини</div>
        </div>
      </div>
    </div>
  </div>
</template>

<script>
export default {
  name: 'WebDesigner',
}
</script>

<style scoped>
.web-designer h1 {
  font-family: 'Pacifico';
  font-style: normal;
  font-weight: 400;
  font-size: 175px;
  line-height: 307px;
  color: black;
  margin: 50px 0 70px;
}
.content {
  display: flex;
  justify-content: space-between;
  align-items: center;
  width: 1200px;
  margin: 0 auto;
}
.left-side {
  width: 395px;
  display: flex;
```

```
  flex-direction: column;
  text-align: start;
}
.left-side .title {
  font-family: 'Anonymous Pro';
  font-style: normal;
  font-weight: 400;
  font-size: 32px;
  line-height: 32px;
  margin-bottom: 10px;
}
.left-side .text {
  font-family: 'Anonymous Pro';
  font-style: normal;
  font-weight: 400;
  font-size: 18px;
  line-height: 18px;
  color: #5B5B5B;
  margin-top: 10px;
}
.photo img {
  max-width: 100%;
  display: block;
  margin: 0 auto;
}
.right-side {
  width: 475px;
  display: flex;
  flex-direction: column;
}
.right-side > div {
  display: flex;
  flex-wrap: wrap;
  justify-content: center;

}
.right-side div div {
  flex-grow: 0;
  padding: 10px;
```

```
    margin: 10px;
    text-align: center;
    font-family: 'Anonymous Pro';
    font-style: normal;
    font-weight: 400;
    font-size: 32px;
    line-height: 32px;
    color: #000000;
    border: 1px solid #000000;
    border-radius: 12px;
  }
  </style>
```

Код stageComp.vue

```
  <template>
    <div class="work-stages">
      <div class="work-title">
        <h2>етапи роботи</h2>
      </div>
      <div class="stage-circles">
        <div class="circle">
          <div class="center">
              <div class="circle-number">1</div>
              <div class="circle-text">Дизайн-концепція</div>
          </div>
            <p>Ви описуєте ціль і задачу роботи, показуєте приклади які вам подобаються.</p>
        </div>
        <div class="circle">
          <div class="center">
              <div class="circle-number">2</div>
              <div class="circle-text">Інтерв'ю</div>
          </div>
           <p>Проводимо детальне інтерв'ю, щоб глибше зрозуміти ваші потреби, очікування та бачення проекту.</p>
        </div>
        <div class="circle">
```

```
            <div class="center">
                <div class="circle-number">3</div>
                <div class="circle-text">Підготовка</div>
            </div>
              <p>Після інтерв'ю ми розробляємо детальний план проекту, включаючи структуру, технічні вимоги та етапи реалізації.</p>
          </div>
          <div class="circle">
            <div class="center">
                <div class="circle-number">4</div>
                <div class="circle-text">Дизайн</div>
            </div>
                  <p>На основі затверджених макетів та концепцій ми переходимо до створення повноцінного дизайну.</p>
          </div>
        </div>
      </div>
    </template>

    <style scoped>
    .work-stages {
      margin: 50px 0;
      text-align: center;
      border-top: 1px solid #000;
    }
    .stage-title {
      border-bottom: 1px solid #000;
      padding-bottom: 10px;
      margin-bottom: 20px;
    }
    .stage-circles {
      display: flex;
      justify-content: center;
      margin: 120px 0 80px;
    }
    .circle {
      position: relative;
      width: 295px;
      height: 295px;
```

```
  color: #000;
  background-color: #fff;
  border-radius: 50%;
  border: 1px solid black;
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: center;
  transition: 0.5s ease;
}
.circle p {
  color: #fff;
  width: 200px;
  padding: 25px 5px;
  font-family: 'Anonymous Pro';
  font-style: normal;
  font-weight: 700;
  font-size: 9px;
  line-height: 9px;
}
.circle:hover {
  background-color: #000;
  color: #fff;
}
.circle-number {
  font-family: 'Anonymous Pro';
  font-style: normal;
  font-weight: 700;
  font-size: 18px;
  line-height: 15px;
}
.center{
  display: flex;
  flex-direction: column;
  justify-content: end;
  align-items: end;
  margin-top: 65px;
}
.circle-text {
```

```
    margin-top: 15px;
    font-family: 'Anonymous Pro';
    font-style: normal;
    font-weight: 700;
    font-size: 18px;
    line-height: 18px;
  }
  .stage-info {
    margin-top: 20px;
  }
  </style>
```

Код footerComp.vue

```
  <template>
    <footer class="footer">
      <div class="container">
        <h2>контакти</h2>
        <ul>
                                                <li     class="number"><a
href="tel:+380981384579">+380981384579</a></li>
                                                 <li     class="email"><a
href="mailto:pavlodesign@gmail.com">pavlodesign@gmail.com</a></li>
        </ul>
        <ul class="links-footer">
            <li><a href="https://www.instagram.com/n1metspavlodesing/"
target="_blank">IG</a></li>
                                  <li><a   href="https://t.me/n1metspavlo"
target="_blank">TG</a></li>
                       <li><a  href="https://www.behance.net/n1metspavlo"
target="_blank">BH</a></li>
        </ul>
      </div>
    </footer>
  </template>

  <style scoped>
  .footer {
```

```
  background-color: #000;
  color: #fff;
  padding: 20px 0;
}
.container {
  max-width: 960px;
  margin: 0 auto;
  padding: 0 15px;
}
.container h2 {
  font-family: 'Anonymous Pro';
  font-style: normal;
  font-weight: 700;
  font-size: 48px;
  line-height: 48px;
}
ul {
  list-style: none;
  padding: 0;
}
ul .number {
  font-family: 'Anonymous Pro';
  font-style: normal;
  font-weight: 400;
  font-size: 20px;
  line-height: 20px;
  margin-bottom: 20px;
}
ul .email {
  font-family: 'Anonymous Pro';
  font-style: normal;
  font-weight: 400;
  font-size: 24px;
  line-height: 24px;
}
ul .number a, ul .email a {
  color: white;
  text-decoration: none;
}
```

```
li {
  margin-bottom: 5px;
}
.links-footer {
  display: flex;
  justify-content: center;
  align-items: center;
}
.links-footer li {
  width: 60px;
  height: 60px;
  border-radius: 50%;
  background-color: white;
  border: 1px solid white;
  transition: 0.3s ease-in-out;
}
.links-footer li a {
  display: flex;
  justify-content: center;
  align-items: center;
  text-decoration: none;
  color: black;
  line-height: 60px;
  font-family: 'Big Shoulders Display';
  font-style: normal;
  font-weight: 400;
  font-size: 18px;
  transition: 0.3s ease-in-out;
}
.links-footer li:hover {
  background-color: black;
}
.links-footer li a:hover {
  color: white;
}
</style>
```

метадані

Заголовок
Сайт-візитка для веб дизайнера

Автор
Німець П. Науковий керівник / Експерт

підрозділ
King Danylo University

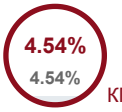
Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв	ß	0
Інтервали	A→	0
Мікропробіли	·	0
Білі знаки	ß	0
Парафрази (SmartMarks)	a	25

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.



КП 1

КП 2

25
Довжина фрази для коефіцієнта подібності 2

12390
Кількість слів

87270
Кількість символів

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)	
1	http://repository.ukd.edu.ua/bitstream/handle/123456789/195/%D0%86%D0%B2%D0%B0%D0%BD%D1%8E%D0%BA%20%D0%A2.%20%D0%AF..pdf?sequence=1	36	0.29 %
2	https://core.ac.uk/download/pdf/78394541.pdf	33	0.27 %
3	https://dspace.uzhnu.edu.ua/jspui/bitstream/lib/35565/1/WEB.pdf	30	0.24 %
4	http://repository.ukd.edu.ua/bitstream/handle/123456789/195/%D0%86%D0%B2%D0%B0%D0%BD%D1%8E%D0%BA%20%D0%A2.%20%D0%AF..pdf?sequence=1	25	0.20 %
5	https://frusia.pro/p/12	20	0.16 %