ДИПЛОМНА РОБОТА

ДР.ПІс – 01.00.000 ПЗ

Група ПІс-2015

Бедрус О. Я.

2019

ПВНЗ Університет Короля Данила

Кафедра Інформаційних технологій та програмної інженерії

УДК 004

# ДИПЛОМНА РОБОТА

Тема ***Інформаційний веб-ресурс портфоліо дизайнера***

Напрям підготовки ***6.050103 «Програмна інженерія»***
(код і назва спеціальності)

## ПОЯСНЮВАЛЬНА ЗАПИСКА

ДР.ПІс – 01.00.000 ПЗ
(позначення)

Студент

*Бедрус О.Я.*
(підпис) (дата) (розшифрування підпису)

Керівник проекту
*к.т.н.* *Мануляк І.З.*
(посада) (підпис) (дата) (розшифрування підпису)

Нормоконтроль
*к.т.н.* *Мануляк І.З.*
(посада) (підпис) (дата) (розшифрування підпису)

Допускається до захисту
Завідувач кафедри
*д.т.н.,доц.* *Мельничук С.І.*
(посада) (підпис) (дата) (розшифрування підпису)

2019

**ПВНЗ УНІВЕРСИТЕТ КОРОЛЯ ДАНИЛА**

Факультет *Інформаційних технологій*
Кафедра *Інформаційних технологій та програмної інженерії*
Напрям підготовки *6.050103 «Програмна інженерія»*

**ЗАТВЕРДЖУЮ:**
Завідувач кафедри ІТПІ
_____ *С.І. Мельничук*
"____" _____ 2019 р.

**ЗАВДАННЯ**
**НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ)**

Студенту *Бедрусу Оресту Ярославовичу*

1. Тема проекту (роботи) *Інформаційний веб-ресурс портфоліо дизайнера*

Затверджена наказом ректора Університету Короля Данила від

2. Термін здачі студентом закінченого проекту (роботи) *10.06.2019 р.*
3. Вихідні дані до проекту (роботи) *Sketch, Principle, Zeplin, JavaScript, Vue.js, Stylus,Webpack*

4. Зміст пояснювальної записки (перелік питань, що їх належить розробити) *1. Огляд та аналіз інформаційного веб-ресурсу портфоліо. 2. Вибір та розробка програмних засобів для розробки веб сайту. 3. Розробка інформаційного веб-ресурсу портфоліо дизайнера засобами Sketch, Principle та JavaScript*

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) *Створення структури сайту, функціональних можливостей сайту. Етапи створення дизайну та анімацій, використані технології та ПЗ*

6. Консультанти з проекту (роботи), із зазначенням розділів проекту, що стосуються

| Розділ | Консультант | Підпис, дата | |
|---|---|---|---|
| | | Завдання видав | Завдання прийняв |
| | | | |
| | | | |
| | | | |
| | | | |

7. Дата видачі завдання 30.10.2018

Керівник Мануляк І.З.

Завдання прийняв до виконання Бедрус О.Я.

## КАЛЕНДАРНИЙ ПЛАН

| Пор № | Назва етапів дипломного проекту (роботи) | Термін виконання етапів проекту (роботи) | Примітка |
|---|---|---|---|
| 1. | Огляд та загальний аналіз web-портфоліо, порівняльний аналіз популярних рішень та їх прямих і можливих аналогів | 30.11.2018 | |
| 2. | Вибір та обґрунтування технологій розробки | 19.12.2018 | |
| 3. | Розробка дизайну сайту в середовищі Sketch, розробка сайту засобами HTML, CSS, JavaScript | 21.02.2019 | |
| 4. | Оформлення пояснювальної записки | 31.05.2019 | |
| 5. | Оформлення графічного матеріалу та підготовка до захисту роботи | 04.06.2019 | |

Студент-дипломник *Бедрус О.Я.*
(підпис) (розшифровка підпису)

Керівник проекту *Мануляк І.З.*
(підпис) (розшифровка підпису)

# АНОТАЦІЯ

Темою даної дипломної роботи є створення веб-портфоліо дизайнера.

В даній дипломній роботі розроблено веб-сервіс, який вирішує проблеми, пов’язані зі складністю просування своїх робіт, а також прискорює пошук клієнтів та роботодавців. Розроблено зручний і швидкий інтерфейс користувача та безпечну і надійну серверну частину.

## SUMMARY

The topic of this thesis is the creation of a designer’s web-based portfolio.

This thesis features a web-service that solves problems related to the challenges of promoting one’s work, as well as speeds up the search for clients and employers. We present a user-friendly and fast user interface and a secure and reliable back-end development.

# РЕФЕРАТ

Розрахунково-пояснювальна записка: 62 сторінки, 22 рисунки, 1 додаток.

Дипломна робота присвячена розробці інформаційного веб-ресурсу портфоліо дизайнера. Метою роботи є розробка особистого сайту, що допоможе молодому спеціалісту вийти на ринок IT.

Перший розділ складається з огляду портфоліо та порівняльного аналізу існуючих рішень на ринку, аналізу вимог та постановки задачі.

В другому розділі описано розробку структури веб-додатку, визначення із інструментами та технологіями розробки, їх обґрунтування, а також розробка потрібних діаграм.

Третій розділ містить опис процесу розробки дизайну інтерфейсу сайту за допомогою візуальних редакторів, а також процес розробки за допомогою технологій HTML, CSS, JavaScript та його фреймворків.

# ЗМІСТ

## ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

CSS — Cascading Style Sheets;

HTML — HyperText Markup Language;

JS — JavaScript;

UI – User Interface;

UX – User Experience;

# ВСТУП

**Актуальність теми** обумовлюється тим, що з плином часу молодим людям без досвіду роботи стає дедалі важче знайти своє місце навіть на ринку стрімко збільшуваного IT сектору, адже, користуючись традиційними сервісами WEB-портфоліо дуже важко вийти на ринок і стати впізнаваним. Пов'язано це в першу чергу з великою кількістю користувачів на таких сайтах, але дедалі частіше користувачі звертають увагу на алгоритми просування користувачів та їх робіт, які сильно направлені на просування вже і без того великих і впізнаваних людей.

**Об'єктом дослідження** виступають найбільш популярні платформи для створення WEB-портфоліо, такі як Behance, Dribbble та інші сайти, які можна застосувати в якості портфоліо.

**Задачі досліджень:**

- розглянути найпопулярніші наявні види WEB-портфоліо для дизайнерів та програмістів, їх прямих та віддалених конкурентів;
- дослідити функціонал, переваги і недоліки, а також спільні та відмінні риси у найбільших мереж, проаналізувати доцільність їх використання;
- визначити найкращі актуальні технології;
- розробити свій інформаційний ресурс для особистого користування.

**Методи дослідження** - спостереження за різними наявними рішеннями та їх порівняльний аналіз, засновані на юзабіліті тестуванні (англ. Usability testing - тестування зручності використання), виявлення прямих та можливих аналогів, з'ясування доцільності використання традиційних сервісів з точки зору cost-benefit аналізу та інших непрямих вимог.

**Результати дослідження:** можна прийти до висновку, що створення свого сайту дозволить зробити власне портфоліо набагато привабливішим для потенційних клієнтів, оформити сайт у власному, унікальному дизайні, організувати правильний user flow (з англ. потік користувача, тобто перехід

користувача від одного сценарію використання до іншого), що дасть змогу викинути непотрібний функціонал та додати необхідний, а крім того використати власне доменне ім’я та зекономити фінанси, якщо порівнювати власне рішення з платними версіями сервісів, що вже існують на ринку.

# 1 ОГЛЯД ТА АНАЛІЗ ІНФОРМАЦІЙНОГО ВЕБ-РЕСУРСУ ПОРТФОЛІО

## 1.1 Загальний огляд портфоліо

Ні для кого не секрет, що за останні десятиліття мережа інтернет стрімко розвивалася – швидкість мережевого з'єднання зросла в декілька десятків, якщо не сотень, разів, а WEB-розробка докорінно змінилася і вже ніколи не буде такою, як раніше.

З розвитком нових веб-технологій, додатки і сервіси стають зручнішими, більш функціональними, простішими у користуванні і відповідно приваблюють більшу кількість користувачів.

Збільшення швидкості з'єднання, розповсюдженість мобільного інтернету і зменшення його вартості з однієї сторони, а також бурхливий розвиток WEB-технологій з іншої сприяли розвитку не однієї, а, напевно, всіх індустрій на ринку України і світу. Тепер більше не треба купувати колонку в газеті, білборд чи рекламу на телебаченні, щоб купити якусь річ, найняти працівника чи просувати свій бізнес. Всі ці можливості спочатку з'явилися в нас вдома на комп'ютері, а згодом і в кишені, на екрані смартфона.

Нові технології стрімко входять в усі сфери діяльності людини. Як і всі інші речі, що піддалися "діджиталізації", тобто оцифровуванню, дедалі більшої популярності набувають веб-портфоліо – сайти, на яких UI/UX дизайнери, промислові дизайнери, фотографи, WEB-розробники, тощо, можуть створювати свої сторінки та ділитися своїми досягненнями зі світом (рис. 1.1).

Портфоліо (від італ. portafoglio – портфель) – це зібрання робіт, напрацювань, виконаних проектів конкретної особи чи компанії [1]. В наш час портфоліо є невід'ємним атрибутом будь-якого молодого спеціаліста, адже люди з сильним портфоліо приваблюють набагато більше робочих і комерційних пропозицій. Більше того, майже всі вакансії, навіть рівня Junior, вимагають обов'язкову наявність портфоліо [2].

Але не всі володіють навичками web-верстки чи frontend-розробки, не всі мають можливість купити щомісячно домен та хостинг, а тим більше налаштувати сервер вручну.

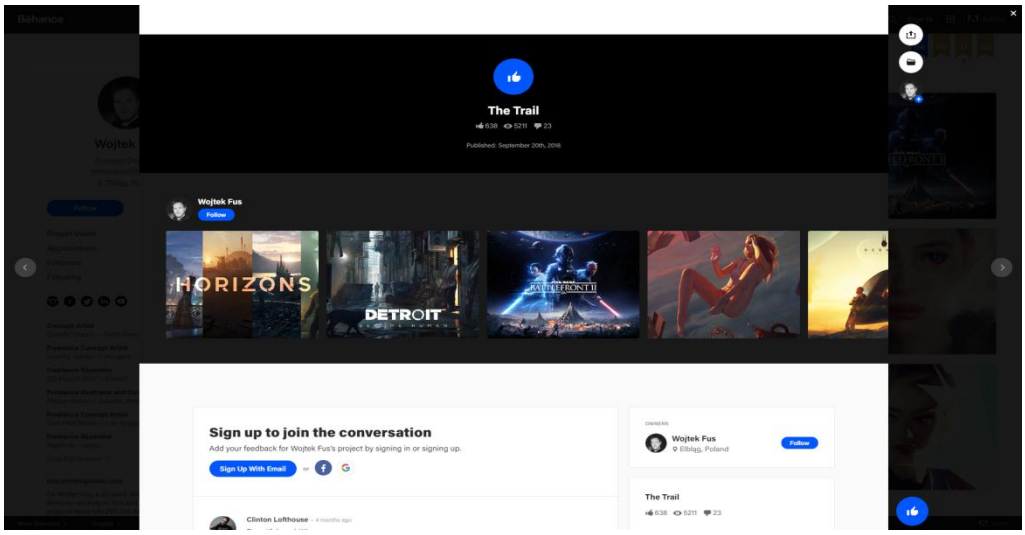


Рисунок 1.1 – Соціальні можливості веб-сервісу Behance:
“лайки”, коментарі та інші функції

Тому не дивно, що з часом з’являється все більше та більше сервісів WEB-портфоліо, де кожен охочий може створити собі сторінку та завантажувати свої роботи. З однієї сторони такі сайти корисні для компаній чи приватних підприємців, що хочуть знайти працівників, адже зазвичай розробник чи дизайнер може вказати, що він відкритий для пропозицій, а з іншої – допомогти молодим спеціалістам стати впізнаваними на ринку, адже ці ресурси мають вбудований соціальний функціонал: “лайки”, коментарі, особисті повідомлення тощо.

## 1.2 Аналіз діючих інформаційних ресурсів з портфоліо

Як було сказано вище, розвиток інтернету сприяв появі багатьох сервісів WEB-портфоліо. Найпопулярнішими, на мою думку, можна назвати такі спілки

для дизайнерів як Behance та Dribbble, а для програмістів – GitHub та CodePen. В більш широкому сенсі WEB-портфоліо можна назвати будь-який фотохостинг, серед яких найбільш відомими на сьогодні є Instagram, Pinterest та Flickr. Також для розміщення портфоліо часто обирають мікроблоги, такі як Twitter та Tumblr. І навіть враховуючи, що сервіси не спеціалізовані для розміщення портфоліо, вони все одно мають широкий інструментарій для зручного та швидкого завантаження будь-якого типу інформації.

Розглянемо більш детально сервіси Behance та Dribbble (рис. 1.2).

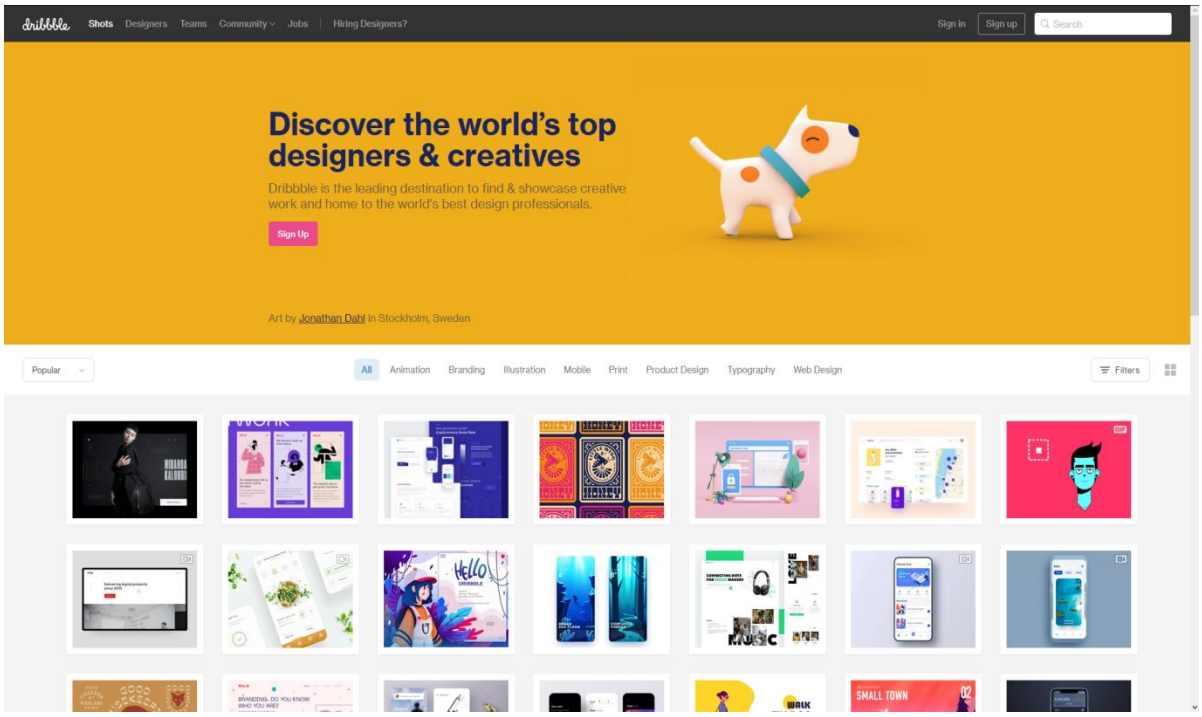


Рисунок 1.2 – Сервіс WEB-портфоліо Dribbble

Dribbble (доступний за адресою www.dribbble.com) – це онлайн-спілка для дизайнерів, аніматорів, ілюстраторів, промислових дизайнерів та ін. Сайт був заснований в 2010 році Деном Цедерхольмом (Dan Cederholm) та Річем Торнеттом (Rich Thornett) [3]. Компанія працює повністю віддалено, без офісів та штаб-квартири [4]. Наразі Dribbble – це другий по величині сервіс WEB-портфоліо в світі, поступаючись тільки сайту Behance, що належить компанії Adobe (рис. 1.4) [5, 6].

Сайт використовує членську систему тільки по запрошеннях. Це означає, що хоча зареєструватися і може кожен, але доступ до деяких функцій доступний тільки для запрошених користувачів [7, 8]. Таких користувачів називають Prospects (англ. бути перспективним). Користувачі, наділені статусом Prospects, не можуть розмістити свої роботи на головній сторінці, але можуть бути підвищені до рівня Players (англ. гравці) запрошеними користувачами. Крім того, у Players є можливість купити платну підписку Pro [7].

В цілому Dribbble дозволяє швидко зареєструватися та завантажувати свої роботи, але в нього є багато серйозних недоліків, які роблять сервіс не дуже зручним і приємним для постійного користування. Наприклад, навіть до сьогоднішнього дня найбільш суперечливою залишається схема членства на сайті, а саме те, що ніким не гарантовано, що користувача хтось запросить і таким чином переведе з групи Prospects до рангу Players, навіть якщо активно коментувати та завантажувати роботи. Це значить, що набувати популярності і просувати своє портфоліо з першого ж дня фактично неможливо, а тому цей сайт не дуже підходить для молодих дизайнерів, в яких ще немає впізнаваності на ринку [8].

До того ж Dribbble має серйозні обмеження по розміру зображення – обов'язково має бути розміром 800x480 або 400x240 та не більше 10 МБ. В іншому випадку алгоритми сайту зменшать або обріжуть зображення, що призведе до втрати частини загальної композиції і якості зображення [7]. Очевидно, що в 2019 році навіть більша версія зображення розміром 800x480 через свою малу роздільну здатність не може передати всю суть складного сайту чи мобільного додатку, враховуючи ще й те, що сайт не дає розмістити декілька зображень в одному записі [9]. Крім того, сайт обмежує кількість записів, які можна розмістити за місяць, тобто викладаючи великий проект, який складається із декількох записів, можна швидко досягти ліміту.

Все це зроблено для того, щоб просувати платну версію акаунту мережі. У платних користувачів є можливість продавати продукти через сайт, прикріплювати більше одного зображення в записі, ставити записи на таймер,

завантажувати відео, переглядати статистику та отримувати ексклюзивні пропозиції від роботодавців [10].

Прямим конкурентом Dribbble є сервіс Behance, що належить компанії Adobe [6, 11]. У нього немає платної версії, а також всі зареєстровані користувачі належать до однієї категорії, тобто не треба чекати запрошення. Але навіть в такому випадку новачку буде важко попасти на головну, адже туди зазвичай попадають найбільш популярні і великі учасники, а сторінки для дебютів, як на Dribbble, немає. Тому стати впізнаваним на такій платформі також досить важко і потребує багато часу і зв'язків.

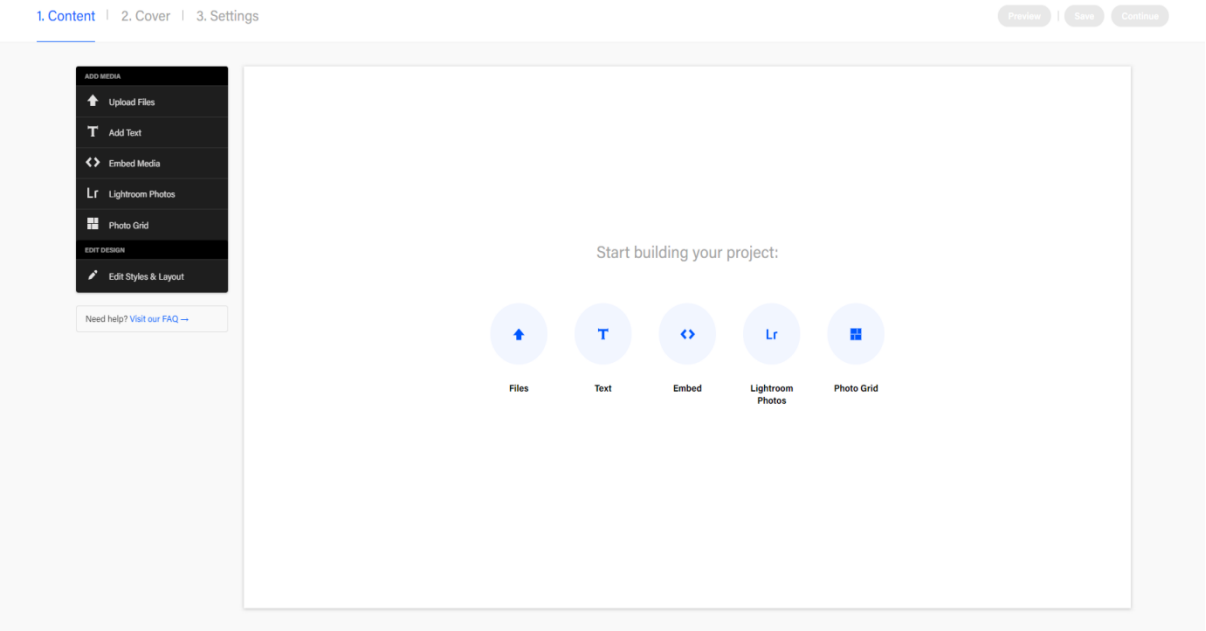


Рисунок 1.3 – Вікно створення проекту на сайті Behance

До однозначної переваги Behance над Dribbble можна віднести створення та редагування проектів (рис.1.5). Розмір та кількість зображень не обмежені, крім того сервіс дозволяє писати повноцінні описи завдяки вбудованому зручному текстовому редактору. Також зображення можна впорядковувати в сітки і вставляти посилання на медіа-файли.

У всіх проектів є функція попереднього перегляду, що дозволяє оцінити якість оформлення перед завантаженням, а також функція збереження без

завантаження, завдяки чому можна закінчити роботу над проектом пізніше, не втрачаючи прогресу.

Крім того, на сайті дуже зручний інтерфейс зі зрозумілими розділами, що дозволяє переглядати роботи, дивитися навчальний матеріал, який створюють провідні дизайнери в колаборації з Adobe, переглядати персоналізовану стрічку новин, засновану на історії переглядів та шукати актуальні вакансії в компаніях зі всього світу. І хоча подібний функціонал є і в Dribbble, мережа Behance справляється зі всіма розділами набагато краще.

Але, незважаючи на всі позитивні якості, у більшості WEB-портфоліо є типові недоліки. Наприклад, користувач не може налаштувати зовнішній вигляд сторінки за своїм смаком, змінити розташування чи порядок блоків, їх колір.

Більше того, всі сайти вимагають використання куки третьої сторони (third party cookies) і працюють некоректно, якщо заблокувати їх програмними засобами браузера, що спричиняє незручності для людей які не хочуть, щоб їхня інформація попала в руки сторонніх ресурсів.

## 1.3 Призначення та вимоги до розробки веб-додатку, постановка задачі

Створення особистого сайту дозволить позбавитись від непотрібного функціоналу, розробити власний, максимально зручний та ефективний інтерфейс для найлегшого донесення інформації до майбутніх клієнтів.

Якщо порівнювати особистий сайт та платну версію вже існуючих рішень, можна виявити, що створити і утримувати власний сайт фінансово набагато вигідніше навіть враховуючи хороший, безлімітний хостинг та дорогу доменну назву, адже Pro-версія Dribbble коштує від 5 до 12 доларів за місяць, в залежності від вибраного плану, що означає від 60 до 144 доларів в рік, тоді як особистий сайт можна покласти на хостинг GitHub, або використати хороший національний хостинг, ціни на безлімітний план якого можна купити всього за 2

долари в місяць і домен .design від 12 до 40 доларів за рік, що в сумі з хостингом дасть нам від 12 до 64 доларів в рік. І хоча використання красивого доменного імені не обов'язково, такі посилання набагато легше запам'ятати і приваблюють вони набагато більше клієнтів, ніж звичайні акаунти в великих мережах.

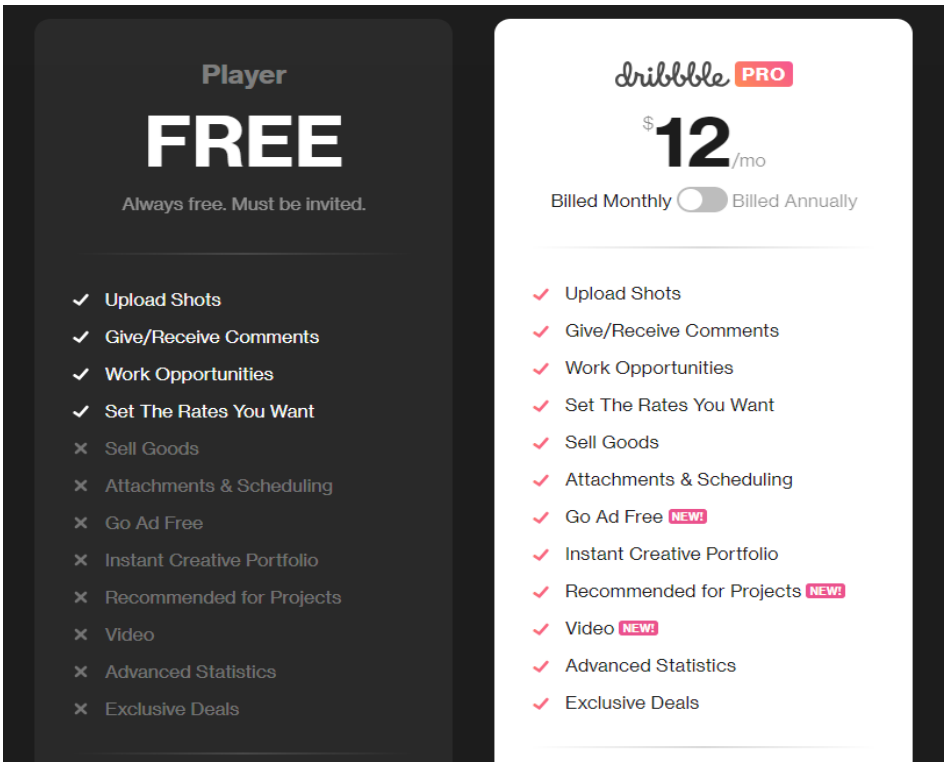


Рисунок 1.4 – Платна версія сервісу Dribbble

Окремим пунктом також можна виділити те, що створення власного сайту є набагато більш престижним, ніж використання готових рішень.

Такий ресурс дозволить підвищити зацікавленість серед потенційних роботодавців, збільшити впізнаваність власного бренду на ринку, а також розробити власний унікальний стиль, який неможливо застосувати в наявних рішеннях.

Розробка будь-якого продукту, сайту чи додатку вимагає детального аналізу вже існуючих та перспективних рішень, вивчення актуальних трендів, створення ретельного плану, побудови правильних специфікацій, хоча б декількох прототипів та постійного тестування на всіх етапах розробки. При створенні портфоліо дизайнера найбільш пріорітетним має бути зовнішній

вигляд ресурсу. Крім того, не менш важливим має бути простота, зрозумілість інтерфейсу, можливість для користувача легко та без зайвих зусиль знайти потрібну інформацію, що у великих мережах може бути проблематичним. Також сайт повинен працювати швидко та коректно, тобто правильно обробляти команди за короткий час.

Веб-ресурс використовує клієнт-серверну структуру. Це означає, що клієнтська частина виконується в браузері, і спілкується з сервером за допомогою запитів. Таким чином, при написанні веб-додатку необхідно реалізовувати як клієнтську частину (браузерну, яка подається у вигляді HTML розмітки, CSS стилів, JavaScript коду, відповідального для з'єднання з сервером) так і серверну частину, яка повинна обробляти запити клієнтів.

Для написання якісного веб-додатку необхідно мати великий набір навичок. Для клієнтської частини це:

- навички дизайну, а саме вайрфреймінгу, створення макетів сторінки, створення інтерактивних прототипів та анімацій;
- знання розмітки сторінок HTML або її препроцесорів, мови каскадних таблиць стилів CSS (в даному проекті застосовується Stylus);
- основи скриптового програмування з використанням мови JavaScript для взаємодії з сервером і підвищення інтерактивності сторінок.

Серверну веб-додатку зазвичай пишуть спеціалісти з back-end розробки, а отже це зовсім інша сфера, вона має коректно і надійно опрацьовувати запити клієнта, зберігати сесії клієнтів, а також всю інформацію, що вводиться, та бути стійкою до більшості можливих атак з боку зловмисників.

Якщо підходити до розробки серверної частини традиційним методом, то найбільш популярними технологіями для реалізації задач серверної частини є Python і його фреймворки, ASP.Net, PHP.

Проте з розвитком технологій JavaScript дедалі більше розробників відходять від стандартних методів розробки серверної частини та обирають JavaScript як альтернативу. Останнім часом JavaScript стає незамінним і необхідним інструментом для багатьох розробників. Зараз сфера його

застосування розширюється на інші області, такі як сервери, мікроконтролери, а новітні досягнення в сфері JavaScript дозволяють навіть програмувати штучний інтелект. Дедалі частіше цю мову програмування вибрають престижні університети, щоб навчати студентів основам інформатики.

Головними вимогами до розробки даного продукту є:

- зручний інтерфейс, який дасть можливість клієнтам і легко знайти та ознайомитися з потрібною інформацією;

- максимальна швидкість завантаження веб-сторінок для забезпечення приємного досвіду користувача, незалежно від наявного апаратного забезпечення;

- адекватне відображення сайту на найпопулярніших конфігураціях екранів;

- коректна робота на всіх популярних браузерах і видача чітких релевантних результатів;

- забезпечення умов для безперебійного та стабільного функціонування сайту.

Основні функціональні можливості:

- можливість переглядати основну інформацію, таку як резюме та портфоліо, список всіх актуальних робіт та посилань на соціальні мережі;

- перегляд демо-версій сайтів, мобільних додатків та інших дизайн-проектів;

- форма зворотнього зв’язку, за допомогою якої користувачі відправлятимуть свої ідеї або комерційні пропозиції.

# 2 ВИБІР ТА РОЗРОБКА ПРОГРАМНИХ ЗАСОБІВ ДЛЯ РЕАЛІЗАЦІЇ ВЕБ САЙТУ

## 2.1 Розробка структури веб-додатку та обґрунтування технологій проектування

Ні для кого не секрет, що за останні десятиліття мережа інтернет стрімко розвивалася — швидкість мережевого з'єднання зросла в декілька десятків, якщо не сотень, разів, а WEB-розробка докорінно змінилася і вже ніколи не буде такою, як раніше.

З розвитком нових веб технологій, додатки і сервіси стають зручнішими, більш функціональними, простішими у користуванні і відповідно приваблюють більшу кількість користувачів. Головним завданням будь якого додатку є можливість виконувати поставлену задачу максимально правильно, витрачаючи при цьому мінімальну кількість часу.

Ще 10-12 років назад достатньо було зверстати одну сторінку, яка буде виглядати однаково на всіх пристроях, але тепер треба враховувати інтереси якомога більшої частини користувачів. Отже, перед WEB-розробниками полягає досить складна і ресурсомістка задача: сучасний сайт має адекватно виглядати та швидко працювати на пристроях якомога більшої частини користувачів. Наприклад, сьогодні типовий користувач виходити в інтернет з мобільного телефона, смартфона, планшета, нетбука або ультрабука, ноутбука, настільного ПК, смарт-телевізора (звичайного, 4К чи навіть 8К аналога). Новітні тренди включають в себе навіть відображення WEB-сторінок на екрані смарт-годинників, типовий розмір яких складає приблизно 190x150 пікселів.

З урахуванням всього вищесказаного стає очевидно, що навіть при всьому бажанні розробників підрахувати весь спектр різноманітних роздільних здатностей, а тим більше розробити окрему версію під кожну конфігурацію екрану, просто неможливо. Ситуація стає ще складнішою, якщо також

враховувати ще те, що кожен браузер відображає сторінки не так, як всі інші. Це пов’язано із тим, що не всі браузери підтримують новітні WEB-стандарти та технології, іноді навіть з відставанням на 3-4 роки [12, 13].

Тому інженерами та веб-дизайнерами було введено спочатку термін “responsive design”, а згодом і “adaptive design” [14]. Суть технології полягає в тому, що WEB-сторінка динамічно підлаштовується під розмір екрану, при чому мінятиметься не тільки розмір чи відстань між зображеннями і текстом, а й власне розміщення елементів сторінки. Звичайно, це зовсім не звільняє WEB-розробників та дизайнерів від роботи, але допомагає значно прискорити процес розробки, адже дизайн і верстка відбуваються не під всі наявні конфігурації роздільних здатностей, а лише під найпопулярніші розміри екранів. Достатньо розробити дизайн та зверстати декілька проміжних варіантів (наприклад, телефон — планшет — комп’ютер), а все інше зроблять технології. В професійних колах такі сайти називають “резиновими”. Така назва з’явилась завдяки тому, що сайт ніби розтягується і стягується при збільшенні або зменшенні розміру сторінки.


Рисунок 2.1 — Етапи проектування та розробки веб-сайтів

Як було зазначено вище, створення будь-якого програмного продукту вимагає ретельного планування і створення детальних прототипів для спрощення подальшої розробки. На сьогодні найпопулярнішими інструментами для створення дизайн-прототипів є Sketch та Figma, а останнім часом набирає популярності Adobe XD. Не дивлячись на те, що у Adobe XD та Figma є

безкоштовні плани для користувачів, для створення прототипу сайту буде використано демо-версію Sketch.

На відміну від конкурентів, які доступні на всіх операційних системах, Sketch можна завантажити тільки для macOS. Незважаючи на це, Sketch — це одна з найстаріших та безперечно найпопулярніша програма для створення прототипів веб-сайтів та мобільних додатків, яка стала популярною завдяки своєму зручному користувацькому інтерфейсу, широкому набору функцій, хорошій швидкодії та виключній надійності в роботі.

Крім того, на відміну від Figma, для роботи в Sketch не потрібно з'єднання з мережею Інтернет, адже Figma працює тільки в браузері [15, 16].

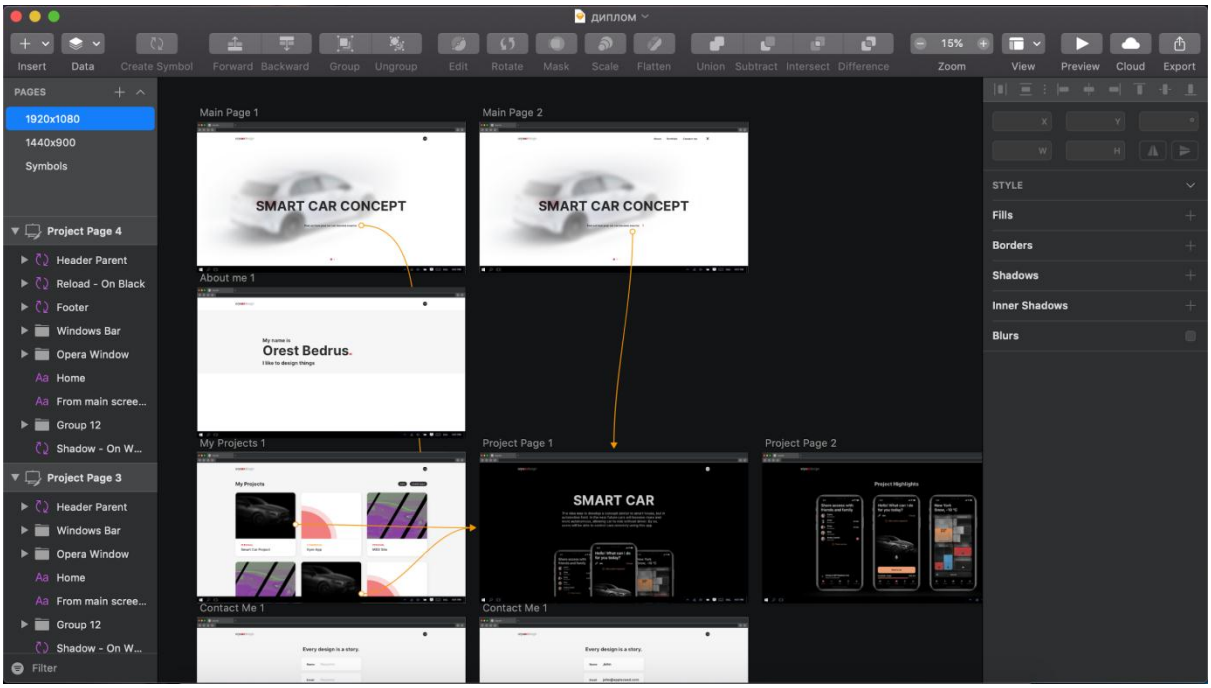


Рисунок 2.2 — Робоче середовище додатку Sketch

Широкі функціональні можливості, а також великий спектр різноманітних плагінів від сторонніх розробників дозволяють прискорити процес розробки, створити та експортувати повний набір потрібних матеріалів та даних.

Для таких цілей використовується плагін, а також однойменний додаток для macOS та інших операційних систем Zeplin. Він дозволяє експортувати файл із Sketch у вигляді веб-сторінок, подивитися розміри, відступи та копіювати CSS-стилі вибраних елементів [17].

Також, для зручного та швидкого звернення до потрібної інформації, у функціонал додатку входить можливість створення єдиної для усього проекту бібліотеки кольорів та шрифтів, що значно спрощує пошук потрібних стилів, особливо в великих проектах. Також розробники можуть експортувати векторні зображення у потрібному їм форматі, наприклад у вигляді SVG-коду чи в форматі растрових зображень [17].

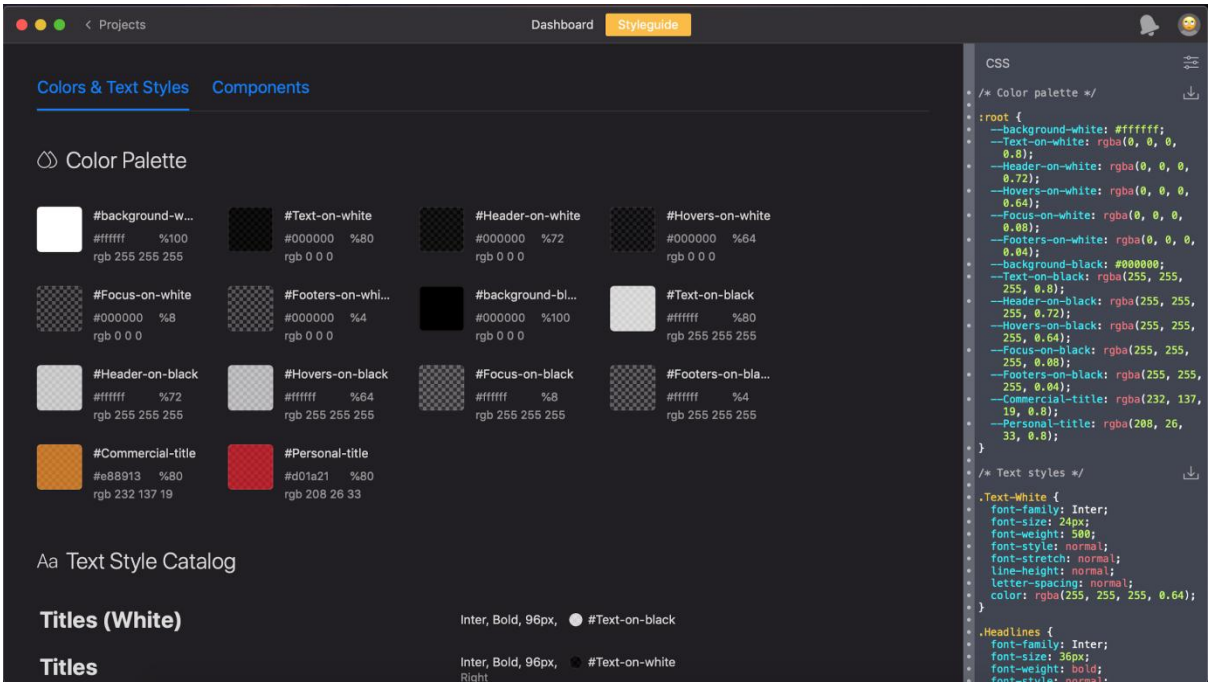


Рисунок 2.3 — Кольорова палітра, каталог текстових стилів та генератор CSS стилів в додатку Zeplin

Звичайно, Zeplin не є автоматизованим генератором веб-сайтів на основі макетів, скоріше просто прискорює створення монотонних елементів, як-от кнопок чи стилів тексту. Також цей інструмент є незамінним для роботи в групі, адже з його допомогою можна поділитись проектом з кількома людьми, а після оновлення файлів в проекті у всіх учасників автоматично буде остання версія, завантажена з “хмари”, що звільняє від потреби надсилання нової версії персонально кожному із учасників [17].

Наступними незамінним інструментами, які використовують для більш глибокого рівня дизайну, є інструменти для створення інтерактивних прототипів. Такі інструменти дозволяють швидко створити інтерактивну модель додатку чи

сайта, на якій можна проводити основні ролі взаємодії: натискати на кнопки, гортати сторінки, переходити на різні розділи і багато чого іншого. Зазвичай такі макети потрібні для того, щоб показати замовникам живу демонстрацію основних функцій проекта, який ще не готовий у вигляді реальної програми, але знаходиться на етапах ранньої розробки, де потрібно узгодити усі ключові питання, які можуть виникнути на більш пізніх етапах. Звичайно, створення таких прототипів займає більше часу, ніж створення звичайних, не інтерактивних макетів, але менше, ніж написання справжньої програми чи сайту.

Прикладами таких інструментів можна назвати безкоштовну Origami Studio, а також Principle. Створення інтерактивних прототипів також входить до можливостей Adobe XD, але на даному етапі ця програма не здатна повністю забезпечити функціонал, доступний у конкурентів. Для свого проекту я обрав програму Principle, адже вона найбільш тісно працює в тандемі зі Sketch, а також, на відміну від Origami Studio, дозволяє створити робочий простір із декількох екранів. Тут варто зазначити, що для створення будь-яких інтерактивних прототипів бажано мати хоча б два екрани, які будуть показувати початковий і кінцевий стан (наприклад, на першому екрані у нас є приховане меню із кнопкою, що його відкриває, а на другому екрані меню вже відкрите і кнопка перетворилася в хрестик, який повертає меню до початкового стану; це лише приклад, а в реальних проектах кількість таких екранів може перевищувати 100), а в Origami все відбувається за допомогою спеціальної візуальної мови програмування Patches [18] і з ускладненням проекту стає дуже важко відслідкувати, яка функція за що відповідає, хоча такий підхід має безперечні переваги над тим, який пропонує Principle. Крім того, Origami Studio набагато гірше обробляє макети веб-сторінок, адже вона розроблялась скоріше для мобільних додатків. Також набір функцій додатку Principle має перевагу над традиційними інструментами для створення анімацій, наприклад Adobe After Effects.

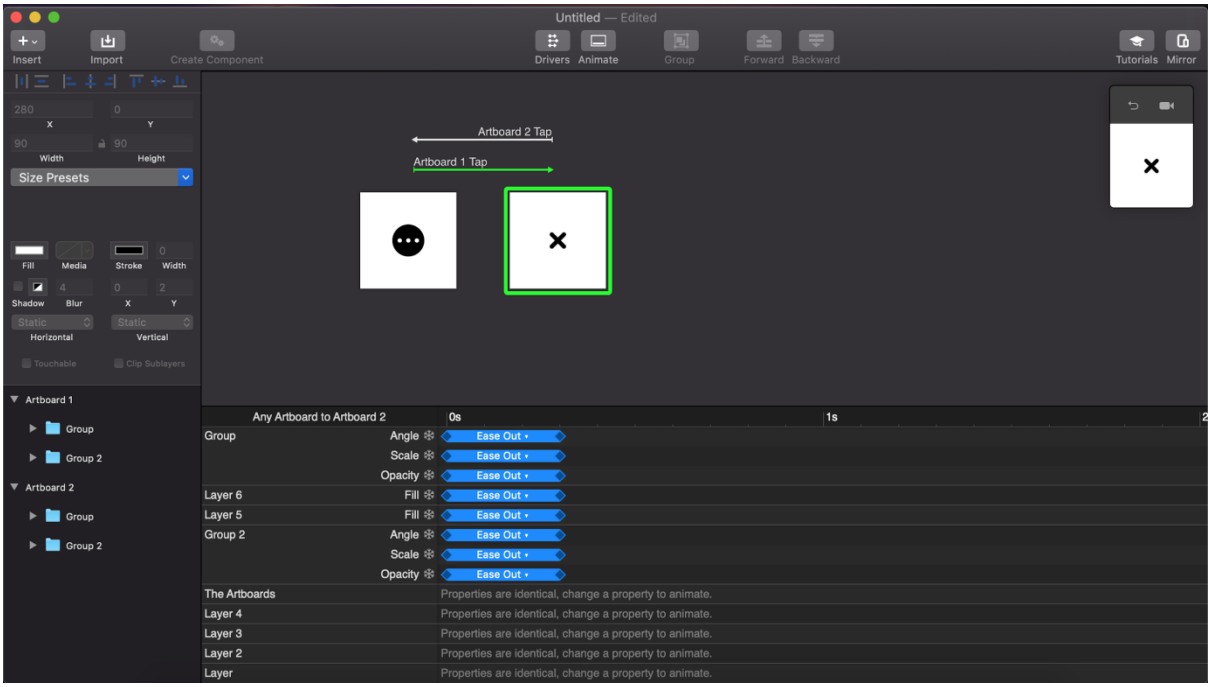


Рисунок 2.4 — Робоче середовище додатку Principle

Перевага полягає в тому, що в Principle не треба розставляти ключові кадри, прописувати анімації вручну також не потрібно, адже можна вказати лише початковий і кінцевий стан, вибрати тип і тривалість анімації, а все інше буде згенеровано програмними засобами [19]. Звичайно, при створенні великих проектів у додатку Principle також є свої недоліки, пов’язані із адекватністю обробки файлів, імпортованих із програми Sketch, тому кожну сторінку додатку чи сайту варто зберігати в окремому файлі.

Одним із найважливіших етапів розробки будь-якого сайту чи додатку є вибір стеку технологій, адже зазвичай саме він визначає доступні можливості, а також швидкодію роботи будь-якого програмного продукту. Тому варто приділити вивченню наявних і перспективних технологій достатню кількість часу, зробити дослідження, знайти і проаналізувати найпопулярніші варіанти і вибрати технології, які найкраще справлятимуться із поставленими функціональними і нефункціональними вимогами.

Важливі критерії при виборі стеку технологій:

- сумісність, яка визначає доступні технології і інструменти розробки;
- бюджет, відведений на розробку, інтеграція з платними сервісами;

- час, відведений на розробку, терміни здачі проекту;
- тип проекту, його розмір та складність;
- призначення проекту, потенційна аудиторія.

Прості проекти це зазвичай сайти-візитки, лендінг-сторінки, малі інтернет-магазини, прості програми. Для задач подібного плану в більшості випадків вже є готові рішення — шаблони або CMS. Середні проекти — це крупні інтернет-магазини і маркетплейси, портали національного масштабу, та інші різноманітні сервіси. Зазвичай використовуються фреймворки.

До складних проектів відносять величезні портали, соціальні мережі, пошукові системи, інноваційні та нетипові рішення. Ядро таких проектів зазвичай розробляється на чистій (нативній) мові програмування.

Також вартує звертати свою увагу на те, наскільки добре задокументований фреймворк, інструмент чи бібліотека. Адже детальна, повна документація дозволяє легко навчитися працювати з новими інструментами, знаходити найбільш підходящі методи розв’язування необхідних задачі чи виправляти помилки, з якими доводиться зустрічатися в процесі розробки;

- якщо для поставленої задачі вже існує готове рішення, то постає питання доцільності створення всього з нуля, адже можна використати вже розроблений алгоритм;
- якщо для розробки проекту використовуються платні сервіси, варто порахувати вартість підтримки такого ресурсу в довготривалій перспективі;
- вимоги до навантажень, тобто приблизна оцінка кількості відвідувань в день.

В залежності від навантаження необхідно вибрати відповідний сервер, щоб забезпечити постійну, стабільну роботу і запобігти перевантаження сервера:

- вимоги до безпеки: слід оцінити рівень важливості і конфіденційності особистих даних і вжити необхідних заходів для їхнього захисту;
- кросплатформеність: можливість використовувати розроблений продукт на всіх наявних типах популярних операційних систем, браузерів тощо;

- можливість інтеграції з іншими рішеннями: в загальному випадку це означає забезпечення можливості розвитку і вдосконалення розробленого проекту в майбутньому.

Вибираючи технологію за такими критеріями, можливо домогтися об'єктивності і зробити найкращий вибір з точки зору часових і фінансових витрат.

Для розробки даного програмного забезпечення було обрано наступний стек технологій:

Графічні редактори, інструменти для цифрового дизайну:

- Sketch (версія 55.1) — редактор векторної графіки, інструмент для графічного дизайну, виключно для операційної системи macOS;
- Principle (версія 5.8) — інструмент для створення інтерактивних прототипів та анімацій, також для операційної системи macOS;
- Zeplin (версія 2.0) — інструмент для співпраці UI/UX дизайнерів та Front-End розробників. В проекті використовується для утримання та швидкого доступу до всіх іконок, шрифтів, бібліотеки кольорів та отримання CSS-коду елементів, створених в Sketch.

Основні технології:

- Vue.js (версія 2.6.10) — це прогресивний фреймворк JS для створення користувацьких інтерфейсів;
- Vue Router (версія 3.0.3) — офіційна бібліотека маршрутизації для Vue.js;

Препроцесори:

- Stylus (версія 1.5.3) — препроцесор для CSS.

Бібліотеки:

- Lodash (версія 4.17.5) — для роботи з масивами і об’єктами.

Планувальник:

- Webpack (версія 4.34.0).

Інші інструменти:

- Babel — для переходу від ECMAScript (ES6/7/8) до ES5;

- ESlint — для обробки помилок.

JavaScript фреймворки (від англ. framework — каркас) — це готові рішення чи шаблони, що допомагають прискорити процес розробки [20]. Вибір правильного JavaScript фреймворку є першочерговим завданням, яке постає перед розробником, адже, на відміну від бібліотеки, фреймворк визначає архітектуру і початкові правила поведінки додатку [21].

За допомогою правильного JavaScript фреймворку можна створювати динамічні сайти, мобільні додатки та навіть повноцінні програми, використовуючи JavaScript, HTML та CSS. Також за допомогою фреймворків можна розробляти функціональні модулі, тобто різні онлайн-інструменти.

Для розробки свого проекту я вибрав веб фреймворк Vue.js, адже він, на відміну від монолітних фреймворків, виступає радше як бібліотека, тому що його можна впроваджувати поступово [21].

Розробив Vue.js Еван Ю (Evan You), колишній співробітник Google і Meteor Dev Group [22]. Почав розробляти фреймворк він в 2013, а в лютому 2014 відбувся перший публічний запуск [22]. Vue широко використовується серед китайських компаній, наприклад: Alibaba, Baidu, Xiaomi, Sina Weibo та ін. Він входить в ядро Laravel і PageKit. Нещодавно безкоштовна система управління репозиторіями GitLab теж перейшла на Vue.js [23].

В кінці вересня 2016 вийшла в реліз версія Vue.js 2.0 з упором на продуктивність — тепер використовується віртуальний DOM, підтримується серверний рендеринг, можливість використовувати JSX тощо. Хоча зараз він підтримується тільки співтовариством, він тримається гідно навіть на рівні продуктів таких гігантів, як Google і Facebook (Angular 1.7.8 і React 16.8.6), і поступово наздоганяє їх по популярності [24].

Не всі знають чому саме Vue, що він робить і в яких саме випадках його можна і потрібно використовувати. Технічно кажучи, Vue.js визначається як ViewModel шар шаблону MVVM. Він з'єднує модель і подання до двостороннього зв'язування даних.

Ядро Vue вирішує задачі рівня представлення (view), що робить простішою інтеграцію з іншими проектами та бібліотеками. З іншого боку, Vue можна застосовувати для створення односторінкових додатків, також відомих як SPA або Single-Page Applications.

Одним з ключових моментів в роботі Vue.js є віртуальний DOM. Структура веб-сторінки, як правило, описується за допомогою DOM (Document Object Model), яка представляє організацію елементів html на сторінці. Для взаємодії з DOM (додавання, зміни, видалення html-елементів) застосовується JavaScript. Але коли ми намагаємося маніпулювати html-елементами за допомогою JavaScript, то ми можемо зіткнутися зі зниженням швидкості роботи додатка, особливо при зміні великої кількості елементів. А операції над елементами можуть зайняти деякий час, що неминуче позначиться на користувацькому досвіді.

Для прискорення роботи і збільшення ефективності Vue.js використовує віртуальний DOM. Віртуальний DOM є легкою версією звичайного DOM. Якщо додатку потрібно дізнатися інформацію про стан елементів, то відбудеться звернення до віртуального DOM. Якщо дані, які використовуються в додатку Vue.js, змінюються, то зміни спочатку вносяться в віртуальний DOM. Потім Vue вибирає мінімальний набір компонентів, для яких треба виконати зміни на веб-сторінці, щоб реальний DOM відповідав віртуальному. Завдяки віртуальному DOM підвищується продуктивність роботи.

Vue.js дуже гнучкий і менш прямолінійний в використанні. Це дозволяє структурувати додаток відповідно до вимог. Також фреймворк підтримується всіма браузерами, які сумісні з ECMAScript 5. На даний момент це все сучасні браузери, в тому числі IE11 [25].

Приклад оголошення і використання класу в Vuejs:

```
v-app(dark)
    div.beta.blue.ps-1 Beta
    v-toolbar(app)
      router-link(to="/").title
        span Schedan
        sup.pa-1 {{ currentTime }}
      v-spacer
```

```
      div(v-if="user")
        span Привіт,
          b  {{ user.name }}
        v-btn(v-if="user.admin" to="/admin" icon)
          v-icon edit
        v-btn(@click="logout" icon)
          v-icon exit_to_app
      div(v-else)
        v-btn(to="login") Вхід
        v-btn(to="registration") Реєстрація
    loading(v-if="loading")
    v-content
      transition(name="move" mode="out-in")
        router-view
```

Vue.js є інтерпретатором без суворої типізації, і може виконуватися в різних середовищах, кожне зі своїми власними особливостями сумісності, тому необхідно проводити перевірку, чи код працює у відповідності до очікувань в переліку більшості можливих конфігурацій.

Компанія також надає клієнтські бібліотеки, які дозволяють інтеграцію із додатками для Android, iOS, JavaScript/Node.js, Java, Objective-C та Swift. База даних також доступна через REST API та прив'язки до декількох сценаріїв JavaScript, таких як AngularJS, React, Ember.js та Backbone.js.

Наступним кроком в плануванні розробки є вибір препроцесорів HTML і CSS. Основне завдання препроцесора — це збільшення читабельності для розробника за допомогою надання більш “людських” синтаксичних конструкцій, щоб зробити проект простішим для розуміння, а також прискорити розробку та підтримку проектів [26].

Написаний за допомогою препроцесорів код має низку переваг і націлений на:

- зрозумілість для людини, тобто здатність швидко прочитати та зрозуміти код неозброєним оком;
- логічність, тобто закономірність та послідовність у відповідності до здорового сенсу;
- ефективність, тобто швидкість виконання.

Для даного проекту було обрано препроцесор Stylus для CSS.

CSS препроцесор — це надбудова для CSS, яка додає нові можливості для мови CSS, додає нові синтаксичні конструкції та змінює старі.

Stylus — це препроцесор CSS, основним завданням якого є розширення можливостей написання CSS коду, прискорює написання коду і в рази полегшує його обслуговування [27].

Препроцесор Stylus підтримує вкладені правила, які роблять CSS код більш читабельним, логічним і природним, також підтримує вкладені властивості, шаблонні селектори [27].

У Stylus можна задати порядок дії при операціях круглими дужками. Також препроцесор має набір вбудованих функцій, яким можна передавати параметри. Змінні можна використовувати при найменуванні селекторів і CSS властивостей.

Фрагмент коду написаного на Stylus:

```
    table.table tbody td:first-child, table.table tbody
td:not(:first-child), table.table tbody th:first-child, table.table
tbody th:not(:first-child), table.table thead td:first-child,
table.table thead td:not(:first-child), table.table thead th:first-
child, table.table thead th:not(:first-child) {
      border-right: 1px solid hsla(0,0%,100%,.12);
      border-left: 1px solid hsla(0,0%,100%,.12);
    }
    table.table thead {
      border-top: 1px solid hsla(0,0%,100%,.12);
    }
    .scale-enter-active, .scale-leave-active {
      transition: opacity .35s ease, transform .35s ease;
    }
    .scale-enter, .scale-leave-active {
      opacity: 0;
      transform: scale(0.95);
    }
```

Одним з найважливіших аспектів побудови ефективного і швидкісного сайту є оптимізація і автоматизація. Для цього існує безліч рішень і одним з них є Webpack.

Webpack це інструмент збірки веб-додатків, що написані на JavaScript, з відкритим вихідним кодом. Хоча він і призначений в першу чергу для JavaScript, але може трансформувати й інші інтерфейсні ресурси, такі як HTML, CSS і

зображення, якщо встановлені відповідні плагіни. Webpack приймає модулі з залежностями і генерує статичні засоби, що представляють ці модулі [28, 29].

Webpack приймає залежності і генерує графік залежностей, що дозволяє веб-розробникам використовувати модульний підхід для розробки своїх веб-додатків. Він може бути використаний з командного рядка або може бути налаштований за допомогою конфігураційного файлу, який називається webpack.config.js.

Встановлення Webpack:

```
npm install -g webpack
```

Ініціалізація Webpack:

```
export default function(webpack, plugins, args, config,
taskTarget, browserSync) {
  let dirs = config.directories;
```

Щоб працювати з Webpack необхідно створювати завдання, що відповідають за кожну з функцій. За допомогою завдання “watch” можна відстежувати розробку і зміну файлів на сервері:

```
webpack.task('watch', () => {
    if (!args.production) {
      webpack.watch([
        path.join(dirs.source, dirs.styles,
'**/*.{scss,sass}'),
        path.join(dirs.source, dirs.modules,
'**/*.{scss,sass}')
      ], ['sass']);    }  });}
```

## 2.2 Розробка мапи сайту, функціональні можливості користувачів, діаграми user flow та use case

Планування будь-якого сайту починається з аналізу та встановлення всіх типів користувачів (наприклад, гість, зареєстрований користувач, адміністратор тощо) та функціональних можливостей, які будуть у тих чи інших користувачів.

Для того, щоб найбільш правильно описати всі типи користувачів і їх функціональні можливості, використовується діаграма use case. Найпростіша діаграма use case являє собою уявлення про взаємодію користувача з системою,

яка показує взаємозв'язок між користувачем і різними випадками використання, в яких задіяний користувач. Діаграма use case може ідентифікувати різні типи користувачів системи та різні випадки використання, а також часто супроводжуватиметься іншими типами діаграм. Випадки використання представлені або колами, або еліпсами [30].

Оскільки розроблюваний веб-ресурс призначений для особистого використання, а роботи на ньому буду викладати тільки я, на сайті буде відсутня реєстрація користувачів та взагалі будь-які типи користувачів, окрім гостей.

На сайті також відсутня панель адміністратора та будь-які системи управління контентом. Нижче буде наведена низка причин, чому я відмовився від подібних технологій.

Перша і найголовніша причина полягає в тому, що для кожного дизайн-проекту створюється своя окрема сторінка, адже, на відміну від традиційних портфоліо, де інформація подається у вигляді статичних або анімованих зображень, на власному сайті можна зробити набагато більш інтерактивні презентації з використанням JavaScript, тому наповнення таких сторінок виходить занадто неоднорідним, щоб створити якийсь єдиний шаблон чи рішення.

Друга причина полягає в тому, що наповнення сторінки зазвичай міняють дуже рідко, адже сторінку створюють вже тоді, коли дизайн остаточно завершено і презентація повністю готова.

Третя причина полягає в тому, що для даного рішення немає ситуацій, коли треба щось динамічно генерувати на сервері, адже сайт цілком і повністю підпорядковується одній людині, і, як було сказано вище, через специфіку створення презентацій проектів кожна сторінка створюється вручну.

Четверта і остання причина полягає в тому, що статичні сайти набагато стійкіші до навантажень, що дозволяє використовувати хостинг з безкоштовним тарифом і при цьому його буде вистачати для повноцінного функціонування сайту [31].

Крім того, враховуючи причини, зазначені вище, на сайті відпадає потреба в базі даних.

Саме тому діаграма use case сайту буде включати в себе лише користувачів-гостей.

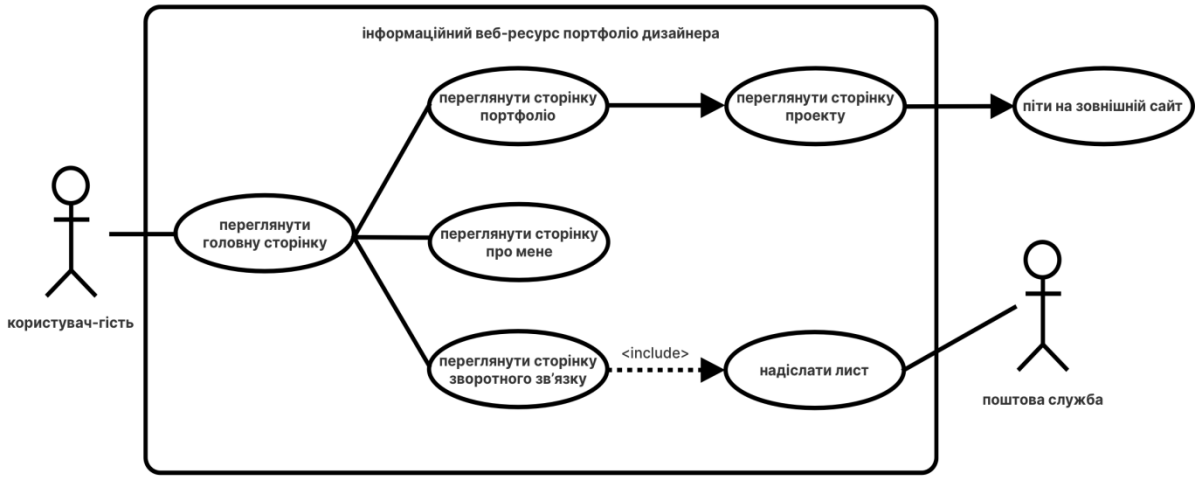


Рисунок 2.5 — Use case діаграма інформаційного ресурсу портфоліо дизайнера

Після того як діаграма use case готова, можна приступати до створення мапи сайту, яка включає в себе всі сторінки сайту та їх зв’язки, тобто всі можливі переходи.

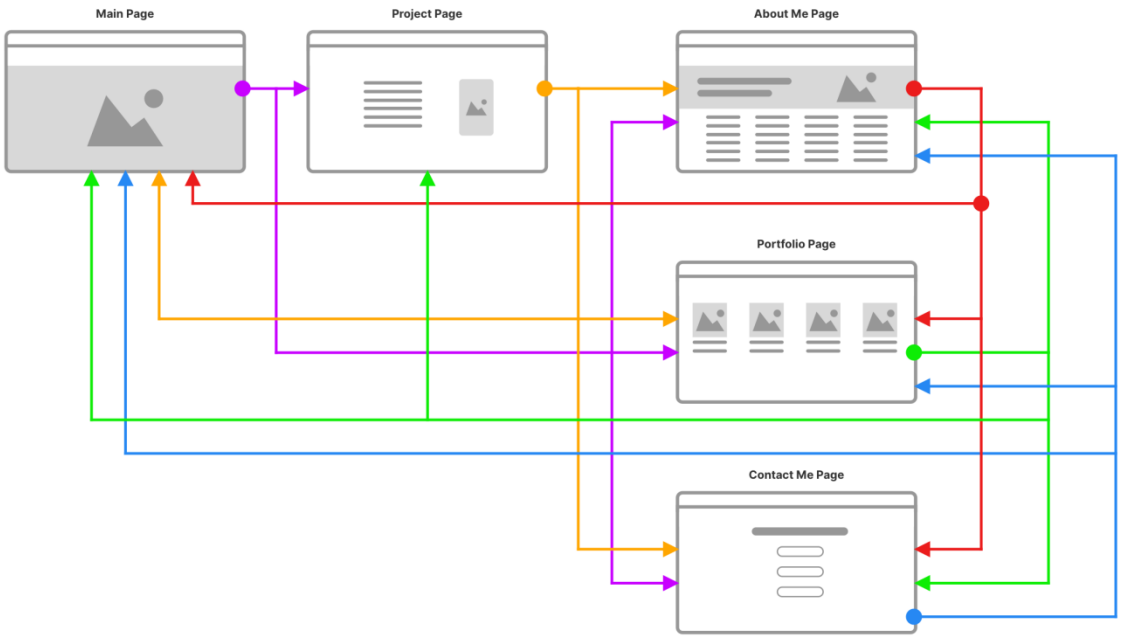


Рисунок 2.6 — Мапа інформаційного ресурсу портфоліо дизайнера

Важливим етапом створення мапи будь-якого сайту є головна сторінка, адже саме вона зазвичай формує перше враження від ресурсу та в цілому направляє користувача. Отже, на головній сторінці буде вітрина з новими роботами та кнопки з закликом до дії.

Як стає зрозуміло з рисунку наведеного нижче, майже всі сторінки сайту взаємопов'язані, адже на сайті використовується універсальна панель навігації, що містить в собі посилання на головну сторінку, інформацію про дизайнера, портфоліо а також сторінку зворотнього зв'язку.

Виключенням є тільки сторінки конкретних проектів, адже перейти на них можна тільки зі сторінки Портфоліо або з головної (на мапі сайту сторінки всіх проектів представлені як одна).

Тому для того, щоб краще направити користувача, створюють діаграму user flow. User flow (шлях, маршрут користувача) — це спосіб зображення переміщення користувача по сайту або його взаємодії з ним. Зазвичай інтернет-маркетологи стараються побудувати user flow таким чином, щоб якомога легше і швидше донести до користувачів потрібну інформацію, яка в свою чергу сприяє виконанню користувачем цільових дій. Тому для свого проекту я розробив три можливі сценарії user flow:

- зеленим позначено перший і найкращий сценарій. З головної сторінки потенційний клієнт переходить на сторінку проекту, йому все подобається і він переходить на сторінку з формою зворотнього зв'язку для того, щоб надіслати деталі свого замовлення;
- помаранчевим зображено альтернативний розвиток першого сценарію, при якому користувач переглянув сторінку проекту, після чого у нього ще залишилися додаткові питання, тому він перейшов на сторінку "Про мене" або/та "Портфоліо", після чого, як і в першому сценарії, перейшов на сторінку зворотнього зв'язку;
- і останній, найгірший варіант, що не зображено на діаграмі. Користувач рухається по сайту як в першому чи другому варіанті, але йому

нічого не подобається і він покидає сайт, замість того, щоб перейти до форми зворотнього зв’язку.

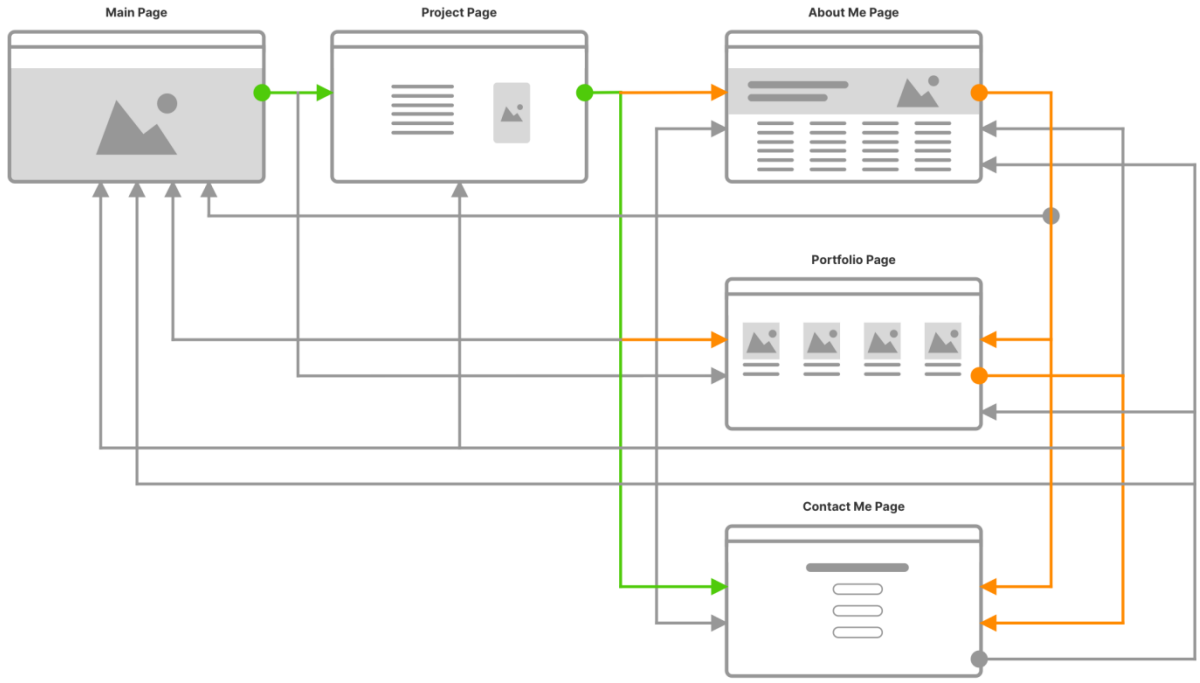


Рисунок 2.7 — Діаграма user flow, що включає в себе різні варіанти маршруту користувача

Звичайно, кожен користувач унікальний і у кожного може бути своя подорож по сайту, яка відрізняється від усіх вищеназваних сценаріїв, але в загальному сайт проектується для того, щоб направляти користувачів так, як зазначено вище.

Після того, як складено приблизну мапу сайту та продумано користувацькі маршрути, можна приступати до розробки дизайну візуального інтерфейсу. Загальноприйнято, що перший етап робить UX engineer або маркетолог, а зовнішній вигляд – UI designer, але останнім часом ці спеціальності перестали розділяти і називають одним універсальним словом — UX/UI designer [32].

# 3 РОЗРОБКА ІНФОРМАЦІЙНОГО ВЕБ-РЕСУРСУ ПОРТФОЛІО ДИЗАЙНЕРА ЗАСОБАМИ SKETCH, PRINCIPLE ТА JAVASCRIPT

## 3.1 Розробка інтерфейсу та анімацій в портфоліо дизайнера, реалізація структури сайту

Перш за все, при створенні будь якого інтерфейсу, треба визначитись із загальною естетикою. Сюди можна включити кольори, шрифти, розмір та форму кнопок, зовнішній вигляд іконок. Як відомо, дизайн може мати багато форм та викликати багато емоцій. Він може бути веселим і кумедним, а може бути темним і гнітючим. Може бути сучасним і трендовим, а може бути більш класичним, з використанням старомодних шрифтів із засічками. В питанні вибору мови дизайну треба бути дуже обережним, адже дизайн в стилі мережі “Макдональдс” навряд чи підійде для компанії, яка виробляє дорогі швейцарські годинники. Крім того, є багато випадків, коли ті чи інші дизайнерські рішення не витримують перевірки часом, а тому потребують редизайну. Звичайно, будь-який дизайн потрібно створювати в контексті своєї епохи.

Якщо звернутися до історії, можна побачити, що найперші персональні комп’ютери були або чорно-білими, або чорно-зеленими. Вже через кілька років з’явилися кольорові дисплеї, але їх роздільна здатність була доволі невисокою, тому перші дизайнери старалися звертатися до абстрактних, символічних знаків, що супроводжувалися текстовим поясненням. Приблизно з середини 90-х став проявлятися тренд на ускладнення елементів інтерфейсу – комп’ютери того часу отримали новітні кольорові стандарти і розробники, звичайно, хотіли цим скористатися, щоб відрізнитися від конкурентів (рис. 3.1). Замість плоских елементів на заміну прийшли об’ємні, кольорові іконки та кнопки з ефектами засвітів, тіні та напівпрозорості. Дедалі частіше розробники та дизайнери стали звертатися до об’єктів реального світу та зображувати їх у світі цифровому [33].

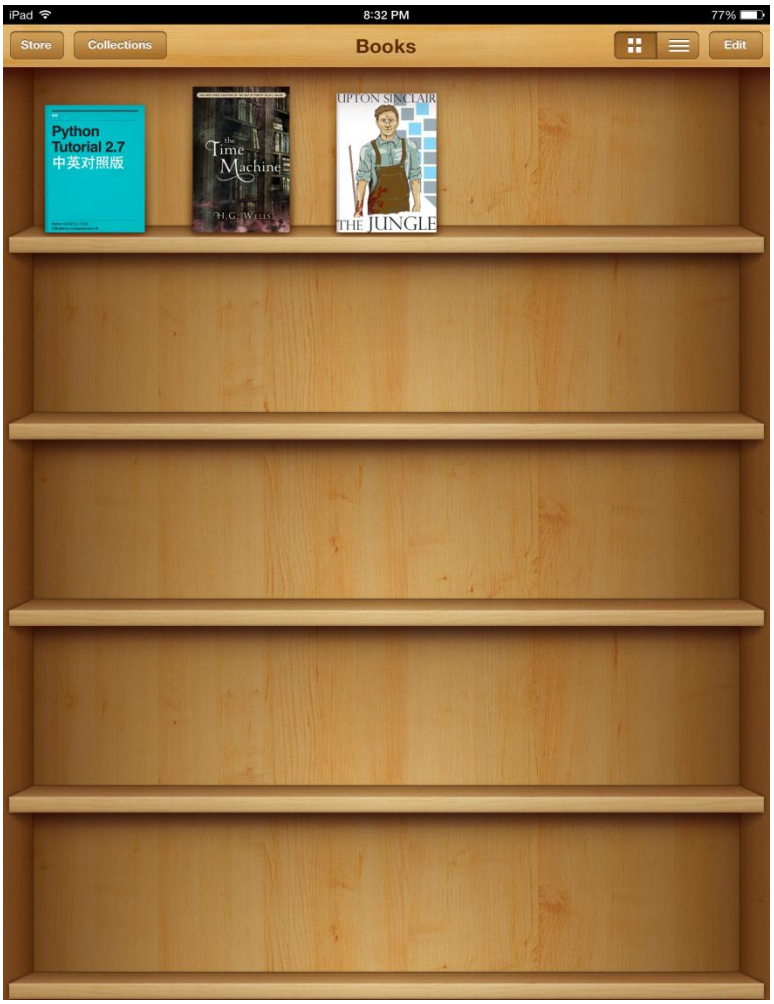


Рисунок 3.1 – Системний додаток iBooks для операційної системи Apple iOS 6, 2012 рік

Та очевидно, що у такого підходу було дуже багато мінусів, серед яких можна виділити головні:

- користувачеві важко сконцентруватися на досягненні своїх цілей;
- складність масштабування на різні роздільні здатності, через використання текстурованих поверхонь;
- використання великої кількості ефектів, таких як бліки та тіні, забирає багато дорогоцінного місця на екрані;
- складність розробки, підбору матеріалів для дизайну;
- оптимізація, швидкість виконання, використання ресурсів батареї телефону, адже рендеринг високоякісних зображень може створювати сильне навантаження на процесор пристрою.

Тому не дивно, що починаючи з 2012-2013 років лідери індустрії стали знову повертатися до плоского дизайну [34]. Такий глобальний тренд відбувається не тільки в UX/UI дизайні, а майже всюди: в брендингу, в архітектурі, в автомобільному дизайні, в промисловому дизайні тощо.

Оскільки у сьогоднішнього підходу до дизайну є багато беззаперечних переваг, стає очевидно, чому обрано саме його при створенні дизайну сайту. Сайт має бути створеним у сучасній, плоскій манері з використанням малої кількості кольорів та візуальних ефектів.

Отже, визначившись із графічним стилем, треба обрати кольорову гаму сайту. Станом на сьогодні дуже поширені яскраві градієнти з різноманітними кольорами, що переливаються.

Але використання таких градієнтів набуло надмірного масштабу, деколи породжуючи комічні ситуації. Крім того, яскрава символіка буде відволікати користувачів від ознайомлення з матеріалами сайту.


Рисунок 3.2 – Головна сторінка інформаційного
веб-ресурсу портфоліо дизайнера

Майже з самого початку прийнято рішення створювати сайт в традиційних кольорах для друку: білому та чорному. Такі кольори дозволяють абстрагуватися

від всього непотрібного та сконцентруватися на головному, а також роблять сайт візуально більшим. Крім того, такі кольори гарно відображаються на всіх типах дисплеїв завдяки своїй контрастності. Відомо, що багато брендів сьогодні переключаються на елегантну чорно-білу схему.

Головна сторінка інформаційного веб-ресурсу портфоліо дизайнера представлена на рисунку 3.2.

Як видно з рисунку 3.2, головна сторінка є вітриною останніх завантажених робіт, і хоча з неї можна перейти на інші сторінки, вона спроектована так, щоб привертати увагу користувача до проектів. Цей ефект досягається завдяки великій назві проекту, посиланням з червоною стрілкою та прихованій навігації (рис. 3.3).

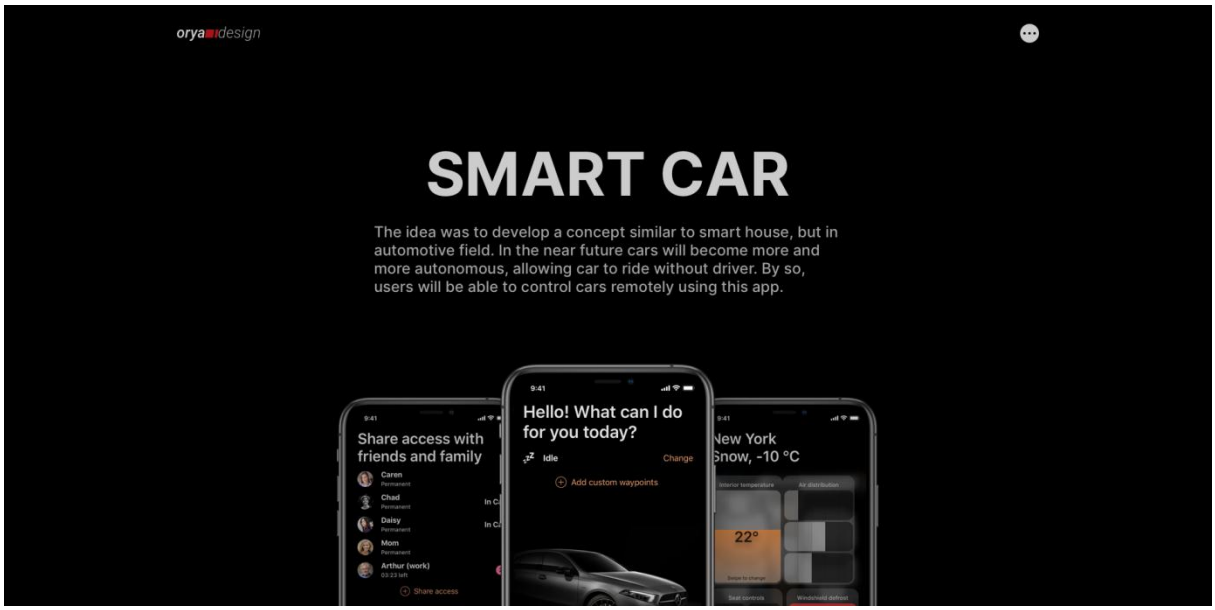


Рисунок 3.3 – Сторінка проекту “Smart car”: заголовок і короткий опис

В деяких місцях основні кольори міняються місцями – наприклад, на сторінках деяких проектів фон сторінки стає чорним, а текст – білим. Чорний фон глибокий, він, як і білий, в великих кількостях візуально збільшує простір сайту та гарно поєднується з будь-яким вмістом сторінки.

Крім того, вміст на сторінках проектів настільки неоднорідний, що в залежності від контексту фон може набувати й інших кольорів.

На всіх інших сторінках, де вміст однорідний, фон або білий, або світло-сірий, щоб виділити об’єкти на передньому плані (рис. 3.4).

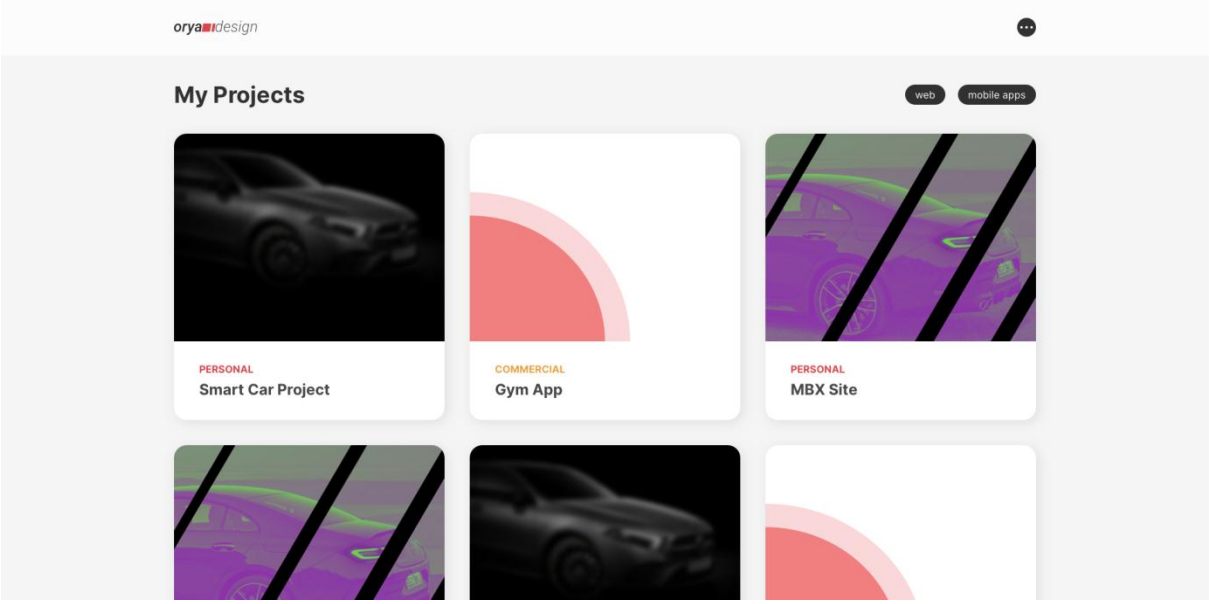


Рисунок 3.4 – Сторінка “My projects”: використання світло-сірого фону для створення ефекту глибини

Наступною важливою задачею було створення форми зворотнього зв’язку. Після створення декількох прототипів я зупинився на таких полях:

– ім’я користувача;

– email користувача;

– опис проекту.

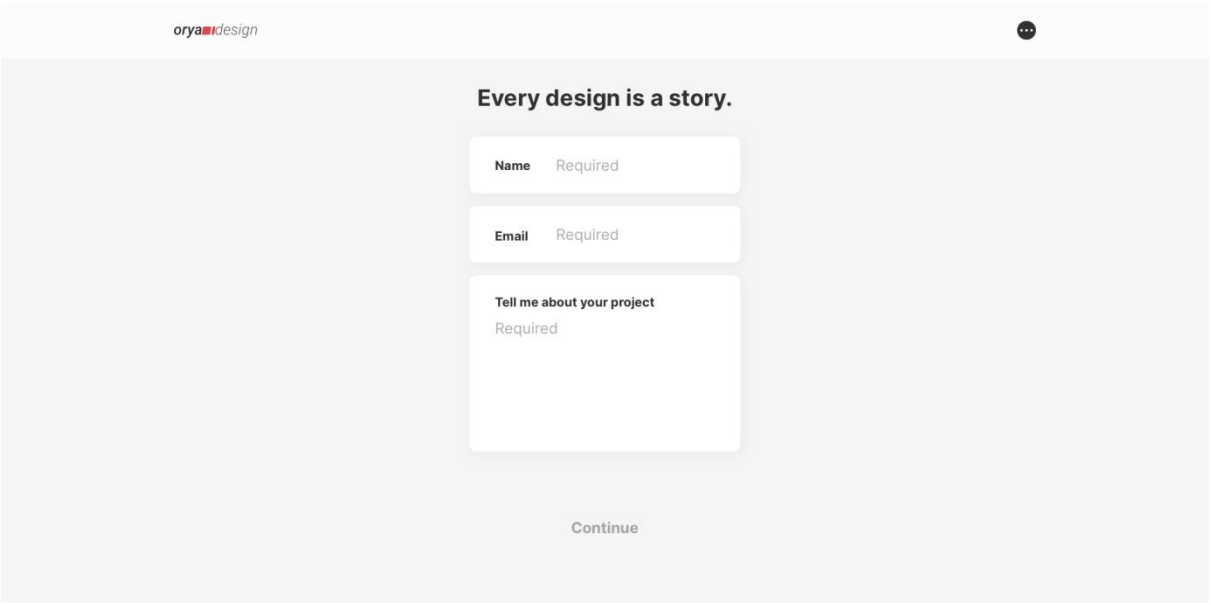


Рисунок 3.5 – Форма зворотнього зв’язку на сторінці “Contact me”

В результаті ми отримуємо ім’я людини, до якої звертаємось, поштову скриньку, на яку треба надіслати відповідь та невеликий опис бажаного проекту, що дозволить отримати контекст вже з першого повідомлення. Крім того, така форма економить час, адже користувач не перевантажений зайвою інформацією, яку зазвичай треба вводити на інших сайтах (рис. 3.5).

На всіх сторінках є приховане меню навігації, з якого можна перейти на такі сторінки:

– сторінка “Про мене”;

– сторінка “Портфоліо”;

– сторінка “Зворотній зв’язок”.

Крім того, з будь-якої сторінки можна перейти на головну, натиснувши на логотип.


Рисунок 3.6 – Меню навігації в прихованому вигляді


Рисунок 3.7 – Меню навігації в повному вигляді

Оскільки на сайті дуже мало розділів, то було застосоване горизонтальне меню навігації, що розкривається і закривається. Таким чином сайт набуває ще чистішого і простішого вигляду коли меню не потрібне, а коли воно потрібне – його завжди можна знайти, натиснувши на кнопку із трьома крапками (рис. 3.6, рис. 3.7).

Також варто зазначити, що на сайті відсутні “хлібні крихти”.

“Хлібні крихти” – це допоміжний елемент навігації, який допомагає користувачеві відстежувати своє місце розташування в програмі, на сайті, або серед документів. Зазвичай вони потрібні для того, щоб користувач міг покроково повертатися назад в складних продуктах [35]. Наприклад, “хлібні крихти” можуть набувати наступного вигляду (рис. 3.8):

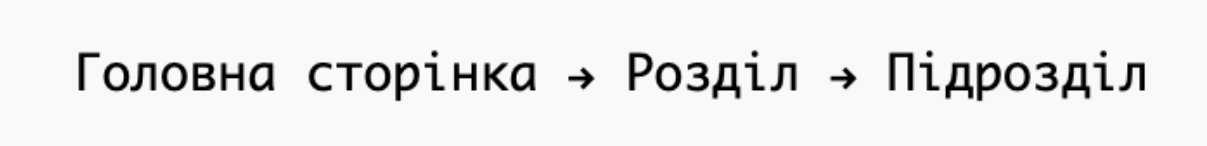


Рисунок 3.8 – “Хлібні крихти” на сайті Wikipedia

Але повертаючись до мапи сайту стає зрозуміло, що майже з будь-якої сторінки можна перейти будь-куди, використовуючи меню навігації, крім того, тільки одна сторінка виступає батьківською по відношенню до іншої. Це сторінка Portfolio, з якої можна перейти на сторінки окремих проектів. В зв’язку із цим на пізніх етапах дизайну прийнято рішення про відмову від “хлібних крихт”, адже на зручність навігації це ніяк не вплине та дасть змогу зробити сайт ще радикальнішим в плані простоти використання.

Іншим важливим етапом створення будь-якого хорошого користувацького інтерфейсу є анімація. Насправді, мало хто задумується над дизайном анімацій та тим, як вони можуть покращити користувацький досвід. Зазвичай програми вважають серією екранів або станів, а анімацію – як спосіб подорожувати між цими станами. Більшість дизайнерів і розробників згадують про анімації під кінець розробки проекту, та вибирають будь-які підходящі анімації, але такий підхід не дуже правильний.

Звичайно, коли анімації красиві, відповідають загальному графічному стилю, задають тон та швидкість виконання будь-яких функцій це добре, але треба дотримуватися головного правила: анімації мають мати сенс у контексті виконуваної дії.

Через анімації можна допомогти користувачеві та донести трішки більше інформації про те, що відбувається на екрані, як додаток реагує на введену інформацію. Тому при створенні дизайну сайту з ранніх етапів проектувалися всі кнопки та іконки таким чином, щоб їх можна було пов’язати з інформативними анімаціями.

Розрізняють два головні типи анімацій [36]:

– State-driven animation (рис. 3.9);

– Action-driven animation (рис. 3.10).

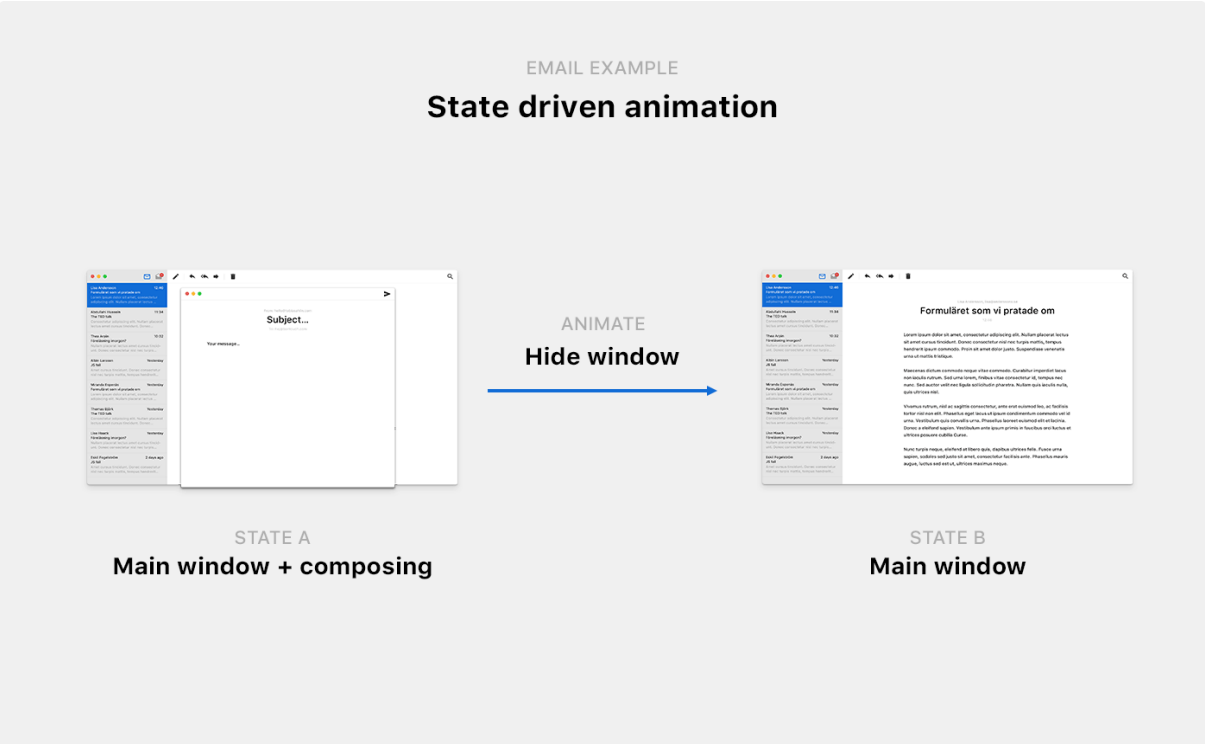


Рисунок 3.9 – Принцип роботи state driven animation
на прикладі поштового додатку

Перший підхід заснований на тому, що у елемента інтерфейсу є два стани, а введення даних користувача перемикає їх.

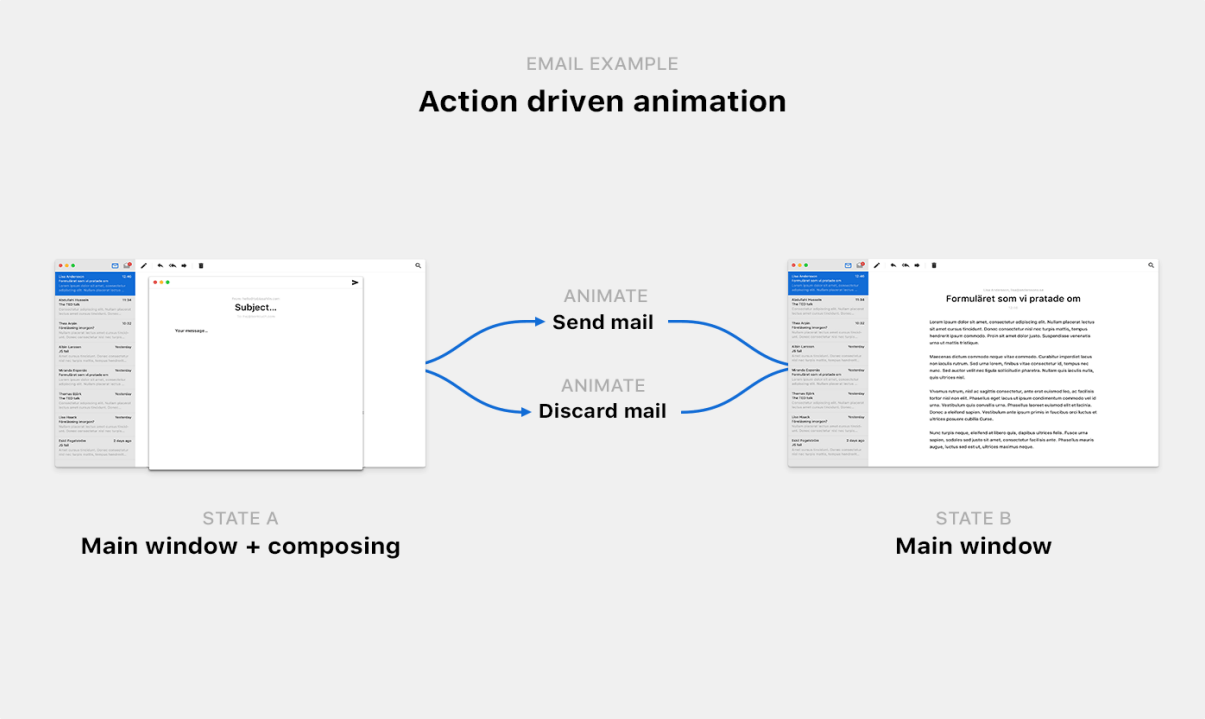


Рисунок 3.10 – Принцип роботи action driven animation
на прикладі поштового додатку

У другому випадку у нас не дві, а три анімації, які спрацьовують в залежності від дії, яку вибирає користувач. Це дозволяє створити візуальну різницю між різними діями та дати користувачеві контекст. Вибравши кнопку Cancel ми чітко показуємо, що модальне вікно скасовується за допомогою анімації зменшення і зникнення. Коли вибрано позитивну дію (робити щось), ми робимо навпаки: ініціюємо анімацію збільшення і зникнення, наближаючи модальне вікно до користувача. Іншими словами, ми не тільки розрізняємо стани, але й як користувач подорожує між цими станами, тобто які дії виконуються.

Заради справедливості варто сказати, що action driven animation не є моделлю, яка стверджує, що вона абсолютно точно пояснює або доводить, як потрібно анімувати програму. Це спрощений спосіб подумати про рух, який можна застосувати для перегляду анімації інтерфейсу програми або продукту, щоб створити більш осмислені анімації.

Створення анімацій, не дивлячись на те, що мало хто надає їм потрібного значення, є доволі трудомістким процесом. Є дуже багато властивостей, які можна змінити в будь-якій сутності: колір, прозорість, товщина контурів, розмір, розміщення на осях координат X Y Z, внутрішня та зовнішня тінь тощо. Є багато типів анімацій, які можна застосувати при зміні тих чи інших властивостей. Крім того, не варто забувати, що всі анімації мають свій окремий вимір – час.

Занадто довгі анімації можуть призвести до відчуття втоми, так, ніби система завантажується та працює повільно, а занадто швидкі анімації можуть призвести до того, що користувач не прослідкує за нею, а отже не помітить, що саме, де і як сталось. Крім того рух виглядатиме неправдоподібно. Як сказано в Human Interface Guidelines (мова дизайну Apple), анімації треба використовувати з розумом, адже надмірне використання анімації може призвести до відволікання користувачів, а крім того, може призвести до дезорієнтації [37].

Крім того, рух в анімації в тій чи іншій мірі має відповідати законам фізики, тобто предмети, що рухаються, мають прискорюватися та сповільнюватися як реальні об’єкти [37, 38, 39].

Тому для того, щоб пов’язати два стани в меню навігації було створено спеціальні анімації для об’єктів, які приймають участь у переході. Щоб показати, як пов’язані перший і другий стан (рисунки 3.6 і 3.7), створено анімацію руху пунктів меню з правої сторони вліво, від кнопки меню до свого місця. Якщо натиснути на хрестик, то пункти меню сховаються назад, а якщо обрати будь-який із пунктів меню, то відбудеться ще одна анімація руху вліво, а також зникнення за допомогою збільшення прозорості. Крім того, кнопка меню відіграє спеціальну анімацію перетворення на хрестик за допомогою оптичного обману. Ефект перетворення досягається завдяки тому, що одночасно один елемент зникає, а інший з’являється. Крім того, для збільшення зв’язку двох елементів (три крапки розміщені горизонтально, а хрестик під кутом 45 градусів) елементи одночасно змінюють кут нахилу, що показує користувачу походження і взаємозв’язок двох елементів.

### 3.2 Розробка механізмів роботи функціональних можливостей сайту

Для зручного зберігання коду та хостингу сайту було обрано сервіс GitHub.com. Оскільки сайт заснований на розподіленій системі контролю версій Git, з його допомогою можна зручно переглядати і повертатися до будь-яких версій проекту, а також об’єднуватись з іншими програмістами та розробляти проекти разом.

Весь основний код програми знаходиться в папці src (рис.3.11). Там ж знаходяться наступні елементи:

– папка assets – в ній знаходяться всі зображення;

– папка components – папка, в якій знаходяться всі компоненти. Компонент – це сутність, яка складається з декількох елементів (наприклад, хедер);

– папка pages – це папка для сторінок. Фактично, сторінка це те ж, що і компоненти, тільки сторінки під’єднані до Vue Router, який їх перемикає.

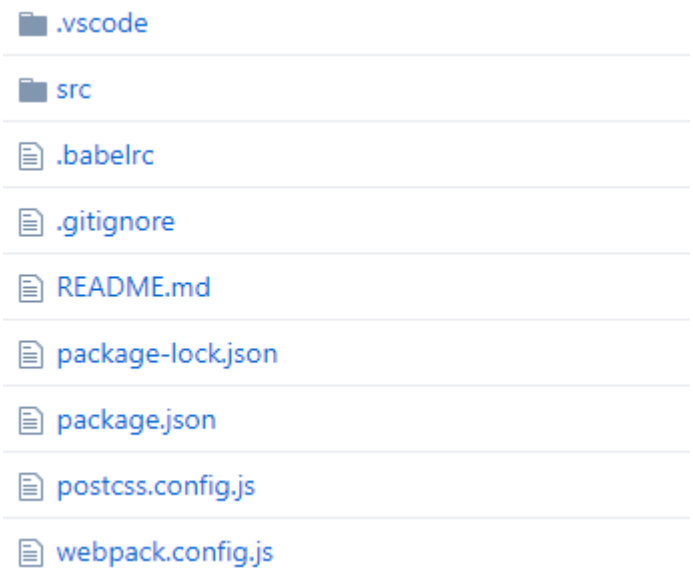


Рисунок 3.11 – Структура виконаного проекту на сайті Github.com

Змінні для стилів елементів, що були створені за допомогою Zeplin, зберігаються в файлі variables.styl.

```
// PALETTE

background-white = #ffffff
text-on-white = rgba(0, 0, 0, 0.8)
header-on-black = rgba(0, 0, 0, 0.72)
hovers-on-white = rgba(0, 0, 0, 0.64)
focus-on-white = rgba(0, 0, 0, 0.08)
footers-on-white = rgba(0, 0, 0, 0.04)

bacgkround-black = #000000
text-on-black = rgba(255, 255, 255, 0.8)
header-on-white = rgba(255, 255, 255, 0.72)
hovers-on-black = rgba(255, 255, 255, 0.64)
focus-on-black = rgba(255, 255, 255, 0.08)
footers-on-black = rgba(255, 255, 255, 0.04)
commercial-title = rgba(232, 137, 19, 0.8)
personal-title = rgb(208, 26, 33, 0.8)

base-shadow = 4px 4px 20px 0 footers-on-white

list-reset()
  list-style none
  padding 0
  margin 0

xs = 480px
sm = 640px
md = 768px
lg = 1024px
```

```
xl = 1280px
full-hd = 1920

header-height = 90px
max-container-width = 1490px
```

В цьому ж файлі використовуються брейкпоінти і деякі інші важливі значення.

Відправною точкою всього веб-додатку є файл App, що знаходиться в папці src. В цьому файлі знаходяться всі кореневі компоненти і роутер, а також деякі локальні і глобальні стилі:

```
<template>
  <div class='main'>
    <Header v-bind:onBlack="isDarkTheme" />
    <router-view @setDarkTheme="setDarkTheme"></router-view>
    <Footer v-bind:onBlack="isDarkTheme" />
  </div>
</template>

<script>
import Header from 'components/Header'
import Footer from 'components/Footer'

export default {
  data() {
    return {
      isDarkTheme: false
    }
  },
  components: {
    Header,
    Footer
  },
  computed: {
    MainContainer() {
      return this.$route
    }
  },
  methods: {
    setDarkTheme(value) {
      this.isDarkTheme = value
    }
  }
}
</script>

<style lang='stylus' scoped>
.main
```

```
  display flex
  flex-direction column
  min-height 100vh
</style>

<style lang='stylus'>
  @import '~@/variables'
  html
    font-size 10px

  body
    font-family 'Inter', Arial, sans-serif
    overflow-x hidden

  .page
    margin-top 90px
    box-sizing border-box
    min-height 100vh
    flex-grow 1
</style>
```

Один з найменших, але найчастіше використовуваних компонентів, це ImportedImage. Він знаходиться в папці components. За допомогою нанотехнологій (з 16 по 24 рядок) він імпортує в себе всі зображення, що знаходяться в папці assets. Зображення при цьому обробляються за допомогою Webpack: він завантажує їх в корінь сайту, змінює назву на хеш і повертає замість картинки посилання на неї. Посилання в цьому компоненті записуються в об’єкт. Після чого, шляхом використання компонента ImportedImage, з використанням назви зображення, можна легко вставити будь-яке зображення з папки assets:

```
<template>
  <img
    v-bind:role='injectable ? "image" : ""'
    v-bind:src='imageSource'
    v-bind:alt='altAttribute'
    v-bind:aria-label='injectable ? altAttribute : ""'
    v-bind:aria-hidden='!altAttribute'
    style='fill: currentColor'
    ref='image'
  >
</template>

<script>
import SVGInject from '@iconfu/svg-inject'
```

```
    const importImages = require => require
      .keys()
      .reduce((images, key, index, array) => {
        const imagesKey = key.replace('./',
'').replace(/\.(svg|png|jpg|jpeg)/, '')
        images[imagesKey] = require(key)
        return images
      }, {})

    const images = importImages(require.context('../assets/',
true, /\.(svg|png|jpg|jpeg)$/))
    console.log(images)

    export default {
      props: {
        name: String,
        altAttribute: String,
        injectable: Boolean
      },
      data() {
        return {
          imageSource: images[this.name]
        }
      },
      mounted() {
        this.injectable && SVGInject(this.$refs.image)
      }
    }
    </script>
```

Крім того, у ImportedImage є прапорець Injectable. Якщо надати йому значення True, то замість звичайного тега <img> компонент буде відмальовувати inline-SVG. Це зроблено для зручності, адже потім такі компоненти можна анімувати за допомогою SVG-анімації.

Він семантично вставляє зображення:

– якщо вставлено inline-SVG, йому буде додана роль “image”, адже по замовчуванню електронні книги і роботи бачать картинку як graph;

– якщо знову-таки вставили inline-SVG, то йому буде добавлено атрибут aria-label, який відіграє роль alt-атрибута в таких випадках;

– якщо не передано в компонент пояснення, що знаходиться на зображенні (alt-атрибут), значення атрибута зміниться на aria-hidden=“true”, щоб електронні книги і роботи не спіткнулись на незрозуміле зображення.

Компонент зображення:

```
    <template>
      <router-link to="/home" class="link">
        <ImportedImage class='logo-full' v-show="!onBlack"
:injectable="false" name="logo-on-white" altAttribute="Orya Design"
/>
        <ImportedImage class='logo-full' v-show="onBlack"
:injectable="false" name="logo-on-black" altAttribute="Orya Design"
/>
        <ImportedImage class='logo-mini' :injectable="false"
name="logo-mini" altAttribute="Orya Design" />
      </router-link>
    </template>

    <style lang="stylus" scoped>
    @import '~@/variables'

    .link
      display block

    .logo-mini
      display none
      height 17px
      width 28px

    @media (max-width md)
      .logo-mini
        display block
      .logo-full
        display none
    </style>

    <script>
    import ImportedImage from 'components/ImportedImage'

    export default {
      props: {
        onBlack: Boolean
      },
      components: {
        ImportedImage
      }
    }
    </script>
```

Цей компонент рендерить різні версії логотипу в залежності від того, мобільна це версія чи десктопна. Також в залежності від кольорової гами можуть змінюватися кольори логотипу.

Оскільки компонент кнопки More (кругла кнопка з трьома крапками) має складну анімацію, він виконаний за допомогою inline-SVG, а анімований за

допомогою CSS-властивості transition. По наведенні SVG міняє колір компонента з чорного на білий. (див. код, починаючи з hover):

```
<template>
  <svg
    v-on:mouseover="setOnHover(true)"
    v-on:mouseout="setOnHover(false)"
    role="button"
    aria-label="more"
    tabindex="0"
    class="button"
    v-bind:class="{ 'on-black': onBlack }"
    v-bind:viewBox="viewBox"
    ref="mainElement"
    v-on:keypress="emulateClick"
    xmlns="http://www.w3.org/2000/svg"
  >
    <path class="path" fill-rule="evenodd" v-bind:d="path" />
  </svg>
</template>

<script>
const pathRegular = '(тут був код SVG елемента)'
const viewBoxRegular = '0 0 30 30'
const pathHover = '(тут був код SVG елемента)'
const viewBoxHover = '0 0 34 34'

export default {
  data() {
    return {
      path: pathRegular,
      viewBox: viewBoxRegular
    }
  },
  props: {
    onBlack: Boolean
  },
  methods: {
    setOnHover(isHovered) {
      this.path = isHovered && !this.onBlack ? pathHover :
pathRegular
      this.viewBox = isHovered && !this.onBlack ? viewBoxHover
: viewBoxRegular
    },
    emulateClick(event) {
      console.log(event)
      if (event.key === 'Enter' || event.key === ' ') {
        this.$refs.mainElement.dispatchEvent(new
Event('click'))
      }
    }
  }
```

```
}
</script>


<style lang="stylus" scoped>
@import '~@/variables'

.button
  height 34px
  width 34px
  box-shadow 0 0 0 0 focus-on-white
  transition all .2s
  cursor pointer
  border-radius 50%

  & .path
    fill text-on-white;
    fill-opacity 1

    stroke hovers-on-white
    stroke-opacity 0
    stroke-width 2

    transition all .2s



  &:hover .path
    fill-opacity 0
    stroke-opacity 1


  &:focus
    outline none
    box-shadow 0 0 0 60px focus-on-white


  &.on-black
    box-shadow 0 0 0 0 focus-on-black

    & .path
      fill text-on-black
      stroke none


    &:hover .path
      fill-opacity 1


    &:focus
      outline none
      box-shadow 0 0 0 60px focus-on-black
</style>
```

Але зміни одного кольору недостатньо, тому виконується метод setOnHover, який замінює Path зображення, а також SVG-зображенню присуджується роль кнопки за допомогою атрибутів tabindex, aria-label="more", role="button". Крім того, натискаючи на кнопку пробіл або enter емулюється натиск на кнопку, тому поведінка і семантика цього елемента аналогічна елементу <button>.

Надалі буде представлено частини коду елементів, які дуже схожі на More, тому буде представлено тільки відмінності. Оскільки форма кнопки Close (хрестика) не змінюється при наведенні курсора, вона обгорнута в тег <button>, що в свою чергу по замовчуванню надає їй поведінку кнопки, а також спрощує її структуру. За всі анімації відповідає директива transition, представлена Vue. В неї передається клас з розписаною анімацією. За анімовану зміну кнопки More на Close і назад відповідає код з файлу navigation.vue, наведений нижче:

```
@keyframes moreEnter
  from
    transform translateY(-50%) rotate(90deg) scale(1.5)
    opacity 0

  to
    transform translateY(-50%) rotate(0) scale(1)
    opacity 1


@keyframes moreLeave
  from
    transform translateY(-50%) rotate(0) scale(1)
    opacity 1

  to
    transform translateY(-50%) rotate(-90deg) scale(0.5)
    opacity 0
```

MoreEnter показує спочатку елемент в збільшеному, повернутому і прозорому стані і потім плавно змінює його на нормальний стан.

MoreLeave чинить аналогічно в зворотному порядку, але зі зменшенням (замість збільшення, як би це було, якби був дійсно зворотний порядок).

Потім ці анімації передаються CSS-класам в наступних рядках коду:

```
.button-enter-active
```

```
  animation moreEnter .3s
.button-leave-active
  animation moreLeave .3s
```

За появу і зникнення безпосередньо самих пунктів навігації відповідають анімації navigationEnter і navigationEnterMobile.

Фактично вони роблять одну задачу: щоб пункти меню плавно виїжджали зі зменшенням прозорості (ефект вицвітання, тільки навпаки). Ховається анімація за допомогою тієї ж навігації, тільки задом наперед:

```
.navigation-list-enter-active
  animation navigationListEnter .3s
  @media (max-width md)
    animation navigationListEnterMobile .3s
.navigation-list-leave-active
  animation navigationListEnter .3s reverse ease-in
  @media (max-width md)
    animation navigationListEnterMobile .3s reverse ease-in
```

Фактично мобільна версія робить те ж, тільки відрізняється від десктопної іншими властивостями трансформації.

Для коректного відображення робіт на сторінці портфоліо використовується компонент PortfolioItem.vue. Він приймає в себе посилання на зображення, назву проекту, його тип (особистий чи зроблений на замовлення), а також посилання на сторінку проекту. Нижче наведена частина коду:

```
<template>
<li class="portfolio-item-wrapper">
<router-link tag="div" class="portfolio-item" :to="link">
<a class="portfolio-item-container">
<img class="item-image" :src="image"/>
<div class="item-content">
<div class="item-type">{{ type }}</div>
<div class="item-name">{{ name }}</div>
</div>
</a>
</router-link>
</li>
</template>

<script>
import image from 'assets/smart-car-project-preview-sm.png'

export default {
props: {
image: String,
type: String,
```

```
name: String,
link: String
}
}
</script>
```

SmartPhoneImage.vue отримує растрове зображення і обертає її в екран iPhone X, як показано на зображенні, наведеному нижче. Крім того, цей компонент заставляє зображення зберігати оригінальні пропорції зображення при зміні розміру екрана. Це необхідно виправляти власноруч, адже якщо в CSS задавати відступи зверху/знизу в відсотках, він буде розраховуватися від ширини батьківського класу.

В папці Pages знаходяться всі сторінки веб-додатку. Кожна зі сторінок включає в себе декілька компонентів з папки components, але фактично там знаходиться звичайна верстка HTML. Для скорочення тексту не включено список CSS стилів, а просто приведу приклад коду сторінки portfolio:

```
<template>
  <main class='page portfolio'>
    <div class='heading-container'>
      <div class='heading' role="heading" aria-level="2">My
Projects</div>
      <div class='filters'>
        <button class='filter-button'>Web</button>
        <button class='filter-button'>Mobile Apps</button>
      </div>
    </div>
    <ul class="portfolio-list">
      <PortfolioItem v-for="item in items"
        :key="item.link"
        :image="item.image"
        :type="item.type"
        :name="item.name"
        :link="item.link"
      />
    </ul>
  </main>
</template>

<script>
import CloseButton from 'components/CloseButton'
import PortfolioItem from 'components/PortfolioItem'

const smartCarProject = {
  image: require('assets/smart-car-project-preview-sm.png'),
  type: 'commercial',
```

```
    name: 'Smart Car Project',
    link: '/smart-car-project'
  }

  export default {
    data() {
      return {
        items: [smartCarProject]
      }
    },
    components: {
      CloseButton,
      PortfolioItem
    },
    mounted() {
      this.$emit('setDarkTheme', false)
    }
  }
  </script>
```

Всі сторінки працюють приблизно однаково: кожна сторінка при рендері створює подію (рядок 43-45 в коді, наведеному вище), яка вказує на те, яку тему треба увімкнути – темну чи світлу. Цю подію бачить компонент App і передає цю інформацію компонентам, що не знаходяться на сторінці (навігація і нижній колонтитул).

Для збірки використовується Webpack, файл якого знаходиться в головній папці. В цьому файлі підключені всі плагіни і лоадери, які потрібні для того, щоб перетворити файли з форматом .vue в HTML, CSS та JavaScript.

```
extensions: ['.js', '.vue']
```

В фрагменті коду, наведеному вище, наведено можливість, яка дозволяє імпортувати файли .js і .vue, не вказуючи формат.

```
alias: {
      'components': path.resolve(__dirname, 'src/components'),
      'fonts': path.resolve(__dirname, 'src/fonts'),
      'assets': path.resolve(__dirname, 'src/assets'),
      'config': path.resolve(__dirname, 'src/app.config.js'),
      'libs': path.resolve(__dirname, 'src/libs'),
      '@': path.resolve(__dirname, 'src')
    }
```

Також в цьому файлі описана система alias (див. вище), яка дозволяє не використовувати відносні шляхи, тобто замість ../../../components/ImportedImage писати просто components/ImportedImage.

Останнім файлом, що використовується в проекті, є postcss.config.js. В ньому використовується єдиний плагін – prefixer. Він допомагає проставити вендорні префікси для неоднозначних CSS-властивостей, щоб вони коректно працювали на всіх типах браузерів.

```
module.exports = {
  plugins: {
    autoprefixer: { browsers: ['last 3 versions'] }
  }
}
```

# ВИСНОВКИ

Під час виконання даного дипломного завдання було розроблено інформаційний веб-ресурс портфоліо дизайнера.

Back end реалізований за допомогою таких веб технологій - PHP, MySql, HTML, CSS. Проект розроблений з розрахунком на подальше розширення і додавання нових функціональних можливостей. Тому інтеграція нових функцій займатиме мінімум часу і зусиль.

Результатом роботи є повністю функціональний, протестований і готовий до використання веб-сайт.

Тестування сайту відбувалося в два етапи. Перший етап полягав в тестуванні додатку в різних браузерах (Google Chrome, Mozila, Opera, Internet Explorer). Необхідність такого тестування полягає в тому, щоб виявити і виправити помилки, допущенні під час розробки. Другий етап тестування необхідний для опрацювання всіх можливих сценаріїв, для того, щоб переконатися, що все працює правильно і ніяких помилок в проектуванні не допущено.

Дипломна робота включає в себе вичерпний теоретичний і практичний опис проекту та кожної його частини.

Дипломний проект виконано у повній відповідності до завдання і всіма нормативними вимогами. Додаток дозволяє адміністратору керувати (додавати, редагувати, видаляти) всіма статтями, які відображаються на сайті, реєструватися новим користувачам, відвідувати особистий кабінет, та писати коментарі.

## ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ


1. Online Etymology Dictionary: Portfolio. URL: https:// www.etymonline.com/search?q=Portfolio (дата звернення: 25.02.2019).
2. Work.ua. Портфоліо: навіщо і кому воно потрібно. URL: https://www.work.ua/ru/articles/jobseeker/664/ (дата звернення: 25.02.2019).
3. Webdesignerdepot Staff. Interview with Dribbble's co-founder Dan cederholm. URL: https://www.webdesignerdepot.com/2011/05/interview-with-dribbbles-founder-dan-cederholm/ (дата звернення: 25.02.2019).
4. Aidan Hornsby. How Dribbble's 23 Remote Employees Keep Half a Million Designers Happy. URL: https://www.getflow.com/blog/project-management-at-dribbble (дата звернення: 25.02.2019).
5. Alexa.net. Behance.net Competitive Analysis, Marketing Mix and Traffic. URL: https://www.alexa.com/siteinfo/behance.net (дата звернення: 27.02.2019).
6. Frederic Lardinois. Adobe Acquires Social Portfolio Platform Behance To Power Its Creative Cloud Community Features. URL: https:// techcrunch.com/2012/12/20/adobe-acquires-social-media-platform-behance-to-power-its-creative-cloud-community-features/ (дата звернення: 27.02.2019).
7. Cameron Chapman. The Ultimate guide to everything Dribbble. URL: https://www.webdesignerdepot.com/2013/11/the-ultimate-guide-to-everything-dribbble/ (дата звернення: 27.02.2019).
8. Dave Gamache. Dribbble, one year on: does it live up to the hype? URL: https://thenextweb.com/dd/2011/03/29/dribbble-one-year-on-does-it-live-up-to-the-hype/ (дата звернення: 27.02.2019).
9. Dribbble.com. Introducing Rebounds. URL: https://dribbble.com/shots/ 3029-introducing-rebounds (дата звернення: 27.02.2019).
10. Dribbble.com. Introducing Playbook. URL: https://dribbble.com/shots/ 2720205-Introducing-Playbook (дата звернення: 28.02.2019).

11. Ray Wang. News Analysis: Adobe Behances The Creative Class with $150M Community Acquisition. URL: http://blog.softwareinsider.org/2013/02/11/ news-analysis-adobe-behances-the-creative-class-with-150m-community-acquisition/ (дата звернення: 29.02.2019).

12. HTML5Test. How well does your browser support HTML5? URL: https://html5test.com/results/desktop.html (дата звернення: 01.03.2019).

13. Wikipedia. Comparison of web browsers. URL: https://en.wikipedia.org/ wiki/Comparison_of_web_browsers (дата звернення: 01.03.2019).

14. Ethan Marcotte. Responsive Web Design. URL: http://alistapart.com/article/ responsive-web-design/ (дата звернення: 01.03.2019).

15. Bohemian Coding. Sketch 1.0 Finally Released. URL: https:// web.archive.org/web/20110711125106/http://www.bohemiancoding.com/about/blog/sketch-1-0-finally-released/ (дата звернення: 01.03.2019).

16. Firma.com. Figma vs Sketch. URL: https://www.figma.com/figma-vs-sketch/ (дата звернення: 01.03.2019).

17. Антон Олійник. Zeplin.io: Спосіб подружити дизайнера і верстальника. URL: https://ux.pub/kak-podruzhit-dizajnera-i-verstalshhika-zeplin-com/ (дата звернення: 04.03.2019).

18. Origami.design. Patches. URL: https://origami.design/documentation/ basics/ Patches.html (дата звернення: 04.03.2019).

19. Principle. Documentation. URL: https://principleformac.com/docs.html (дата звернення: 04.03.2019).

20. Docforge.com. Web Application Framework. URL: https://web.archive.org/web/20150723163302/http://docforge.com/wiki/Web_application_framework (дата звернення: 05.03.2019).

21. Shubham Kumar. Difference Between Library and Framework. URL: https://www.c-sharpcorner.com/UploadFile/a85b23/framework-vs-library/ (дата звернення: 05.03.2019).

22. Claire Legros. Logiciel libre les limites du modele du benelovat. URL: https://www.lemonde.fr/economie/article/2018/11/11/logiciel-libre-les-limites-du-modele-du-benevolat_5382054_3234.html (дата звернення: 05.03.2019).

23. Jacob Schatz. How we do Vue: one year later. URL: https:// about.gitlab.com/2017/11/09/gitlab-vue-one-year-later/ (дата звернення: 06.03.2019).

24. Black duck. Languages: Vue.js. URL: https://www.openhub.net/p/vue-js/analyses/latest/languages_summary (дата звернення: 06.03.2019).

25. Filipova O. Learning Vue.js 2 Learn how to build amazing and complex reactive web applications easily with Vue.js. — Birmingham: Packt Publishing Ltd, 2016. — 334 c. (дата звернення: 06.03.2019).

26. Anna Monus. 10 reasons to use a CSS Preprocessor in 2018. URL: https://raygun.com/blog/10-reasons-css-preprocessor/ (дата звернення: 11.03.2019).

27. Stylus. Built-in functions. URL: http://stylus-lang.com/docs/bifs.html (дата звернення: 11.03.2019).

28. Addy Osmani. Webpack Introduction. URL: https:// developers.google.com/web/fundamentals/performance/webpack/ (дата звернення: 11.03.2019).

29. Mark Brown. A Beginner’s Guide to Webpack 4 and Module Bundling. URL: https://www.sitepoint.com/beginners-guide-webpack-module-bundling/ (дата звернення: 11.03.2019).

30. Олександр Бабич. Лекція 7: Діаграми прецендентів: крупним планом. URL: https://www.intuit.ru/studies/courses/1007/229/lecture/5962 (дата звернення: 29.03.2019).

31. William Vincent. Static vs Dynamic Websites: Pros and Cons. URL: https://wsvincent.com/static-vs-dynamic-websites-pros-and-cons/ (дата звернення: 29.03.2019).

32. Олексій Сегодін. Що таке UX/UI дизайн насправді? URL: https://habr.com/ ru/post/321312/ (дата звернення: 29.03.2019).

33. Tim Worstall. The Real Problem with Apple: Skeumorphism in iOS. URL: https://www.forbes.com/sites/timworstall/2012/09/12/the-real-problem-with-apple-skeuomorphism-in-ios/ (дата звернення: 29.03.2019).

34. Kate Moran. Flat Design: Its Origins, Its Problems, and Why Flat 2.0 Is Better for Users. URL: https://www.nngroup.com/articles/flat-design/ (дата звернення: 29.03.2019).

35. Susan Fowler and Victor Stanwick (2004). Web Application Design Handbook (англ). Morgan Kaufmann. с. 50—51. (дата звернення: 01.04.2019).

36. Tobias Ahlin. Meaningful Motion With Action-Driven Animation. URL: https://tobiasahlin.com/blog/meaningful-motion-w-action-driven-animation/ (дата звернення: 01.04.2019).

37. Apple Human Interface Design. Visual Design: Animation. URL: https://developer.apple.com/design/human-interface-guidelines/ios/visual-design/animation/ (дата звернення: 01.04.2019).

38. Микола Геллар. Принципи анімації для UX і UI дизайнерів, від простого до складного. URL: https://ux.pub/principy-animacii-dlya-ux-i-ui-dizajnerov-ot-prostogo-k-slozhnomu/ (дата звернення: 01.04.2019).

39. Isssara Willenskomer. Creating Usability with Motion: The UX In Motion Manifesto. https://medium.com/ux-in-motion/creating-usability-with-motion-the-ux-in-motion-manifesto-a87a4584ddc (дата звернення: 02.04.2019).

# Додаток А

Файл app.vue

```
<template>
  <div class='main'>
    <Header v-bind:onBlack="isDarkTheme" />
    <router-view @setDarkTheme="setDarkTheme"></router-view>
    <Footer v-bind:onBlack="isDarkTheme" />
  </div>
</template>

<script>
import Header from 'components/Header'
import Footer from 'components/Footer'

export default {
  data() {
    return {
      isDarkTheme: false
    }
  },
  components: {
    Header,
    Footer
  },
  computed: {
    MainContainer() {
      return this.$route
    }
  },
  methods: {
    setDarkTheme(value) {
      this.isDarkTheme = value
    }
  }
}
</script>

<style lang='stylus' scoped>
.main
  display flex
  flex-direction column
  min-height 100vh
</style>

<style lang='stylus'>
  @import '~@/variables'
  html
    font-size 10px
```

```
  body
    font-family 'Inter', Arial, sans-serif
    overflow-x hidden

  .page
    margin-top 90px
    box-sizing border-box
    min-height 100vh
    flex-grow 1
</style>
```

Файл index.js

```
import Vue from 'vue'
import VueRouter from 'vue-router'
import App from '@/App'
import VueGlide from 'vue-glide-js'
import 'vue-glide-js/dist/vue-glide.css'
import 'normalize.css'
import '@/fonts/inter/inter.css'
import router from '@/router'

Vue.use(VueRouter)
Vue.use(VueGlide)

new Vue({
  el: document.querySelector('#app'),
  router,
  render: h => h(App)
})
```

Файл router.js

```
import VueRouter from 'vue-router'
import Home from '@/pages/Home'
import About from '@/pages/About'
import ContactMe from '@/pages/ContactMe'
import Portfolio from '@/pages/Portfolio'
import SmartCarProject from '@/pages/SmartCarProject'

export default new VueRouter({
  routes: [
    { path: '', component: Home },
    { path: '/home', component: Home },
    { path: '/about', component: About },
    { path: '/contact-me', component: ContactMe },
    {
      path: '/portfolio',
      component: Portfolio
    },
    {
      path: '/smart-car-project',
```

```
      component: SmartCarProject
    }
  ]
})
```

Файл variables.styl

```
// PALETTE

background-white = #ffffff
text-on-white = rgba(0, 0, 0, 0.8)
header-on-black = rgba(0, 0, 0, 0.72)
hovers-on-white = rgba(0, 0, 0, 0.64)
focus-on-white = rgba(0, 0, 0, 0.08)
footers-on-white = rgba(0, 0, 0, 0.04)

bacgkround-black = #000000
text-on-black = rgba(255, 255, 255, 0.8)
header-on-white = rgba(255, 255, 255, 0.72)
hovers-on-black = rgba(255, 255, 255, 0.64)
focus-on-black = rgba(255, 255, 255, 0.08)
footers-on-black = rgba(255, 255, 255, 0.04)
commercial-title = rgba(232, 137, 19, 0.8)
personal-title = rgb(208, 26, 33, 0.8)

base-shadow = 4px 4px 20px 0 footers-on-white

list-reset()
  list-style none
  padding 0
  margin 0

xs = 480px
sm = 640px
md = 768px
lg = 1024px
xl = 1280px
full-hd = 1920

header-height = 90px
max-container-width = 1490px
```

Файл header.vue

```
<template>
  <header class="header" :class="{ 'on-black': onBlack }">
    <div class="header-container">
      <Logo :onBlack="onBlack"/>
      <Navigation :onBlack="onBlack"/>
    </div>
  </header>
</template>
```

```
<script>
import Logo from 'components/Logo'
import Navigation from 'components/Navigation'

export default {
  data() {
    return {
      navIsOpen: false
    }
  },
  components: {
    Navigation,
    Logo,
  },
  props: {
    onBlack: Boolean
  }
}
</script>

<style lang="stylus" scoped>
@import '~@/variables'

.header
  position fixed
  display flex
  justify-content center
  align-items center

  width 100%
  height 90px
  box-sizing border-box

  padding-left 40px
  padding-right 40px

  z-index 50

  background header-on-white

  &.on-black
    background header-on-black

  @supports (backdrop-filter blur(20px))
    backdrop-filter blur(30px)

  @media (max-width md)
    padding-left 20px
    padding-right 20px
    height 80px

.header-container
```

```
    display flex
    justify-content space-between
    align-items center

    max-width max-container-width
    width 100%
    height 100%
    box-sizing border-box
</style>
```

Файл moreButton.vue

```
<template>
  <svg
    v-on:mouseover="setOnHover(true)"
    v-on:mouseout="setOnHover(false)"
    role="button"
    aria-label="more"
    tabindex="0"
    class="button"
    v-bind:class="{ 'on-black': onBlack }"
    v-bind:viewBox="viewBox"
    ref="mainElement"
    v-on:keypress="emulateClick"
    xmlns="http://www.w3.org/2000/svg"
  >
    <path class="path" fill-rule="evenodd" v-bind:d="path"
/>
  </svg>
</template>

<script>
const pathRegular = 'M15 30C6.716 30 0 23.284 0 15 0 6.716
6.716 0 15 0c8.284 0 15 6.716 15 15 0 8.284-6.716 15-15 15zm0-
13a2 2 0 1 0 0-4 2 2 0 0 0 0 4zm-8 0a2 2 0 1 0 0-4 2 2 0 0 0 0
4zm16 0a2 2 0 1 0 0-4 2 2 0 0 0 0 4z'
const viewBoxRegular = '0 0 30 30'
const pathHover = 'M17 33C8.163 33 1 25.837 1 17S8.163 1 17
1s16 7.163 16 16-7.163 16-16 16zm0-15a1 1 0 1 0 0-2 1 1 0 0 0 0
2zm-8 0a1 1 0 1 0 0-2 1 1 0 0 0 0 2zm16 0a1 1 0 1 0 0-2 1 1 0 0
0 0 2z'
const viewBoxHover = '0 0 34 34'

export default {
  data() {
    return {
      path: pathRegular,
      viewBox: viewBoxRegular
    }
  },
  props: {
    onBlack: Boolean
  },
```

```
      methods: {
        setOnHover(isHovered) {
          this.path = isHovered && !this.onBlack ? pathHover :
pathRegular
          this.viewBox = isHovered && !this.onBlack ?
viewBoxHover : viewBoxRegular
        },
        emulateClick(event) {
          console.log(event)
          if (event.key === 'Enter' || event.key === ' ') {
            this.$refs.mainElement.dispatchEvent(new
Event('click'))
          }
        }
      }
    }
    </script>


    <style lang="stylus" scoped>
    @import '~@/variables'

    .button
      height 34px
      width 34px
      box-shadow 0 0 0 0 focus-on-white
      transition all .2s
      cursor pointer
      border-radius 50%

      & .path
        fill text-on-white;
        fill-opacity 1

        stroke hovers-on-white
        stroke-opacity 0
        stroke-width 2

        transition all .2s



      &:hover .path
        fill-opacity 0
        stroke-opacity 1


      &:focus
        outline none
        box-shadow 0 0 0 60px focus-on-white


      &.on-black
```

```
        box-shadow 0 0 0 0 focus-on-black

        & .path
          fill text-on-black
          stroke none


        &:hover .path
          fill-opacity 1


        &:focus
          outline none
          box-shadow 0 0 0 60px focus-on-black
    </style>
```

Файл smartphoneImage.vue

```
    <template>
      <div class="smartphone-image">
        <div class="smartphone-image-stretcher"></div>
        <Image name="smartphone" class="phone" />
        <img :src="image" :alt="altAttribute" :aria-
hidden="!altAttribute" class="image" />
      </div>
    </template>

    <script>
    import Image from 'components/Image'

    export default {
      props: {
        image: String,
        altAttribute: String
      },
      components: {
        Image
      }
    }
    </script>


    <style lang="stylus" scoped>
    .smartphone-image
      position relative
      overflow hidden

    .smartphone-image-stretcher
      content ''
      padding-bottom 199.4%

    .phone
      position absolute
```

```
    top 0
    left 0
    width 100%
    height 100%
    z-index 3

  .image
    position absolute
    top 0
    left 0
    object-fit cover
    height 100%
    width 100%
    border-radius 20px
    z-index 2

  </style>
```

Файл portfolioItem.vue

```
  <template>
    <li class="portfolio-item-wrapper">
      <router-link tag="div" class="portfolio-item"
:to="link">
        <a class="portfolio-item-container">
          <img class="item-image" :src="image"/>
          <div class="item-content">
            <div class="item-type">{{ type }}</div>
            <div class="item-name">{{ name }}</div>
          </div>
        </a>
      </router-link>
    </li>
  </template>

  <script>
  import image from 'assets/smart-car-project-preview-sm.png'

  export default {
    props: {
      image: String,
      type: String,
      name: String,
      link: String
    }
  }
  </script>


  <style lang="stylus" scoped>
  @import '~@/variables'

  .portfolio-item-wrapper
```

```
  width 22vw
  @media (min-width: 1920px)
    width 430px
  @media (max-width: lg)
    width 300px
  @media (max-width: sm)
    width 100%

.portfolio-item
  position relative
  height 0
  padding-bottom 105%
  border-radius 20px
  box-shadow base-shadow
  overflow hidden
  @media (max-width: sm)
    border-radius 0
    padding-bottom 90%

.portfolio-item-container
  position absolute
  text-decoration none
  top 0
  left 0
  display block
  height 100%
  width 100%

.item-image
  background black
  height 70%
  width 100%
  object-fit cover

.item-content
  display flex
  flex-direction column
  justify-content center
  align-items flex-start
  padding 0 9%
  height 30%
  @media (max-width: sm)
    box-sizing border-box
    padding 0 18px
    width 100%
    flex-direction row-reverse
    align-items center
    justify-content space-between

.item-type
  color personal-title
  text-transform uppercase
  font-size 0.8vw
```

```
      padding-bottom 0.04vw
      font-weight bold
      @media (min-width: 1920px)
        font-size 16px
        padding-bottom 8px
      @media (max-width: lg)
        font-size 16px
        padding-bottom 8px

    .item-name
      font-size 1.2vw
      font-weight bold
      color text-on-white
      @media (min-width: 1920px)
        font-size 24px
      @media (max-width: lg)
        font-size 24px
    </style>
```

Файл navigation.vue

```
    <template>
      <div class="navigation" :class="{ 'on-black': onBlack }">
        <transition name="navigation-list">
          <ul v-show="isOpen" class="navigation-list">
            <router-link v-on:click.native="setIsOpen(false)"
class="navigation-item" tag="li" to="/about">
              <a class="navigation-link"><div class="link-
caption">About</div></a>
            </router-link>
            <router-link v-on:click.native="setIsOpen(false)"
class="navigation-item" tag="li" to="/portfolio">
              <a class="navigation-link"><div class="link-
caption">Portfolio </div></a>
            </router-link>
            <router-link v-on:click.native="setIsOpen(false)"
class="navigation-item" tag="li" to="/contact-me">
              <a class="navigation-link"><div class="link-
caption">Contact me</div></a>
            </router-link>
          </ul>
        </transition>
        <div class="button-block">
          <transition name="button">
            <MoreButton v-show="!isOpen" :onBlack="onBlack" v-
on:click.native="setIsOpen(true)" class="button" />
          </transition>
          <transition name="button">
            <CloseButton v-show="isOpen" :onBlack="onBlack" v-
on:click.native="setIsOpen(false)" class="button" />
          </transition>
        </div>
      </div>
```

```
</template>

<script>
import ImportedImage from 'components/ImportedImage'
import MoreButton from 'components/MoreButton'
import CloseButton from 'components/CloseButton'

export default {
  data() {
    return {
      isOpen: false
    }
  },
  methods: {
    setIsOpen(shouldBeOpen) {
      this.isOpen = shouldBeOpen
    }
  },
  components: {
    MoreButton,
    CloseButton
  },
  props: {
    onBlack: Boolean
  }
}
</script>


<style lang="stylus" scoped>
@import '~@/variables'

.navigation
  position relative

  height 100%
  width 35px
  @media (max-width: md)
    position static

.navigation-list
  list-reset()

  position absolute
  top 0
  left 0
  transform translateX(-100%)

  display flex
  height 100%
  @media (max-width: md)
    flex-direction column
    width 100vw
```

```
    top 90px
    z-index 50
    height 100vh
    background header-on-white
    transform translate(0)
    .on-black &
      background header-on-black
    @supports (backdrop-filter blur(20px))
      backdrop-filter blur(30px)

.navigation-link
  display flex
  justify-content center
  align-items center

  height 100%
  width auto
  box-sizing border-box

  text-decoration none
  font-size 18px
  font-weight bold
  color text-on-white
  white-space nowrap
  cursor pointer
  .on-black &
    color text-on-black
  @media (max-width: md)
    font-size 34px
    justify-content flex-start

.link-caption
  padding-left 20px
  padding-right 20px
  &:last-child
    padding-right 40px
  @media (max-width: md)
    margin 0 36px
    padding-top 20px
    padding-left 20px


.button-block
  position relative

  height 100%
  width 34px

.button
  position absolute
  top 50%
  transform translateY(-50%)
```

```
@keyframes moreEnter
  from
    transform translateY(-50%) rotate(90deg) scale(1.5)
    opacity 0

  to
    transform translateY(-50%) rotate(0) scale(1)
    opacity 1


@keyframes moreLeave
  from
    transform translateY(-50%) rotate(0) scale(1)
    opacity 1

  to
    transform translateY(-50%) rotate(-90deg) scale(0.5)
    opacity 0


@keyframes navigationListEnter
  from
    opacity 0
    transform translateX(-80%)

  to
    transform translateX(-100%)
    opacity 1

@keyframes navigationListEnterMobile
  from
    opacity 0
    transform translateX(20%)
  to
    transform translateX(0)
    opacity 1

.button-enter-active
  animation moreEnter .3s
.button-leave-active
  animation moreLeave .3s
.navigation-list-enter-active
  animation navigationListEnter .3s
  @media (max-width md)
    animation navigationListEnterMobile .3s
.navigation-list-leave-active
  animation navigationListEnter .3s reverse ease-in
  @media (max-width md)
    animation navigationListEnterMobile .3s reverse ease-in
</style>
```

Файл navigation.vue

```
<template>
  <footer class="footer" :class="{ 'on-black': onBlack }">
    <div class="footer-container">
      <Logo class='logo' :onBlack="onBlack" />
      <div class="social-links">
        <a class="social-link"
href="https://www.behance.net/oryashh" target="_blank">
          <ImportedImage name="behance" :injectable="true"
aria-label="behance" />
        </a>
        <a class="social-link"
href="https://dribbble.com/beerbongsnbentleys" target="_blank">
          <ImportedImage name="dribble" :injectable="true"
aria-label="dribble" />
        </a>
      </div>
    </div>
  </footer>
</template>

<script>
import Logo from 'components/Logo'
import ImportedImage from 'components/ImportedImage'

export default {
  components: {
    Logo,
    ImportedImage
  },
  props: {
    onBlack: Boolean
  }
}
</script>


<style lang="stylus" scoped>
@import '~@/variables'

.footer
  display flex
  justify-content center
  align-items center

  height 90px
  width 100%
  box-sizing border-box

  background white

  &.on-black
```

```
    background black

.footer-container
  display flex
  justify-content space-between
  align-items center

  max-width max-container-width
  width 100%
  height 100%
  box-sizing border-box

  padding-left 40px
  padding-right 40px

  background footers-on-white

  .on-black &
    background footers-on-black

.social-links
  display flex

  height 100%

.social-link
  display flex
  justify-content center
  align-items center

  height 100%

  padding-left 40px
  padding-right 40px

  transition all .2s

  color text-on-white
  cursor pointer
  @media (max-width md)
    padding-left 18px
    padding-right 18px
  .on-black &
    color text-on-black
  &:hover
    color personal-title
  &:first-child
    margin-left auto
  &:last-child
    padding-right 0


@media (max-width md)
```

```
    .logo
      display none
  </style>
```

Файл portfolio.vue

```
  <template>
    <main class='page portfolio'>
      <div class='heading-container'>
        <div class='heading' role="heading" aria-level="2">My
Projects</div>
        <div class='filters'>
          <button class='filter-button'>Web</button>
          <button class='filter-button'>Mobile Apps</button>
        </div>
      </div>
      <ul class="portfolio-list">
        <PortfolioItem v-for="item in items"
          :key="item.link"
          :image="item.image"
          :type="item.type"
          :name="item.name"
          :link="item.link"
        />
      </ul>
    </main>
  </template>

  <script>
  import CloseButton from 'components/CloseButton'
  import PortfolioItem from 'components/PortfolioItem'

  const smartCarProject = {
    image: require('assets/smart-car-project-preview-sm.png'),
    type: 'commercial',
    name: 'Smart Car Project',
    link: '/smart-car-project'
  }

  export default {
    data() {
      return {
        items: [smartCarProject]
      }
    },
    components: {
      CloseButton,
      PortfolioItem
    },
    mounted() {
      this.$emit('setDarkTheme', false)
    }
  }
```

```
</script>

<style lang="stylus" scoped>
@import '~@/variables'

.portfolio
  display flex
  flex-direction column
  box-sizing border-box
  padding 0 40px
  padding-top calc(90px + 40px)
  margin 0 auto
  width 100%
  max-width 1490px
  background-color #f5f5f5
  @media (max-width: md)
    padding-left 0
    padding-right 0

.portfolio-list
  list-reset()
  display flex
  flex-wrap wrap
  justify-content space-between
  & > li
    margin-bottom 40px
  @media (max-width: md)
    flex-direction column
    justify-content flex-start
    align-items center

.heading-container
  display flex
  justify-content space-between
  margin-bottom 40px
  @media (max-width: md)
    padding: 0 18px
    flex-direction column-reverse
    margin-bottom 16px

.heading
  color text-on-white
  font-weight bold
  font-size 36px

.filters
  display flex
  align-items center
  @media (max-width: md)
    align-self flex-end
    margin-bottom 5px

.filter-button
```

```
  color text-on-black
  background text-on-white
  border none
  display flex
  font-size 16px
  padding 8px 16px
  border-radius 16px
  margin-right 16px
  cursor pointer
  &:last-child
    margin-right 0
</style>
```

**БІБЛІОГРАФІЧНА ДОВІДКА**

Тема дипломної роботи: Інформаційний веб-ресурс портфоліо дизайнера.

Обсяг пояснювальної записки:. ___62____аркуші

Перелік графічних матеріалів:

- таблиць__
- рисунків_____22_____
- додатків_____1 на 17_____аркушів.

Дата закінчення дипломного проекту: “__”_____2019 р.

Студент – дипломник _____
(підпис) (розшифровка підпису)