Міністерство освіти і науки України

Луцький національний технічний університет
(повне найменування закладу вищої освіти)

Факультет комп'ютерних та інформаційних технологій
(повне найменування факультету)

Кафедра комп'ютерної інженерії та кібербезпеки
(повне найменування кафедри)

КВАЛІФІКАЦІЙНА РОБОТА ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

ОНЛАЙН РЕСУРС ДЛЯ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ВЕДЕННЯ БІЗНЕСУ

ONLINE RESOURCE FOR INCREASING BUSINESS EFFICIENCY

спеціальність 123 Комп'ютерна інженерія
(шифр і назва спеціальності)

освітня програма Комп'ютерна інженерія
(назва освітньої програми)

Виконав: здобувач вищої освіти
групи КІ-41
Єремейчук Назарій Іванович

(підпис)

Керівник:
к.т.н., доцент
Костючко Сергій Миколайович

(підпис)

Кваліфікаційну роботу
допущено до захисту
« _____ » червня 2023 р.
Гарант освітньої програми:
к.т.н., доцент
Лавренчук Світлана Василівна

(підпис)

Луцьк – 2023 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *комп'ютерної інженерії та кібербезпеки*
Ступінь вищої освіти: *бакалавр*
Галузь знань: *12 Інформаційні технології*
Спеціальність: *123 Комп'ютерна інженерія*
Освітня програма: *«Комп'ютерна інженерія»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ проф. Н.Чернящук
«_____» _____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Єремейчуку Назарію Івановичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *Онлайн ресурс для підвищення ефективності ведення бізнесу*

Керівник роботи *к.т.н., доцент Костючко С. М.*

затверджені наказом закладу вищої освіти від «28» грудня 2022 року № 982/01-02
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи 01.06.2023р.

3. Вихідні дані до роботи *Джерелом розробки є науково-технічна література та публікації в періодичних виданнях з даного питання, опубліковані зарубіжні та вітчизняні роботи в даній області та різні інтернет-ресурси технічного спрямування*

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити):
Вступ
Визначення вимог до веб-платформи
Розробка архітектури та функціоналу веб-платформи
Опис програмної реалізації веб-сайту та веб-додатка
Висновки

5. Перелік графічного (ілюстративного) матеріалу:

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Визначення вимог до веб-платформи	*Костючко С.М.*		
Розробка архітектури та функціоналу веб-платформи	*Костючко С.М.*		
Опис програмної реалізації веб-сайту та веб-додатка	*Костючко С.М.*		
Висновки			

7. Дата видачі завдання 01.11.2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	*Обгрунтування теми*	До 15.11.2022 р.	Виконано
2.	*Огляд літератури із досліджуваної проблеми*	До 17.12.2022 р.	Виконано
3.	*Розробка технічного завдання, вибір методів та засобів реалізації задачі*	До 02.02.2023 р.	Виконано
4.	*Розробка структури прототипу та проектування системи*	До 02.03.2023 р.	Виконано
5.	*Описати програмне та апаратне середовище функціонування об'єкта програмування*	До 02.04.2023 р.	Виконано
6.	*Практична реалізація об'єкта програмування*	До 15.04.2023 р.	Виконано
7.	*Висновки та пропозиції*	До 02.05.2023 р.	Виконано
8.	*Формування додатків*	До 15.05.2023 р.	
9.	*Нормоконтроль*	До 25.05.2023 р.	Виконано
10.	*Інструментальна перевірка на академічний плагіат*	До 01.6.2023 р	Виконано
11.	*Представлення кваліфікаційної роботи бакалавра до захисту*	До 07.06.2023 р.	Виконано

Здобувач вищої освіти _____ (Єремейчук Н.І.)
(підпис) (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ (Костючко С.М.)
(підпис) (прізвище, ініціали)

АНОТАЦІЯ

Єремейчук Н.І. Онлайн ресурс для підвищення ефективності ведення бізнесу. Рукопис.

Кваліфікаційна робота бакалавра ОП «Комп'ютерна інженерія» спеціальності 123 Комп'ютерна інженерія. Луцький національний технічний університет. Луцьк, 2023.

Кваліфікаційна робота складається з вступу, трьох розділів, висновків, списку використаних джерел, 5 додатків.

Перший розділ присвячено огляду літературних джерел, актуальності теми та її значення для бізнесу; програмному забезпеченню поведінкового аналізу; опису існуючих аналогів веб-платформи та їх аналіз; формулюванню завдань та критеріїв для розробки веб-платформи.

В другому розділі здійснено вибір технологій для розробки веб-платформи та опис її архітектури.

В третьому розділі описано процес розробки та оптимізації програмного продукту, його основні елементи.

Об'єкт дослідження – засоби та форми розробки та налаштування web-платформ, розробка її архітектури та дизайну, реалізація необхідного функціоналу.

Предмет розробки – web-платформа та її оптимізація з метою забезпечення максимальної продуктивності та стабільності.

Метою роботи є розробка web-платформи, яка допоможе підвищити ефективність ведення бізнесу за допомогою забезпечення контролю та аналізу ключових показників діяльності.

Ключові слова: верстка, платформа, оптимізація, дизайн, розробка.

ABSTRACT

Yeremeychuk N.I. Online resource for improving business efficiency. Manuscript.

Qualification work of Bachelor's degree in "Computer Engineering" of specialty 123 Computer Engineering. Lutsk National Technical University. Lutsk, 2023.

The qualification work consists of an introduction, three chapters, conclusions, a list of used sources, and 5 appendices.

The first chapter is devoted to the review of literary sources, the relevance of the topic, and its significance for business; software for behavioral analysis; description of existing analogs of the web platform and their analysis; formulation of tasks and criteria for the development of the web platform.

The second chapter involves the selection of technologies for the development of the web platform and a description of its architecture.

The third chapter describes the process of development and optimization of the software product, its main components.

The object of research is the means and forms of development and configuration of web platforms, the development of its architecture and design, the implementation of the necessary functionality.

The subject of development is the web platform and its optimization in order to ensure maximum productivity and stability.

The aim of the work is to develop a web platform that will help improve business efficiency by providing control and analysis of key performance indicators.

Keywords: markup, platform, optimization, design, development.

ЗМІСТ

ВСТУП

Актуальність теми. В сучасних умовах підвищення ефективності ведення бізнесу є однією з ключових задач для підприємств будь-якого розміру і напрямку діяльності. Для досягнення цієї мети використовуються різноманітні інструменти та технології, серед яких особливе місце займають веб-платформи.

Метою роботи є розробка web-платформи, яка допоможе підвищити ефективність ведення бізнесу за допомогою забезпечення контролю та аналізу ключових показників діяльності.

Об'єкт дослідження – засоби та форми розробки та налаштування web-платформ, розробка її архітектури та дизайну, реалізація необхідного функціоналу.

Предмет розробки – web-платформа та її оптимізація з метою забезпечення максимальної продуктивності та стабільності.

Завдання, які необхідно виконати:

– провести аналіз актуальності та визначити вимоги до веб-платформи для ефективності ведення бізнесу;

– проаналізувати технології які використовуються для веб-платформ;

– розробити архітектуру програмного продукту

– розробити веб-платформу у відповідності до поставлених вимог;

– провести тестування та оптимізацію.

Практичне значення отриманих результатів:

– відповідні вивчені елементи можуть бути використані для покращення навігації, вмісту та загальної функціональності веб-платформи;

– за допомогою отриманих результатів можна вдосконалити процес замовлення в додатку, що зробить процес замовлення більш ефективним і привабливим для користувачів.

РОЗДІЛ 1
ВИЗНАЧЕННЯ ВИМОГ ДО ВЕБ-ПЛАТФОРМИ

1.1 Актуальність теми та її значення для бізнесу

Сьогодні в епоху цифрових технологій і Інтернету ведення бізнесу стає все більш складним і конкурентним завданням. Оптимізація роботи і збільшення продуктивності перетворюється на найважливіші завдання для бізнесу будь-якої масштабності.

Щоб протистояти конкуренції, компанії шукають шляхи, які дозволяють оптимізувати внутрішні бізнес-процеси та збільшити продуктивність роботи співробітників. Водночас, виникає все більше вимог до точності та оперативності рішень, прийнятих компанією, що залежать від ефективного управління даними, процесами та комунікаціями.

Створення веб-платформи для підвищення ефективності ведення бізнесу є актуальною та значущою темою для компаній різного масштабу та профілю. Вона дозволяє зробити роботу бізнесу більш ефективною та продуктивною, зменшити час та зусилля, витрачені на виконання повсякденних завдань, збільшити точність та оперативність прийняття рішень та зменшити ризики помилок.

Також веб-платформа дозволить покращити взаємодію з клієнтами та співробітниками, зменшити витрати на оренду офісних приміщень, скоротити час на обробку документів, зменшити кількість помилок, що дозволить скоротити витрати на їх виправлення. Таким чином, веб-платформа може стати інструментом, що дозволить компаніям збільшити свою прибутковість та конкурентоспроможність на ринку.

Актуальність теми створення веб-платформи для підвищення ефективності ведення бізнесу не може бути переоцінена. Сьогодні вже немає сумнівів у тому, що високотехнологічні рішення забезпечують значний рівень ефективності в будь-якій галузі бізнесу. Віртуальні майданчики та інтернет-технології стають все більш доступними і зручними, що дозволяє створювати та забезпечувати роботу дійсно ефективних веб-платформ [1].

Зараз бізнес-процеси повинні бути організовані не тільки ефективно, але й якісно. Підвищення ефективності ведення бізнесу є одним з найважливіших завдань для будь-якого підприємства. На сьогоднішній день на ринку існує велика кількість різноманітних веб-платформ, що надають різноманітний функціонал для роботи з клієнтами, фінансами та замовленнями. Проте не всі вони є оптимальними і ефективними, а деякі з них можуть бути недостатньо гнучкими та не забезпечувати необхідний рівень зручності та швидкості роботи.

Створення веб-платформи, яка відповідає конкретним вимогам та потребам бізнесу, може допомогти вирішити багато проблем, що виникають в процесі його ведення. Така платформа забезпечить оптимальний рівень зручності та ефективності ведення бізнесу, дозволяючи зосередитися на розвитку підприємства та його стратегічних задачах [7].

Разом з тим, в наш час дедалі більше компаній звертають увагу на використання технологій для автоматизації та оптимізації своїх бізнес-процесів. Відповідно, розробка веб-платформи для підвищення ефективності ведення бізнесу стає дедалі більш актуальною та потрібною.

1.2 Програмне забезпечення поведінкового аналізу

Визначення вимог до веб-платформи є важливим етапом в розробці будь-якого програмного забезпечення. Для досягнення мети - підвищення ефективності ведення бізнесу, необхідно визначити як функціональні, так і нефункціональні вимоги до веб-платформи [17].

Функціональні вимоги - це опис того, що має робити система. Для веб-платформи такі вимоги можуть включати управління клієнтами, замовленнями, фінансами, створення звітів та аналітики, реалізацію онлайн-магазину та інші. Наприклад, функціональна вимога для управління клієнтами може включати створення профілю клієнта, зберігання даних клієнта, історію замовлень та контактну інформацію.

Нефункціональні вимоги - це опис того, як система повинна працювати.

Наприклад, вимоги до продуктивності, безпеки, масштабованості, доступності та інші. Нефункціональна вимога до продуктивності може включати максимальний час відповіді на запит користувача, або максимальну кількість запитів, які можуть бути оброблені веб-платформою за певний час.

Опис потенційних користувачів та їх потреби також є важливим елементом визначення вимог до веб-платформи. Наприклад, веб-платформа може мати різні типи користувачів з різними потребами, такими як адміністратори, менеджери з продажу, клієнти та інші. Відповідно, вимоги до функціоналу та інтерфейсу веб-платформи для кожної групи користувачів можуть відрізнятися.

1.2.1 Функціональні вимоги

Функціональні вимоги є однією з головних складових визначення вимог до веб-платформи для підвищення ефективності ведення бізнесу. Ці вимоги описують, які функції повинна виконувати веб-платформа, щоб задовольнити потреби користувачів та допомогти їм підвищити ефективність свого бізнесу.

Наприклад, функціональні вимоги можуть включати можливість створення та зберігання різних типів документів, в тому числі замовлень, рахунків, звітів та іншої документації, пов'язаної з діяльністю компанії. Крім того, можуть бути вимоги до можливості обміну даними між користувачами та автоматизації деяких бізнес-процесів, таких як створення замовлень або відстеження стану проектів.

Функціональні вимоги можуть також включати можливості аналізу даних, створення звітів та графіків для візуалізації результатів діяльності компанії, що дозволяє зробити більш обґрунтовані рішення та покращити ефективність бізнесу [8].

Важливим аспектом визначення функціональних вимог є їх адаптація до потреб конкретних користувачів. Наприклад, веб-платформа може бути призначена для використання в малих бізнесах, середніх компаніях або великих корпораціях, тому функціональні вимоги повинні бути адаптовані до потреб різних розмірів та типів бізнесу.

Крім того, функціональні вимоги повинні бути чітко визначені та

документовані, щоб забезпечити якість та стабільність роботи.

1.2.2 Нефункціональні вимоги

Нефункціональні вимоги до веб-платформи визначають характеристики, які не відносяться до її функціональності, але впливають на її ефективність, надійність, безпеку, масштабованість, доступність, зручність користування та інші аспекти. До нефункціональних вимог можна віднести наступні:

- Ефективність: платформа повинна працювати швидко та ефективно, забезпечуючи швидкий відгук на запити користувачів, мінімізуючи час завантаження сторінок та оптимізуючи ресурси сервера.
- Надійність: платформа повинна бути надійною та стійкою до збоїв, забезпечуючи безперебійну роботу протягом тривалого часу та мінімізуючи ризики виникнення помилок.
- Безпека: платформа повинна забезпечувати високий рівень безпеки, захищаючи дані користувачів від несанкціонованого доступу та зловживань, забезпечуючи конфіденційність, цілісність та доступність.
- Масштабованість: платформа повинна бути готовою до масштабування та забезпечувати стійку роботу при збільшенні обсягу даних та кількості користувачів.
- Доступність: платформа повинна бути доступною для користувачів з різних пристроїв та браузерів, забезпечуючи підтримку різних мов та культур.
- Зручність користування: платформа повинна бути зручною та простою в користуванні, забезпечуючи інтуїтивно зрозумілий інтерфейс та легкий доступ до функцій та опцій.

1.2.3. Опис потенційних користувачів та їх потреби

Потенційні користувачі веб-платформи можуть бути різного типу, такі як власники бізнесу, менеджери, фінансові аналітики та клієнти. Кожна з цих категорій користувачів має свої потреби від веб-платформи [14].

Власники бізнесу можуть використовувати платформу для моніторингу фінансової діяльності свого бізнесу, стеження за замовленнями та клієнтською базою, аналізу даних та прийняття стратегічних рішень.

Менеджери можуть використовувати платформу для керування процесами замовлення, відстеження діяльності співробітників та контролю якості продукту.

Фінансові аналітики можуть використовувати платформу для аналізу фінансових даних та створення звітів про фінансову діяльність бізнесу.
Клієнти можуть використовувати платформу для замовлення товарів та послуг, стеження за статусом замовлень, спілкування з представниками бізнесу та отримання підтримки.

Потенційні користувачі веб-платформи мають різні потреби та вимоги до функціоналу, що має бути враховано під час її розробки. Важливо забезпечити зручний та простий інтерфейс для користувачів різних категорій та забезпечити необхідний функціонал для кожного з них.

1.3 Опис існуючих аналогів веб-платформи та їх аналіз

Опис існуючих аналогів веб-платформи та їх аналіз є важливим етапом при розробці нового продукту. Він дозволяє оцінити ринок, зрозуміти потреби користувачів та виявити конкурентні переваги.

Один з аналогів, які можна розглянути, це Salesforce - хмарна CRM-система, яка дозволяє компаніям вести облік клієнтів, продажів та маркетингу. Salesforce має багато функцій та інтеграцій з іншими системами, а також має велику спільноту користувачів та розробників.

Інший аналог - Zoho CRM - також хмарна CRM-система з багатьма функціями та інтеграціями. Вона також має додаткові модулі для управління фінансами, ресурсами та проектами [4].

Salesforce та Zoho CRM є сильними гравцями на ринку CRM-систем. Однак, їхні вартості можуть бути надто високими для менших компаній. Також, на ринку є менш відомі CRM-системи, які можуть бути більш доступними та підходять для менших бізнесів.

Аналізуючи існуючі аналоги, можна зрозуміти, які функції та інтеграції вже існують на ринку, які можуть бути корисними для користувачів та які

можуть бути конкурентними перевагами для нової веб-платформи. Також, можна виявити недоліки наявних аналогів та пропонувати кращі рішення.

1.3.1 Порівняння функціоналу та можливостей існуючих аналогів

При порівнянні функціоналу та можливостей існуючих аналогів веб-платформи потрібно звернути увагу на такі аспекти:

- Функціональні можливості: детально описати, які функції доступні користувачам на існуючих платформах та чи є необхідні функції для ефективного ведення бізнесу.
- Комплексність рішення: порівняти, чи містять існуючі платформи модулі управління клієнтами, фінансами, замовленнями та інші, або потребують додаткових інтеграцій.
- Легкість використання: оцінити, наскільки легко користувачі можуть зрозуміти і використовувати існуючі платформи, чи містять вони інтуїтивно зрозумілий інтерфейс та документацію.
- Складність встановлення та підтримки: порівняти, наскільки складно встановлювати та підтримувати існуючі платформи, чи мають вони докладну документацію та підтримку.
- Ціна: оцінити вартість користування існуючими платформами та порівняти з бюджетом на розробку власної веб-платформи.
- Масштабованість та гнучкість: оцінити можливість розширення та адаптації існуючих платформ під потреби конкретного бізнесу.

Після порівняння функціоналу та можливостей існуючих аналогів потрібно зробити висновок про те, чи варто розробляти власну веб-платформу або використати один з існуючих аналогів.

1.3.2 Аналіз переваг та недоліків існуючих аналогів

Аналіз переваг та недоліків існуючих аналогів є важливим етапом розробки веб-платформи, оскільки дозволяє побачити їхні особливості та визначити, які проблеми необхідно вирішити в новій системі [9]. Деякі переваги та недоліки можуть відрізнятися залежно від контексту використання, але основні питання, які потрібно враховувати при аналізі, включають наступне:

Переваги існуючих аналогів:

- Можливість взаємодії з великою кількістю клієнтів та забезпечення стабільної роботи під великим навантаженням.
- Наявність розвинутої системи безпеки та авторизації, яка забезпечує захист даних клієнтів та компанії.
- Наявність різних інструментів для аналізу та звітності, які дозволяють збирати та обробляти інформацію про фінансовий стан компанії та поведінку клієнтів.
- Наявність різних інструментів для покращення зручності користування та підвищення задоволення клієнтів від взаємодії з платформою.

Недоліки існуючих аналогів:

- Обмеження функціоналу та можливостей внесення змін у систему залежно від потреб компанії.
- Високі витрати на підтримку та розвиток існуючої системи.
- Неповна автоматизація певних процесів, що вимагає більшої ручної роботи з боку працівників компанії.
- Неадекватна відповідь на потреби клієнтів та бажання з використання сучасних інструментів для зручності користування платформою.

1.4 Формулювання завдань для розробки веб-платформи

Формулювання завдань для розробки веб-платформи може включати в себе наступні етапи:

- Створення загальної мети проекту: Визначте, що саме ви хочете отримати від своєї веб-платформи. Наприклад, це може бути підвищення продуктивності бізнесу, залучення нових клієнтів, поліпшення відносин з існуючими клієнтами, оптимізація фінансового управління тощо.
- Аналіз цільової аудиторії: Визначте, яка цільова аудиторія буде використовувати вашу веб-платформу. Розгляньте їх потреби, інтереси та очікування.

- Формулювання функціональних вимог: Сформулюйте, які функції повинна мати ваша веб-платформа. Наприклад, можуть бути потрібні модулі для управління клієнтами, фінансами, замовленнями, інвентарем тощо.
- Формулювання нефункціональних вимог: Окрім функціональних вимог, важливо сформулювати такі параметри, як швидкість роботи, безпеку даних, масштабованість та інші параметри, які не пов'язані з функціоналом.
- Визначення дедлайнів: Визначте, коли вам потрібно завершити проект. Створіть чіткий графік робіт, який допоможе вам зберігати проект на правильному шляху.
- Визначення бюджету: Визначте, скільки коштуватиме проект. Включіть в цю суму витрати на розробку, тестування, зберігання та розгортання.
- Визначення команди проекту: Визначте, які люди будуть працювати над проектом. Розгляньте, які навички та знання потрібні для розробки веб-платформи, та які ресурси будуть доступні для реалізації проекту. Включіть у команду розробників, тестувальників, дизайнерів та інших фахівців, які можуть допомогти вам реалізувати проект успішно.
- Визначення технічних вимог: Визначте, які технічні рішення будуть використовуватись для розробки веб-платформи. Розгляньте такі параметри, як вибір мов програмування, баз даних, веб-серверів та інші технології.
- Розробка прототипу: Створіть прототип веб-платформи, який допоможе вам визначити, як буде виглядати та працювати ваша платформа.
- Тестування та виправлення помилок: Після створення прототипу виконайте тестування та виправлення помилок. Цей етап допоможе вам забезпечити якість та безпеку вашої веб-платформи.
- Розгортання веб-платформи: Після успішного завершення розробки та тестування, розгорніть вашу веб-платформу на сервері та забезпечте доступ до неї користувачам.

– Підтримка та розвиток: Після розгортання веб-платформи необхідно забезпечити підтримку та розвиток платформи. Включіть у свій план підтримку та розвиток платформи, також варто планувати можливі розширення функціоналу платформи та вдосконалення інтерфейсу користувача.

1.5 Формулювання критеріїв оцінки результатів розробки

Для формулювання критеріїв оцінки результатів розробки веб-платформи можна використовувати наступні етапи:

– Визначення метрик відповідно до мети проекту: Для оцінки результатів розробки потрібно визначити метрики, які допоможуть вам зрозуміти, наскільки успішним був ваш проект. Наприклад, якщо ваша мета - підвищення продуктивності бізнесу, то можна використовувати метрики, які відображають зменшення часу виконання завдань, збільшення кількості виконаних завдань тощо.
– Формулювання вимог до якості: Визначте, які вимоги до якості повинна відповідати ваша веб-платформа. Наприклад, якість коду, швидкість завантаження сторінок, надійність роботи тощо [21].
– Визначення критеріїв успішності: Визначте критерії успішності для кожного зі специфічних завдань вашої веб-платформи. Наприклад, якщо ви розробляєте платформу для онлайн-торгівлі, критерієм успішності може бути збільшення кількості замовлень та задоволеність клієнтів.
– Визначення критеріїв ефективності: Визначте критерії ефективності для розробки, які відображають ефективність використання ресурсів, таких як час, кошти та людські ресурси.
– Визначення критеріїв використання: Визначте критеріїв використання веб-платформи, які допоможуть вам зрозуміти, наскільки зручно та просто використовувати платформу для користувачів. Наприклад, критерієм використання може бути кількість кліків, необхідних для виконання певної дії, таких як оформлення замовлення або пошук

товарів.

- Оцінка результатів: Після визначення метрик та критеріїв оцінки, потрібно відслідковувати та оцінювати результати розробки веб-платформи відповідно до цих критеріїв. Для цього можна використовувати спеціальні інструменти, такі як аналітичні звіти, опитування користувачів, тестування продуктивності та інші.
- Врахування результатів: На основі отриманих результатів, потрібно внести відповідні зміни та покращення веб-платформи. Цей етап дозволяє покращувати якість та ефективність веб-платформи, а також відповідати на потреби користувачів та бізнесу [10].
- Узагальнення: Формулювання критеріїв оцінки результатів розробки веб-платформи дозволяє визначити цілі та метрики проекту, оцінювати результати та покращувати якість та ефективність продукту. Важливо враховувати потреби користувачів та бізнесу при визначенні критеріїв оцінки та внесенні покращень в продукт.

РОЗДІЛ 2
РОЗРОБКА АРХІТЕКТУРИ ТА ФУНКЦІОНАЛУ ВЕБ-ПЛАТФОРМИ

2.1 Вибір технологій для розробки веб-платформи

Вибір технологій для розробки веб-платформи - це важливий етап у процесі розробки, який визначає якість, швидкість та ефективність розробки.

Основні критерії вибору технологій для розробки веб-платформи:

- Функціональність: Технології повинні дозволяти виконувати всі функції та завдання, які вимагається від веб-платформи.
- Масштабованість: Технології повинні дозволяти розширювати та масштабувати веб-платформу в майбутньому.
- Надійність та безпека: Технології повинні забезпечувати надійність та безпеку веб-платформи.
- Вартість: Вибір технологій повинен бути економічно обґрунтованим та відповідати бюджету проекту.
- Компетенції розробника: Розробник повинен мати досвід роботи з вибраними технологіями та мати необхідні навички для розробки веб-платформи.

При виборі технологій для розробки веб-платформи можна враховувати наступні фактори:

- Мови програмування: веб-платформу можна розробляти на різних мовах програмування, таких як PHP, JavaScript, Ruby, Python, та інші.
- Фреймворки: фреймворки допомагають зменшити час розробки та полегшують роботу розробників. Наприклад, фреймворки Laravel, Django, Ruby on Rails, React, Angular та інші.
- Бази даних: вибір бази даних залежить від вимог до швидкості, надійності та безпеки веб-платформи. Найпопулярніші бази даних для веб-розробки - MySQL, PostgreSQL, MongoDB, та інші.
- Хмарні сервіси: для забезпечення масштабованості та безпеки веб-платформи можна використовувати хмарні сервіси, такі як Amazon Web

Services (AWS), Microsoft Azure, Google Cloud Platform та інші.

Крім того, можна враховувати різні технічні аспекти, такі як швидкість та продуктивність, мобільність, SEO-оптимізація та інші [13].

При виборі технологій варто також звернути увагу на екосистему, підтримку та документацію. Наявність розширень та інструментів для розробки може допомогти зменшити час розробки та полегшити роботу розробників.

Остаточний вибір технологій для розробки веб-платформи залежить від конкретних вимог та цілей проекту, а також від умов бюджету та наявної компетенції розробників.

2.1.1 Верстка

Верстка є важливою складовою веб-розробки, що відповідає за створення дизайну та розміщення елементів на веб-сторінці. Основним завданням верстки є перетворення графічного дизайну веб-сторінки в HTML та CSS код [5]. Веб-сторінки складаються з блоків, що містять різноманітний контент, такий як текст, зображення, відео та інші елементи. Верстка відповідає за розміщення цих блоків та елементів на сторінці, створення їхнього вигляду та взаємодії з користувачем.

Підтримка SEO - верстка повинна бути оптимізована для пошукових систем, включаючи використання правильної структури тегів, метаданів та інших елементів, які допомагають зробити сторінку більш доступною та зрозумілою для роботів пошукових систем.

Для верстки веб-сторінок використовуються різні інструменти та технології, такі як HTML, CSS, JavaScript, бібліотеки та фреймворки. HTML використовується для створення структури та вмісту сторінки, CSS - для задання вигляду та стилізації елементів, а JavaScript - для створення взаємодії з користувачем та динамічного змісту.

Для забезпечення адаптивності та респонсивного дизайну використовуються такі технології, як media queries, flexbox, grid layout та інші. Media queries дозволяють змінювати вигляд елементів в залежності від розміру екрану, а flexbox та grid layout - дозволяють задавати гнучку та адаптивну розмітку сторінки.

Крім того, для більш швидкої та ефективної верстки використовуються різні бібліотеки та фреймворки, такі як Bootstrap, Foundation, Materialize та інші. Вони містять готові компоненти та стилі, які дозволяють швидко створювати зручний та привабливий дизайн сторінки (рисунок 2.1).

Верстка є важливою складовою веб-розробки, що забезпечує оптимальний вигляд та взаємодію веб-сторінки з користувачем. Для успішної верстки необхідно враховувати різні аспекти, такі як адаптивність, семантичність, швидкість завантаження, кросбраузерність та підтримка SEO.

```
<!DOCTYPE html>
<html>
<head>
  <title>Блоки стилів</title>
  <style type="text/css">
  .highlight {
    color: Blue;
    text-decoration: underline;
  }
  </style>
</head>
<body>
Це частина
<span class="highlight"> тексту </span>
з
<span class="highlight">виділеними </span>
елементами
<span class="highlight">в ньому</span>.
</body>
</html>
```

Рисунок 2.1 – Приклад верстки

2.1.2 Bootstrap

Bootstrap є одним з найпопулярніших фреймворків для фронтенд-розробки веб-сайтів та додатків. Він був розроблений командою Twitter з метою полегшення процесу розробки веб-сайтів та забезпечення їх консистентного вигляду та функціональності на різних пристроях та платформах.

Він забезпечує набір готових CSS та JavaScript компонентів, які можуть бути використані для швидкої та простої розробки веб-сайтів. Завдяки своїм вбудованим класам та стилям, Bootstrap дозволяє розробникам створювати

сучасні та зручні інтерфейси з мінімальним написанням власного CSS коду.

Також може бути використаний для розробки різних типів веб-сайтів та додатків, включаючи корпоративні веб-сайти, блоги, магазини, соціальні мережі та багато іншого. Він підтримує також інші технології, такі як SASS, LESS та JavaScript-фреймворки, що дозволяють розширити можливості та функціональність веб-сайту (рисунок 2.2).

Загалом, Bootstrap є потужним інструментом для розробки веб-сайтів та додатків, який дозволяє зменшити час та зусилля, необхідні для їх створення, та забезпечує високу консистентність та респонсивність веб-інтерфейсу.

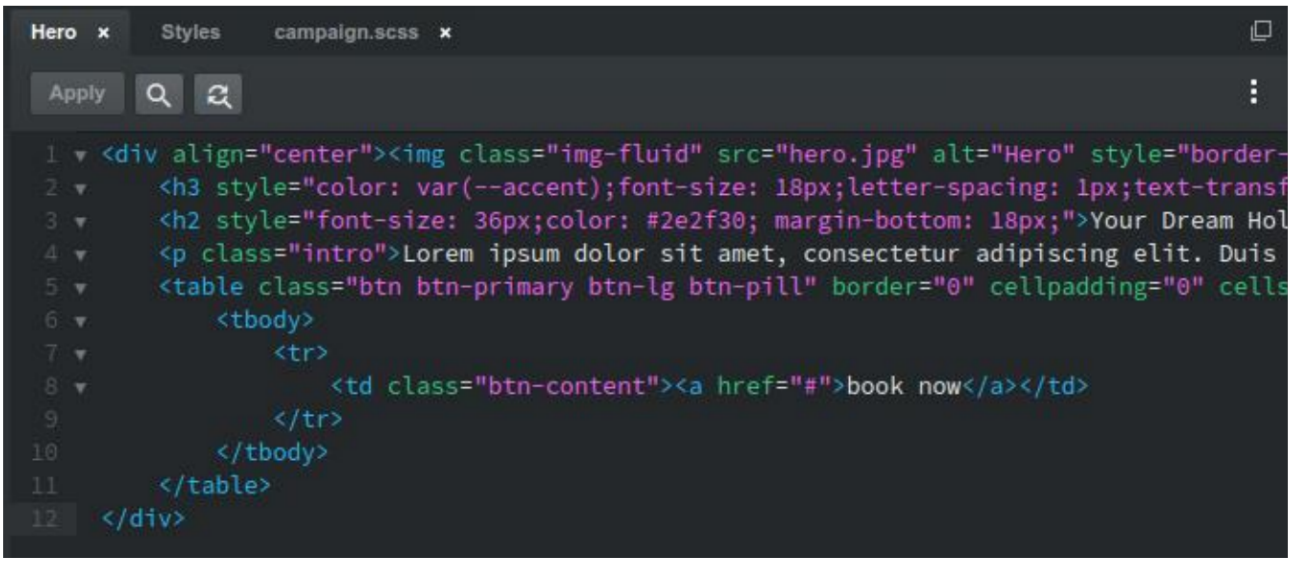

Рисунок 2.2 – Приклад коду Bootstrap

2.1.3 JavaScript

JavaScript є мовою програмування, яка використовується для створення інтерактивних веб-сайтів та додатків. Вона може бути використана для динамічного змінення стилів, зміни HTML-елементів, обробки форм та інших взаємодій з користувачем [3].

Будучи однією з трьох основних технологій веб-розробки разом з HTML та CSS. JavaScript дозволяє розробникам створювати багатофункціональні веб-додатки та інтерфейси з високою взаємодією користувача.

JavaScript є незамінною мовою для веб-розробки, її використовують для створення різноманітних додатків, включаючи веб-магазини, ігри, соціальні

мережі, бізнес-застосунки та інше. Основна роль JavaScript полягає в тому, щоб забезпечити інтерактивність та динамічність веб-сторінок [6].

Ця мова програмування може використовуватись як на стороні клієнта (client-side), так і на стороні сервера (server-side). На стороні клієнта JavaScript виконується в браузері та забезпечує взаємодію користувача з веб-сторінкою. На стороні сервера JavaScript використовується, наприклад, для створення веб-серверів або API.

JavaScript має велику кількість функцій та методів для роботи з DOM (Document Object Model), який представляє структуру HTML-документа. Завдяки цьому розробники можуть змінювати вміст сторінки, стилізацію, додавати ефекти та взаємодію з користувачем без перезавантаження сторінки.

Однією з найбільш важливих функцій JavaScript є обробка подій (event handling). Це означає, що JavaScript може реагувати на події, такі як клік миші, натискання клавіші, наведення курсора на елемент тощо (рисунок 2.3).

Існує велика кількість бібліотек та фреймворків на JavaScript, що дозволяють розробникам прискорювати процес створення веб-додатків та забезпечувати більшу функціональність. Деякі з найпопулярніших фреймворків на JavaScript включають React, Angular та Vue.js [11].

JavaScript є однією з найбільш популярних мов програмування у світі, тому знання JavaScript є дуже корисним для розробників веб-додатків та програмістів взагалі.

```
const car1 = {
  maker: 'Ford',
  model: 'Fiesta',
  drive() {
    console.log(`Driving a ${this.maker} ${this.model} car!`)
  }
}
const anotherCar = {
  maker: 'Audi',
  model: 'A4'
}
car1.drive.bind(anotherCar)()
//Driving a Audi A4 car!

const car2 = {
  maker: 'Ford',
  model: 'Fiesta'
}
const drive = function(kmh) {
  console.log(`Driving a ${this.maker} ${this.model} car at ${kmh} km/h!`)
}
drive.call(car2, 100)
//Driving a Ford Fiesta car at 100 km/h!
drive.apply(car2, [100])
//Driving a Ford Fiesta car at 100 km/h!
```

Рисунок 2.3 – Приклад коду JavaScript

2.1.3.1 React

React є одним з найпопулярніших JavaScript-фреймворків для створення користувацьких інтерфейсів (UI) веб-додатків. Він був розроблений компанією Facebook і випущений в 2013 році. React використовується для створення веб-додатків з високою взаємодією користувача (UI), де зміна стану сторінки призводить до миттєвих змін у відображенні [2].

Основна ідея React полягає в тому, щоб розбити користувацький інтерфейс на компоненти, кожен з яких відповідає за свою частину сторінки. Компоненти можуть бути вкладені один в одного, утворюючи ієрархію компонентів. Кожен компонент може мати свій власний стан (state) та властивості (props), які визначають його поведінку та відображення.

Основні переваги використання React:

- Швидкість та ефективність - React використовує віртуальний DOM (Document Object Model), який забезпечує швидкість та ефективність при зміні стану сторінки.
- Масштабованість - React дозволяє розбити веб-додаток на компоненти, що забезпечує масштабованість та легкість розробки.
- Компонентна структура - React дозволяє відокремити логіку та відображення, що забезпечує більшу контроль над сторінкою та полегшує розробку та підтримку коду.
- Багатофункціональність - React дозволяє легко додавати додаткові функції та бібліотеки для розширення функціональності веб-додатків.
- Підтримка спільноти - React має велику та активну спільноту, що забезпечує швидку підтримку та розвиток фреймворку, а також багато доступних ресурсів для навчання та підтримки.

React можна використовувати як для розробки клієнтської, так і для серверної частини веб-додатків. Для клієнтської частини React може працювати з будь-яким бекендом, включаючи RESTful API та GraphQL. Для серверної частини React може використовуватись у поєднанні з Node.js [16].

У підсумку, React є потужним та ефективним інструментом для розробки веб-додатків з високою взаємодією користувача. Він дозволяє створювати

масштабовані та ефективні веб-додатки, що забезпечують більшу контроль та легкість у розробці та підтримці коду. Багата спільнота та наявність багатьох ресурсів для навчання та підтримки роблять React одним з найкращих виборів для розробки сучасних веб-додатків.

2.1.3.2 Angular

Angular - це фреймворк для створення веб-додатків, який забезпечує простоту та швидкість розробки. Його основні переваги полягають у компонентній структурі, модульності, вбудованій системі тестування та підтримці різних платформ [12].

Angular дозволяє розробляти веб-додатки, використовуючи TypeScript - строго типізований діалект JavaScript, який дозволяє розробникам створювати більш надійний та читабельний код. Крім того, Angular надає широкі можливості для розширення функціональності додатків за допомогою додаткових бібліотек та фреймворків.

Цей фреймворк має велику та активну спільноту розробників, яка забезпечує швидку підтримку та розвиток фреймворку. Також веб-додатки, створені з використанням Angular, можуть бути легко масштабовані та тестовані, що забезпечує високу якість коду та швидкий розвиток проекту.

2.1.3.3 Vue.js

Vue.js - це прогресивний фреймворк для розробки веб-інтерфейсів та односторінкових додатків. Vue.js був розроблений Еваном Ю в 2014 році і з тих пір став одним з найпопулярніших фреймворків у світі веб-розробки.

Однією з основних переваг Vue.js є його простота використання та легкість навчання, що дозволяє швидко почати роботу з ним. Крім того, Vue.js забезпечує широкі можливості для розробки веб-додатків з високою взаємодією користувача та підтримує відокремлену компонентну структуру [15].

Основні переваги використання Vue.js:

- Простота використання та навчання: Vue.js має простий та легкий для зрозуміння API, що дозволяє швидко почати роботу з ним. Крім того, у Vue.js є велика кількість документації та прикладів використання.
- Висока продуктивність: Vue.js забезпечує високу продуктивність за

рахунок відокремленої компонентної структури та ефективного механізму оновлення даних.

- Гнучкість та розширюваність: Vue.js дозволяє розширювати функціональність веб-додатків за допомогою різних бібліотек та модулів. Крім того, Vue.js має велику кількість плагінів, що спрощують роботу з ним.
- Висока взаємодія з користувачем: Vue.js забезпечує високу взаємодію користувача з веб-додатком за допомогою різних директив та додаткових бібліотек.
- Адаптивність: Vue.js підтримує різні платформи та браузери, що дозволяє створювати веб-додатки, які можна запускати на різних пристроях та в різних браузерах.

Одним з ключових понять у Vue.js є компоненти. Компоненти - це окремі частини веб-інтерфейсу, які можна перевикористовувати у різних частинах додатку. Компоненти можна створювати як класи або функції, що повертають об'єкт з описом компонента. Крім того, Vue.js має вбудований механізм зв'язування даних (data binding), що дозволяє автоматично оновлювати веб-інтерфейс при зміні даних у додатку.

Vue.js також має велику кількість різноманітних додаткових бібліотек та розширень, які дозволяють спрощувати роботу з фреймворком та розширювати його функціональність. Наприклад, Vuex - це додаткова бібліотека для керування станом додатку, а Vue Router - це розширення для створення односторінкових додатків зі зручною навігацією.

У загальному, Vue.js - це потужний та гнучкий фреймворк для розробки веб-інтерфейсів та односторінкових додатків, який дозволяє швидко та ефективно створювати високоякісні веб-додатки.

2.2 Опис архітектури веб-платформи

Опис архітектури веб-платформи - це план, який визначає, як різні компоненти системи будуть взаємодіяти між собою та як буде організована

логіка роботи системи.

Архітектура веб-платформи має бути добре спроектованою та організованою, щоб забезпечити ефективність, швидкість та безпеку роботи системи.

Основні компоненти архітектури веб-платформи:

- Клієнтська частина - це частина системи, яка взаємодіє з користувачем. Це може бути веб-браузер або мобільний додаток.
- Серверна частина - це частина системи, яка забезпечує логіку роботи та взаємодіє з базою даних та іншими сервісами. Це може бути написано на різних мовах програмування та використовувати різні фреймворки.
- База даних - це компонент, який забезпечує зберігання даних та дозволяє виконувати операції з даними. Вибір бази даних залежить від вимог до швидкості, надійності та безпеки веб-платформи.
- API (Application Programming Interface) - це інтерфейс, який дозволяє різним компонентам системи взаємодіяти між собою та обмінюватися даними.
- Хмарні сервіси - це додаткові компоненти, які можуть забезпечувати додаткову функціональність, таку як зберігання файлів, електронні листи та інші.

При проектуванні архітектури веб-платформи необхідно враховувати потреби та вимоги до системи, забезпечити її масштабованість та легкість розширення, а також забезпечити безпеку та захист даних. Крім того, слід враховувати фактори, такі як вартість розробки, рівень складності та доступність ресурсів.

Архітектура веб-платформи може бути побудована на різних принципах, наприклад, на основі моделі MVC (Model-View-Controller), REST (Representational State Transfer) або SPA (Single Page Application). Кожна з цих моделей має свої переваги та недоліки, тому вибір залежить від конкретних потреб та вимог до системи [20].

Загалом, архітектура веб-платформи є ключовим фактором успіху будь-якої веб-системи, оскільки вона визначає, як система буде працювати та яким

буде її функціонал. Вірно побудована архітектура забезпечує надійність, ефективність та безпеку роботи веб-платформи, а також сприяє її успішному розвитку та масштабуванню в майбутньому.

2.2.1 Архітектурні рішення та компоненти

Архітектурні рішення та компоненти веб-платформи визначають, як система буде функціонувати та які компоненти будуть включені в її склад.

Основні архітектурні рішення та компоненти веб-платформи:

- Монолітна архітектура - це коли вся логіка роботи системи знаходиться в одному монолітному додатку. Така архітектура є простою у розробці та тестуванні, але може бути складною у підтримці та масштабуванні.
- Розподілена архітектура - це коли логіка роботи системи розподілена між кількома компонентами, що можуть працювати на різних серверах. Така архітектура забезпечує більшу масштабованість та зменшує ризик відмови системи, але може бути складною у розробці та відлагодженні.
- Мікросервісна архітектура - це коли система складається з набору невеликих незалежних сервісів, які можуть працювати на різних серверах. Кожен сервіс має свою власну базу даних та API. Така архітектура забезпечує більшу гнучкість та легкість розширення системи, але може бути складною у відлагодженні та тестуванні.
- Хмарна архітектура - це коли система базується на хмарних технологіях, таких як Amazon Web Services або Microsoft Azure. Така архітектура забезпечує більшу масштабованість та зменшує ризик відмови системи, оскільки компоненти можуть автоматично масштабуватися залежно від навантаження.

Крім того, до компонентів веб-платформи можуть входити різноманітні сервіси, такі як бази даних, сервери, API, системи кешування, моніторингу та логування, системи автентифікації та авторизації, інтеграція зі сторонніми сервісами та бібліотеками, клієнтська частина (фронтенд) та серверна частина (бекенд) та інші [19]. Кожен компонент відповідає за певну функціональність

системи та може бути розглянутий окремо для розробки, тестування та підтримки.

2.2.2 Принципи взаємодії компонентів

Принципи взаємодії компонентів в системі є важливим аспектом при проектуванні та розробці веб-платформи. Взаємодія між компонентами може бути складною, особливо у великих системах з багатьма компонентами, тому необхідно розробляти систему з урахуванням таких принципів:

- Принцип єдиного обов'язку (Single Responsibility Principle, SRP) - кожен компонент системи повинен мати лише один обов'язок. Це допомагає забезпечити простоту та чистоту коду, а також зменшує кількість залежностей між компонентами.
- Принцип відкритості-закритості (Open-Closed Principle, OCP) - компоненти повинні бути відкриті для розширення, але закриті для модифікації. Це означає, що коли необхідно додати новий функціонал до системи, слід це зробити шляхом додавання нових компонентів, а не модифікування існуючих.
- Принцип заміщення Лісков (Liskov Substitution Principle, LSP) - компоненти системи повинні бути замінюваними між собою без зміни правильності роботи програми. Це означає, що класи повинні бути замінювані їх підкласами без впливу на роботу програми.
- Принцип інтерфейсу (Interface Segregation Principle, ISP) - клієнт повинен залежати лише від тих інтерфейсів, які він використовує, а не від усіх доступних в системі інтерфейсів. Це дозволяє зменшити залежності між компонентами та забезпечити більшу гнучкість системи.
- Принцип інверсії залежності (Dependency Inversion Principle, DIP) - компоненти системи повинні залежати від абстракцій, а не від конкретних реалізацій. Це дозволяє зменшити залежності між компонентами та сприяє більшій гнучкості системи при зміні реалізації компонентів.

Ці принципи взаємодії компонентів допомагають створити систему, яка є ефективною, гнучкою та легко зрозумілою для розробників. Крім того, вони

сприяють зменшенню ризику появи помилок та спрощують процес тестування системи. Тому, при проектуванні та розробці веб-платформи, необхідно враховувати ці принципи та дотримуватися їх під час всього процесу [22].

2.2.3. Опис інтерфейсів та функціоналу користувачів

Інтерфейси користувачів (UI) включають в себе всі елементи, якими користувач взаємодіє з системою, такі як кнопки, текстові поля, меню та інші. Опис інтерфейсів може включати в себе опис елементів UI та їх функціональні можливості, наприклад, те, що кнопка виконує дію або відображає інформацію.

Кожен елемент інтерфейсу повинен мати чіткий та доступний для користувача опис того, що він робить та як взаємодія з ним відбувається. Опис інтерфейсу також повинен бути відповідним для цільової аудиторії, що використовує додаток. Наприклад, якщо додаток призначений для дітей, опис інтерфейсу повинен бути більш простим та зрозумілим для них.

Крім опису інтерфейсів, також важливо описати функціонал користувачів. Функціонал користувачів визначає, які дії можуть виконувати користувачі в системі, наприклад, додавати нові записи, видаляти їх, редагувати та інші. Для кожної дії необхідно описати, як вона виконується та які обмеження встановлені на її виконання [18].

Опис інтерфейсів та функціоналу користувачів допомагає розробникам дотримуватися вимог користувачів та забезпечити належний рівень функціональності та зручності використання системи. Крім того, такий опис може бути використаний для тестування системи та навчання користувачів її використанню.

РОЗДІЛ 3
ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ ВЕБ-САЙТА ТА ВЕБ-ДОДАТКА

3.1 Шапка та підвал веб-сайта

Для сайту була розроблена шапка яку зробили адаптивною, за допомогою фреймворка Bootstrap. В шапку входить 4 пункти такі як головна сторінка, про нас, меню та галерея. Також в шапку було додано логотип кав'ярні (рисунок 3.1).

Рисунок 3.1 – Шапка веб-сайту

В підвал сайту входять основні контакти та місцезнаходження кав'ярні (рисунок 3.2).

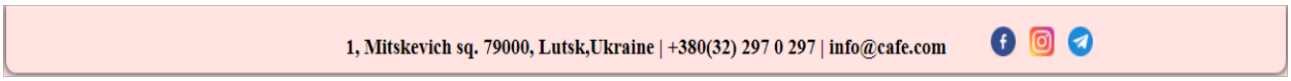

Рисунок 3.2 – Підвал веб-сайту

3.2 Головна сторінка

На головну сторінку добавлено невеликий опис кав'ярні під яким є посилання на сторінку «про нас» (рисунок 3.3).

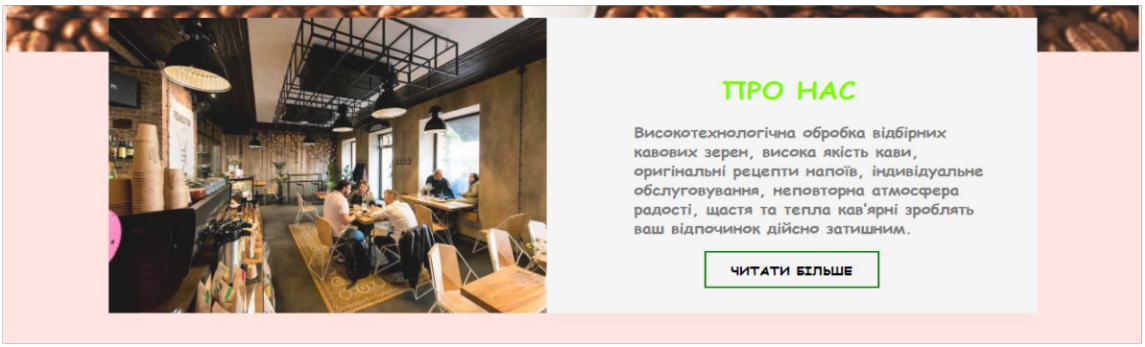

Рисунок 3.3 – Невеликий опис кав'ярні

Також щоб відчути атмосферу кав’ярні було додано світлину з декількома фотографіями та посиланням на «галерею» кав’ярні (рисунок 3.4).

Рисунок 3.4 – Світлина кав’ярні

Щоб показати якість та професіоналізм кав’ярні на сайт було додано нагороди (рисунок 3.5).

Рисунок 3.5 – Нагороди кав’ярні

3.3 Сторінка про нас

На цій сторінці детально описана кав’ярня. Її особливості та навички персоналу (рисунок 3.6).

Рисунок 3.6 – Опис кав’ярні

3.4 Сторінка меню

Сторінка меню показує гарячі, холодні напої та десерти які можна замовити в кав’ярні, також вказана ціна і вага десертів та напоїв (рисунок 3.7, 3.8, 3,9).

Рисунок 3.7 – Гарячі напої

Рисунок 3.8 – Холодні напої

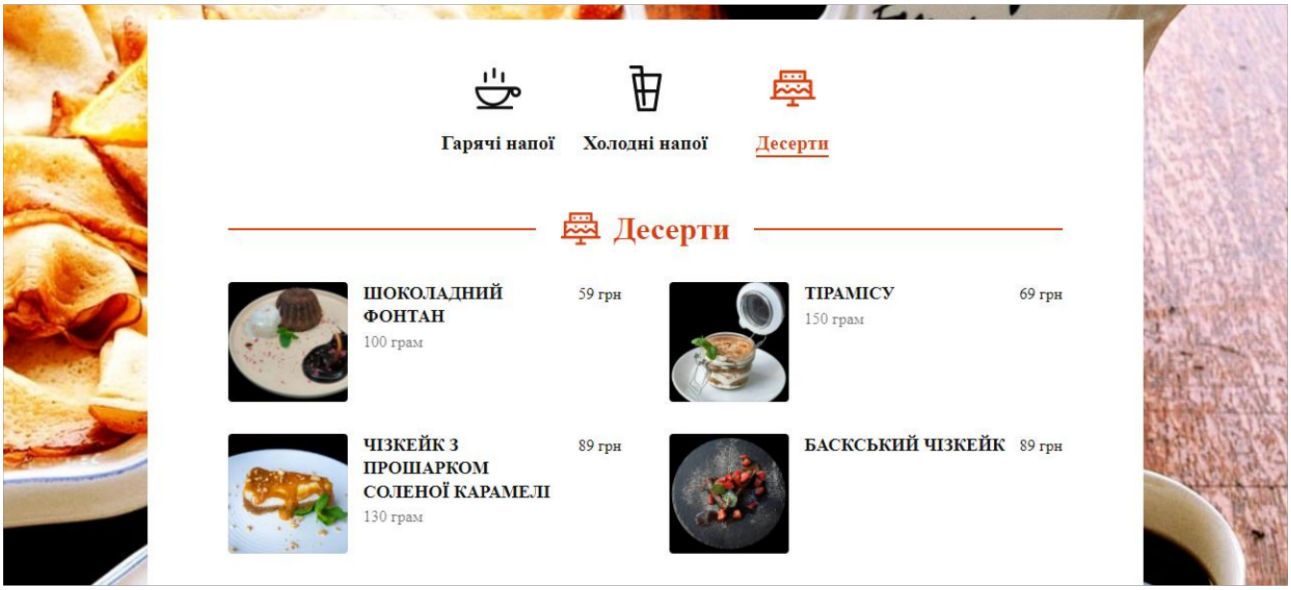

Рисунок 3.9 – Десерти

3.5 Сторінка галерея

Для цієї сторінки був розроблений слайдер який змінює фотографії кожні 7 секунд (рисунок 3.10).

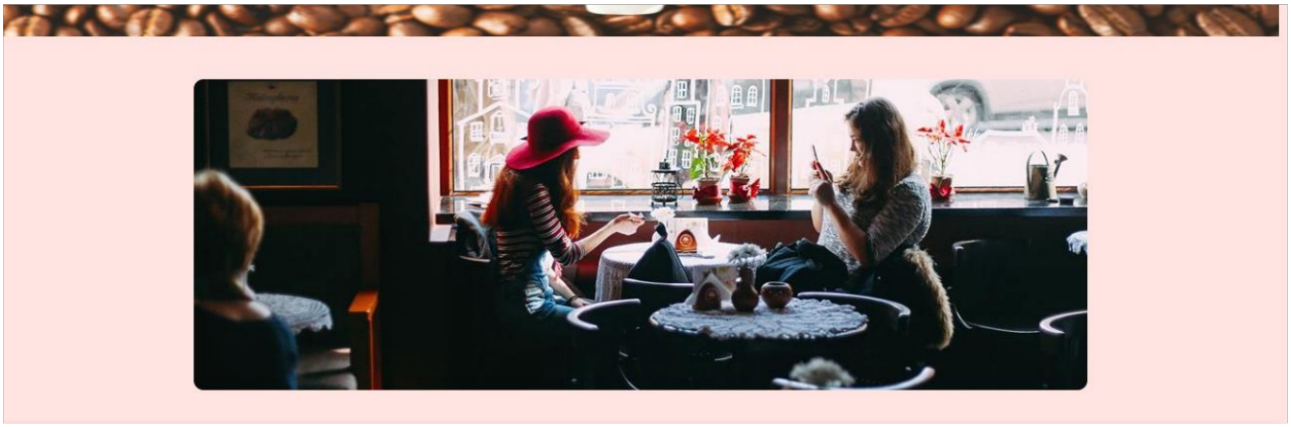

Рисунок 3.10 – Слайдер

3.6 QR-меню

QR-меню для кав'ярні - це цифрова версія традиційного меню, доступ до якої відкривається шляхом сканування QR-коду за допомогою смартфона. QR-коди розташовують на столах або на видному місці в кав'ярні, щоб відвідувачі могли легко їх сканувати. Після сканування QR-коду, відвідувачі переходять на веб-сторінку з меню (рисунок 3.11).

Рисунок 3.11 – QR-код

Просканувавши QR-код нам відкривається сторінка з меню на якій можна подивитися і замовити різні напої та десерти. Це допоможе вам не чекати офіціанта а самому подивитися те що пропунує вам кав’ярня (рисунок 3.12).

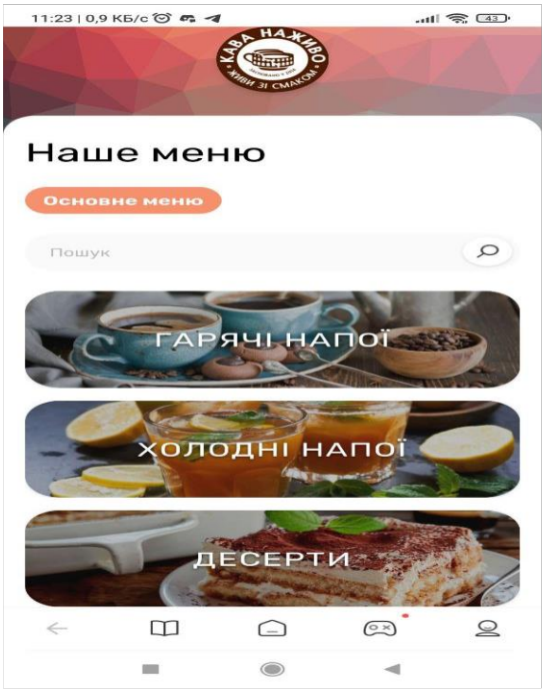

Рисунок 3.12 – QR-меню

Заглянувши на сторінку десерти ми побачимо привабливий зовнішній вигляд десертів їх ціну та вагу. Так само зроблені гарячі і холодні напої (рисунок 3.13).

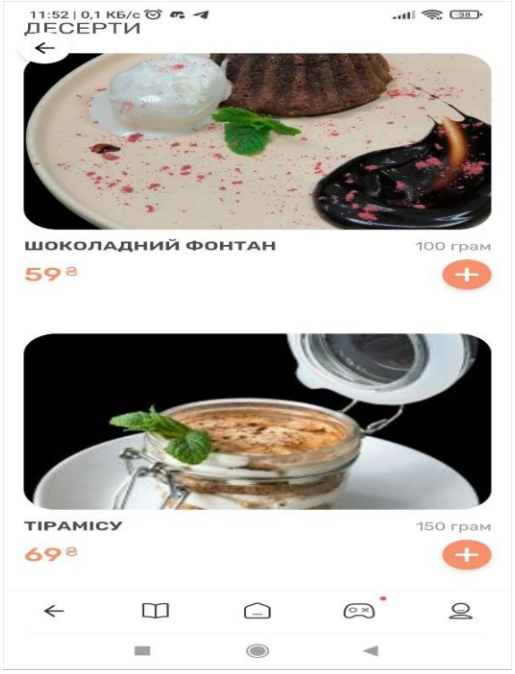

Рисунок 3.13 – Вигляд страв в меню

Також в меню присутній пошук який допоможе знайти те що захочеться наприклад американо (рисунок 3.14).

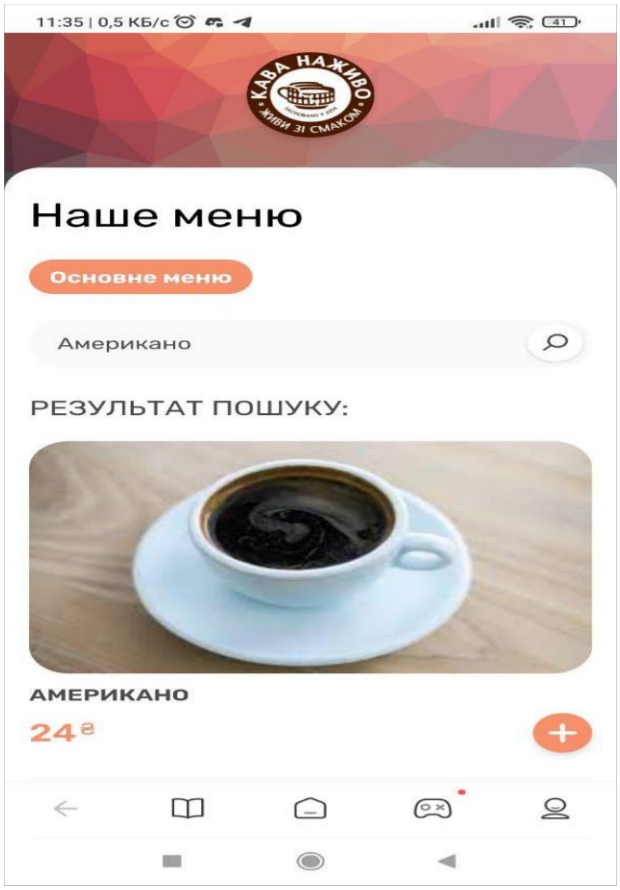

Рисунок 3.14 – Пошук в меню

Веб-платформа стала потужним маркетинговим інструментом. Кав'ярня змогла оголошувати спеціальні акції, нові продукти та інші події безпосередньо своїм клієнтам.

З допомогою цифрових інструментів кав'ярня могла збирати детальні дані про поведінку клієнтів, що дозволило зрозуміти їх потреби глибше і покращити сервіс.

У результаті впровадження веб-платформи було виявлено значне зростання продажів, збільшився рівень задоволення клієнтів і загальна продуктивність бізнесу.

ВИСНОВКИ

У даній кваліфікаційній роботі проведено аналіз актуальності та визначено вимоги до веб-платформи для підвищення ефективності ведення бізнесу за допомогою забезпечення контролю та аналізу ключових показників діяльності.; програмного коду, фреймворків та бібліотек і дизайну для сайту

В процесі розробки було проаналізовано технології, які використовуються для веб-платформ, використано сучасні технології веб-розробки, такі як верстка на HTML/CSS та мова програмування JavaScript.

Розроблено веб-платформу у відповідності до поставлених вимог та додаток, що можуть бути корисним для власників бізнесу, які прагнуть підвищити ефективність своєї діяльності, полегшити процес управління та забезпечити більш ефективне використання ресурсів. Дана платформа має потенціал для подальшого розвитку та вдосконалення залежно від потреб користувачів.

В роботі була розроблена веб-платформа, яка має на меті підвищення ефективності ведення бізнесу. Дана платформа включає в себе сайт та додаток, що дозволяють користувачам отримувати доступ до різноманітних функцій, таких як створення та відстеження замовлень, керування запасами та управління персоналом.

Додаток був протестований та підтверджено його функціональність та відповідність вимогам користувачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ACM Digital Library URL: https://dl.acm.org (дата звернення: 27.02.2023).

2. Bertolli M. React Design Patterns and Best Practices: Build easy to scale modular applications using the most powerful components and design patterns. Packt Publishing, 2019. 320 p.

3. Code Institute. URL: https://codeinstitute.net/global/blog/advantages-of-javascript/ (дата звернення: 18.05.2023).

4. Computer Science Review URL: https://www.sciencedirect.com/journal/computer-science-review (дата звернення: 13.04.2023).

5. CSS-Tricks URL: https://css-tricks.com (дата звернення: 17.04.2023).

6. Eloquent JavaScript URL: https://eloquentjavascript.net (дата звернення: 04.05.2023).

7. Google Developers URL: https://developers.google.com/?hl=ru (дата звернення: 12.02.2023).

8. IEEE Xplore Digital Library URL: https://ieeexplore.ieee.org/Xplore/home.jsp (дата звернення: 24.02.2023).

9. International Journal of Web Services Research (JWSR) URL: https://www.igi-global.com/journal/international-journal-web-services-research/1075 (дата звернення: 12.03.2023).

10. Journal of Web Engineering URL: https://www.worldscientific.com/worldscinet/jwe (дата звернення: 21.04.2023).

11. Learn to code, change your career. Code Institute. URL: https://codeinstitute.net/global/blog/advantages-of-javascript/ (дата звернення: 18.05.2023).

12. MDN Web Docs URL: https://developer.mozilla.org/ru/ (дата звернення: 15.02.2023).

13. Microsoft Developer Network URL: https://developer.microsoft.com/en-us/ (дата звернення: 22.03.2023).

14. Miletsky J.I. Principles of Internet Marketing: New Tools and Methods for Web Developers : Course Technology Press Boston, MA, United States, 2020. 480 p.

15. Mozilla Developer Network - Progressive Web Apps URL: https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps (дата звернення: 16.04.2023).

16. Sidelnikov G. React.js Book: Learning React JavaScript Library From Scratch. River Tigris LLC, 2016. 350 p.

17. Simplilearn | online courses - bootcamp & certification platform. Simplilearn.com. URL: https://www.simplilearn.com (дата звернення: 13.05.2023).

18. Springer: Computer Science URL: https://www.springer.com/gp/computer-science (дата звернення: 07.04.2023).

19. The rumpus.net. The Rumpus.net. URL: https://therumpus.net/ (дата звернення: 13.05.2023).

20. The Service Worker Lifecycle URL: https://web.dev/service-worker-lifecycle/ (дата звернення: 16.04.2023).

21. W3C Standards URL: https://www.w3.org/standards/ (дата звернення: 15.02.2023).

22. Web Standards: The What, The Why, And The How URL: https://www.smashingmagazine.com/2019/01/web-standards-guide/ (дата звернення: 10.05.2023).

ДОДАТКИ

ДОДАТОК А

Код файлу golovna.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset=«UTF-8»>
<link rel=«stylesheet» href=«style.css» />
<title>sait</title>
</head>
<body >
<header >
<nav class=«top-menu» « >
  <a class=«navbar-logo» href=«‹‹‹><img src=«./img\kava_nazhyvo_otzyv.png» width=«75»
height=«55» ></a>
  <ul class=«menu-main»>
    <li><a href=«‹‹‹>ГОЛОВНА</a></li>
    <li><a href=«‹‹‹>ПРО НАС</a></li>
    <li><a href=«‹‹‹>МЕНЮ</a></li>
    <li><a href=«‹‹‹>ГАЛЕРЕЯ</a></li>
  </ul>
</nav>
</header>
<div class=«golov»>
  <img src=«./img\coffe.jpeg» width=«100%» height=«600» >
  <div class=«text»>
    <h1>Найкраща кав’ярня у Луцьку!</h1>
    <p><h1>МИ ВАС ЛЮБИМО!</h1></p>
  </div>
</div>
<div class=«kart_text»>
<img src=«./img\kavka.jpg» width=«500» height=«400» >
<div class=«per_text»>
<p><h1>ПРО НАС</h1></p>
<h3><p>Високотехнологічна обробка відбірних кавових зерен, висока якість
   кави, оригінальні рецепти напоїв, індивідуальне обслуговування,
       неповторна атмосфера радості, щастя та тепла кав’ярні зроблять
              ваш відпочинок дійсно затишним.</p></h3>
<p><h4>ЧИТАТИ БІЛЬШЕ</h4></p>
</div>
</div>
<div class=«menush»>
<p><h1>Наші</h1></p>
<p><h1>СВІТЛИНИ</h1></p>
<img src=«./img\svit3.jpg» width=«300» height=«200» >
<img src=«./img\svit1.jpg» width=«300» height=«200» >
<img src=«./img\svit2.jpg» width=«300» height=«200» >
<img src=«./img\svit4.jpg» width=«300» height=«200» >
<img src=«./img\svit5.jpg» width=«300» height=«200» >
<img src=«./img\svit6.jpg» width=«300» height=«200» >
<p><h4>ДИВИТИСЯ БІЛЬШЕ</h4></p>
</div>
<div class=«nagi»>
```

```
<img src=«./img\coffe.jpeg» width=«100%» height=«600» >
<div class=«text_nagi»>
<p><font size=«10»> Наші </font></p>
<p><font size=«15»>НАГОРОДИ</font></p>
</div>
<div class=«photo_nagi»>
<img src=«./img\nagi_1.jpg» width=«250px» height=«400px» >
<img src=«./img\naggg.jpg» width=«250px» height=«400px» >
</div>
</div>
<footer class=«foot»>
<div class=«foo_text»>
<p><h3>1, Mitskevich sq. 79000, Lutsk,Ukraine | +380(32) 297 0 297 | info@cafe.com</h3></p>
</div>
<div class=«seti»>
<a href=««><img src=«./img\facebook.png» width=«35px» height=«35px»></a>
<a href=««><img src=«./img\insta.png» width=«35px» height=«35px»></a>
<a href=««><img src=«./img\telega.png» width=«35px» height=«35px»></a>
</div>
</footer>
<script src=«script.js»>
</script>
</body>
</html>
```

ДОДАТОК В

Код файлу pro_nas.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset=«UTF-8»>
<link rel=«stylesheet» href=«style.css» />
<title>sait</title>
</head>
<body >
<header >
<nav class=«top-menu» « >
  <a class=«navbar-logo» href=«‹‹‹><img src=«./img\kava_nazhyvo_otzyv.png» width=«75»
height=«55» ></a>
  <ul class=«menu-main»>
    <li><a href=«‹‹‹>ГОЛОВНА</a></li>
    <li><a href=«‹‹‹>ПРО НАС</a></li>
    <li><a href=«‹‹‹>МЕНЮ</a></li>
    <li><a href=«‹‹‹>ГАЛЕРЕЯ</a></li>
  </ul>
</nav>
</header>
<div class=«golov»>
  <img src=«./img\coffe.jpeg» width=«100%» height=«450» >
  <div class=«text_pro»>
    <h1>Найкраща кав’ярня у Луцьку!</h1>
    <p><h1>МИ ВАС ЛЮБИМО!</h1></p>
  </div>
</div>
<div class=«kart_text_pro»>
<img src=«./img\kavka_2.jpg» width=«500» height=«400» >
<div class=«per_text_pro»>
<p><h1>ПРО НАС</h1></p>
<p><h3>Найбільша наша особливість - це кава. Ми пропонуємо великий вибір сортів кави ,
які ви можете спробувати на місці або купити з собою. Кожна порція кави готується з
особливою увагою і любов'ю.
Крім кави, у меню є великий вибір чаю, соків, десертів. Особливу увагу заслуговує власна
випічка кав'ярні - млинці з домашнім сиром та ягодою, шоколадний фонтан.
Персонал кав'ярні завжди привітний та готовий допомогти з вибором напою або закуски.
Вони мають глибокі знання про каву та вміють порекомендувати щось особливе.
Наша кав'ярня  - це ідеальне місце для зустрічей з друзями, роботи або просто
відпочинку.</h3></p>
</div>
</div>
<footer class=«foot»>
<div class=«foo_text»>
<p><h3>1, Mitskevich sq. 79000, Lutsk,Ukraine | +380(32) 297 0 297 | info@cafe.com</h3></p>
</div>
<div class=«seti»>
```

```
<a href=«‹›><img src=«./img\facebook.png» width=«35px» height=«35px»></a>
<a href=«‹›><img src=«./img\insta.png» width=«35px» height=«35px»></a>
<a href=«‹›><img src=«./img\telega.png» width=«35px» height=«35px»></a>
</div>
</footer>
</body>
</html>
```

ДОДАТОК С

Код файлу menu.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset=«UTF-8»>
<link rel=«stylesheet» href=«style.css» />
<title>sait</title>
</head>
<body >
<header >
<nav class=«top-menu» « >
  <a class=«navbar-logo» href=««‹><img src=«./img\kava_nazhyvo_otzyv.png» width=«75»
height=«55» ></a>
  <ul class=«menu-main»>
    <li><a href=««‹>ГОЛОВНА</a></li>
    <li><a href=««‹>ПРО НАС</a></li>
    <li><a href=««‹>МЕНЮ</a></li>
    <li><a href=««‹>ГАЛЕРЕЯ</a></li>
  </ul>
</nav>
</header>
<div class=«golov»>
  <img src=«./img\coffe.jpeg» width=«100%» height=«450» >
  <div class=«text_pro»>
    <h1>Найкраща кав’ярня у Луцьку!</h1>
    <p><h1>МИ ВАС ЛЮБИМО!</h1></p>
  </div>
</div>
<div class=«vse_menu»>
<img src=«./img\fon_menu.jpg» width=«100%» height=«1050» >

<div class=«menu_menu»>
<div class=«vib_menu»>
<script src=«https://static.elfsight.com/platform/platform.js» data-use-service-core defer></script>
<div class=«elfsight-app-40c1cda2-a05c-47d8-aa86-c3f280868936»></div>
  </div>
</div>
</div>
<footer class=«foot»>
<div class=«foo_text»>
<p><h3>1, Mitskevich sq. 79000, Lviv,Ukraine | +380(32) 297 0 297 |
info@theglorycafe.com</h3></p>
</div>
<div class=«seti»>
<a href=««‹><img src=«./img\facebook.png» width=«35px» height=«35px»></a>
<a href=««‹><img src=«./img\insta.png» width=«35px» height=«35px»></a>
<a href=««‹><img src=«./img\telega.png» width=«35px» height=«35px»></a>
</div>
</footer>
```

ДОДАТОК D

Код файлу galerei.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset=«UTF-8»>
<link rel=«stylesheet» href=«style.css» />
<title>sait</title>
</head>
<body >
<header >
<nav class=«top-menu» « >
  <a class=«navbar-logo» href=«‹‹‹><img src=«./img\kava_nazhyvo_otzyv.png» width=«75»
height=«55» ></a>
  <ul class=«menu-main»>
    <li><a href=«‹‹‹>ГОЛОВНА</a></li>
    <li><a href=«‹‹‹>ПРО НАС</a></li>
    <li><a href=«‹‹‹>МЕНЮ</a></li>
    <li><a href=«‹‹‹>ГАЛЕРЕЯ</a></li>
  </ul>
</nav>
</header>
<div class=«golov»>
  <img src=«./img\coffe.jpeg» width=«100%» height=«450» >
  <div class=«text_pro»>
    <h1>Найкраща кав’ярня у Луцьку!</h1>
    <p><h1>МИ ВАС ЛЮБИМО!</h1></p>
  </div>
</div>
<div class=«slider»>
    <div class=«slide» style=«background-image: url('img/svita1.png');»></div>
    <div class=«slide» style=«background-image: url('img/svita2.png');»></div>
    <div class=«slide» style=«background-image: url('img/svita3.png');»></div>
    <div class=«slide» style=«background-image: url('img/svita4.png');»></div>
    <div class=«slide» style=«background-image: url('img/svit5.jpg');»></div>
    <div class=«slide» style=«background-image: url('img/svit6.jpg');»></div>
</div>
<footer class=«foot»>
<div class=«foo_text»>
<p><h3>1, Mitskevich sq. 79000, Lviv,Ukraine | +380(32) 297 0 297 |
info@theglorycafe.com</h3></p>
</div>
<div class=«seti»>
<a href=«‹‹‹><img src=«./img\facebook.png» width=«35px» height=«35px»></a>
<a href=«‹‹‹><img src=«./img\insta.png» width=«35px» height=«35px»></a>
<a href=«‹‹‹><img src=«./img\telega.png» width=«35px» height=«35px»></a>
</div>
</footer>
```

```
<script>
let slides = Array.from(document.querySelectorAll('.slide'));
let currentSlide = 0;
setInterval(() => {
    slides[currentSlide].style.opacity = 0;
    currentSlide = currentSlide < slides.length - 1 ? currentSlide + 1 : 0;
    slides[currentSlide].style.opacity = 1;
}, 3000);
</script>
</body>
</html>
```

ДОДАТОК Е

Код файлу style.css

```
body{
	background-color:MistyRose;
}
.top-menu {
  background: rgba(255,255,255,.5);
  box-shadow: 0 0 7px rgba(0,0,0,.3),0 0 7px rgba(0,0,0,.3),0 0 7px rgba(0,0,0,.3),0 0 7px
rgba(0,0,0,.3);
  padding: 5px;
background-color:MistyRose;
border-top-left-radius: 10px;
border-top-right-radius: 10px;
}
.top-menu:after {
  content: «««;
  display: table;
  clear: both;
}
.navbar-logo {display: inline-block;
}
.menu-main {
  list-style: none;
  margin: 0;
  padding: 0;
  float: right;
}
.menu-main li {display: inline-block;}
.menu-main a {
  text-decoration: none;
  display: block;
  position: relative;
  line-height: 61px;
  padding-left: 20px;
  font-size: 18px;
  letter-spacing: 2px;
  font-family: 'Arimo', sans-serif;
  font-weight: bold;
  color: black;
  transition:.3s linear;
}
.menu-main a:before {
  content: «««;
  width: 9px;
  height: 9px;
  background: #F73E24;
  position: absolute;
  left: 50%;
  transform: rotate(45deg) translateX(6.5px);
  opacity: 0;
  transition: .3s linear;
}
```

```
.menu-main a:hover:before {opacity: 1;}
@media (max-width: 660px) {
.menu-main {
  float: none;
  padding-top: 20px;
}
.top-menu {
  text-align: center;
  padding: 20px 0 0 0;
}
.menu-main a {padding: 0 10px;}
.menu-main a:before {transform: rotate(45deg) translateX(-6px);}
}
@media (max-width: 600px) {
.menu-main li {display: block;}
}
.golov {
  position: relative;
}
.golov{
        opacity:0.9;
}
.golov h1{
        font-family: cursive;
        color:LawnGreen;
}
.text {
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
  border: 4px solid green;
  padding: 10px;
  text-align:center;
}
.text h2{
        text-align:center;
        color:pink;
}
.foot{
         background: rgba(255,255,255,.5);
box-shadow: 0 0 7px rgba(0,0,0,.3),0 0 7px rgba(0,0,0,.3),0 0 7px rgba(0,0,0,.3),0 0 7px
rgba(0,0,0,.3);
  padding: 5px;
background-color:MistyRose;
border-bottom-left-radius: 10px;
border-bottom-right-radius: 10px;
position: relative;
}
.foo_text{
        text-align:center;
}
.seti{
        margin-left:985px;
```

```
        margin-top:-52px;
}
.seti img{
        padding: 5px;
}
.kart_text{
        border: 0px solid PaleGreen;
        background-color:WhiteSmoke;
        margin-top:150px;
        position: relative;
transform: translate(-50%, -50%);
height:400px;
  width:1060px;
  left:50%;
}
.kart_text img{
        opacity:0.9;
}
.per_text{
        margin-left:600px;
        position: absolute;
        margin-top:-350px;
}
.per_text h4{
        border: 2px solid Green;
        width:200px;
        height:50px;
        text-align:center;
        line-height: 50px;
        margin-top:10px;
        margin-left:80px;
        font-family: cursive;
}
.per_text h4:hover{
        background-color:Green;
        opacity:0.7;
}
.per_text h1{
        font-family: cursive;
        margin-left:100px;
        color:LawnGreen;
}
.per_text h3{
        margin-right:30px;
        color:gray;
        font-family: cursive;
}
.menush{
        border: 0px solid red;
        width:100%;
        left:50%;
        text-align:center;
        transform: translate(-50%, -50%);
        position: relative;
```

```
margin-top:150px;
}
.menush img{
	padding:20px;
}
.menush h1{
	font-family: cursive;
	color:LawnGreen;
}
.menush h4{
border: 2px solid green;
	width:200px;
	height:50px;
	text-align:center;
	line-height: 50px;
	margin-top:10px;
	left:50%;
	transform: translate(-50%, -50%);
	position: relative;
	font-family: cursive;
	margin-top:50px;
}
.menush h4:hover{
	background-color:green;
	opacity:0.7;
}
.nagi{
	margin-top:-320px;
	position: relative;
	opacity:0.9;
}
.text_nagi{
	position: absolute;
	 top: 50%;
  left: 70%;
  transform: translate(-50%, -50%);
  font-family: cursive;
	color:LawnGreen;
	text-align:center
}
.photo_nagi{
	position: absolute;
	 top: 50%;
  left: 25%;
  transform: translate(-50%, -50%);
}
.photo_nagi img{
	padding:20px;
}
.text_pro {
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
```

```
  border: 0px solid MistyRose;
  padding: 10px;
}
.kart_text_pro{
        border: 0px solid white;

        margin-top:300px;
        position: relative;
transform: translate(-50%, -50%);
  width:1150px;
  left:50%;
}
.kart_text_pro img{
        opacity:0.9;
}
.per_text_pro{
        margin-left:600px;
        position: absolute;
        margin-top:-420px;
        font-family: cursive;
        text-align:center;
}
.per_text_pro h1{
        color:LawnGreen;
}
.per_text_pro h3{
        color:
}
:root {
  --main-color: #333;
  --main-width: 600px;
}
* {
  box-sizing: border-box;
}
.main_wraper {
  background: var(--main-color);
  height: calc(var(--main-width) / 1.75);
  margin: 20px auto;
  overflow: hidden;
  position: relative;
  width: var(--main-width);
}
.main_wraper>img {
  background: radial-gradient(#fff, var(--main-color));
  float: left;
  height: 25%;
  padding: 15px;
  position: relative;
  transition: all 0.5s ease-out;
  width: 20%;
}

.main_wraper>img:hover {
```

```
  cursor: pointer;
  padding: 3px;
  transition: all 0.3s ease-out;
}
.view_wraper {
  background: radial-gradient(#fff, var(--main-color));
  float: right;
  height: 75%;
  position: relative;
  right: 0;
  top: 0;
  width: 80%;
}
.view_image {
  background: transparent no-repeat center;
  background-size: contain;
  height: 100%;
  margin: 0 auto;
  width: 100%;
}
.thumb-select {
  filter: grayscale(100%);
  opacity: .2;
  pointer-events: none;
}
.thumb-hide {
  display: none;
}
.arrow_wrap {
  background: rgba(255, 255, 255, .3);
  border: 4px solid rgba(255, 255, 255, .6);
  border-radius: 20px;
  box-shadow: 1px 4px 20px -5px #000 inset;
  color: rgba(255, 255, 255, .8);
  cursor: pointer;
  font: 24px/32px 'Arial';
  height: 40px;
  outline: none;
  position: absolute;
  text-align: center;
  text-decoration: none;
  top: 50%;
  user-select: none;
  width: 40px;
}
.arrow_wrap:hover {
  color: rgba(255, 255, 255, 1);
  text-shadow: 1px 2px 6px #000;
}
.arrow_wrap:active {
  text-shadow: 0 0 1px #000;
}
.arrow-rew {
  left: 20px;
```

```
}
.arrow-fwd {
  right: 20px;
}
.vse_menu{
        position:relative;
}
.menu_menu{
        position:absolute;
        border:1px solid white;
        background-color:white;
        width:1000px;
        height:900px;
        top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
}
.vib_menu a{
        text-align:center;
        display: inline-block;
}
}
.swiper-wrapper{
}
.swiper-slide{
  overflow: hidden;
  @extend %backface_visibility_hidden;
  .slide-bgimg{
    position:absolute;
    top:0;
    left:0;
    width:100%;
    height:100%;
    background-position:center;
    background-size:cover;
  }
  .entity-img{
    display:none;
  }
  .content{
    position:absolute;
    top:40%;
    left:0;
    width:50%;
    padding-left:5%;
    color:#fff;
    .title{
      font-size:2.6em;
      font-weight:bold;
      margin-bottom:30px;
    }
    .caption{
      display:block;
      font-size:13px;
```

```
      line-height:1.4;
      transform:translateX(50px);
      opacity:0;
      transition:opacity .3s ease, transform .7s ease;
      &.show{
        transform:translateX(0);
        opacity:1;
      }
    }
  }
}
[class^=«swiper-button-»]{
  width:44px;
  opacity:0;
  visibility:hidden;
  @extend %transition_all_03s;
}
.swiper-button-prev{
  transform:translateX(50px);
}
.swiper-button-next{
  transform:translateX(-50px);
}
.slider {
    width: 100%;
    height: 400px;
    position: relative;
    overflow: hidden;
}
.slide {
    width: 70%;
    height: 80%;
    position: absolute;
    background-size: cover;
    background-position: center;
    transition: opacity 1s;
    opacity: 0;
          left:15%;
          top:10%;
          border-top-left-radius: 10px;
    border-top-right-radius: 10px;
    border-bottom-left-radius: 10px;
    border-bottom-right-radius: 10px;
}
.slide:first-child {
    opacity: 1;
}
```