Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

Навчально-науковий інститут деревообробних та комп'ютерних технологій і дизайну
(повне найменування інституту, назва факультету(відділення))

Кафедра інформаційних систем і комп'ютерного моделювання
(повна назва кафедри (предметної циклової комісії))

Пояснювальна записка

до дипломної роботи

перший (бакалаврський)

(рівень вищої освіти)

на тему: «Розроблення платформи для продажу майстер-класів засобами Angular»

Виконав студент 4 курсу, групи ІСТ-41
спеціальності: 126
„Інформаційні системи та технології"
(шифр і назва напрямку підготовки спеціальності)

Комаров Віталій Володимирович
(прізвище, ініціали)

Керівник: проф.Шабатура Ю.В.
(прізвище, ініціали)

Рецензент: Крошний І.М.
(прізвище, ініціали)

Львів-2023

Національний лісотехнічний університет України

(повне найменування вищого навчального закладу)

ННІ деревооброблювальних та комп'ютерних технологій і дизайну

Кафедра Інформаційних систем і комп'ютерного моделювання

Рівень вищої освіти перший (бакаравський)

Спеціальність 126 *„Інформаційні системи та технології"*

ЗАТВЕРДЖУЮ:

В.о завідувача кафедри ІСКМ

_____Сторожук О.Л.

„21" ____11____ 2022.

ЗАВДАННЯ

НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Комаров Віталій Володимирович

(прізвище, ім'я, по батькові)

1.Тема магістерської роботи: Розроблення платформи для продажу майстер-класів засобами Angular.

керівник роботи проф. Шабатура Ю.В.,

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від "21" 11 2022 року, №С-521

2.Термін подання студентом проекту(роботи) 10 червня 2023р

3. Вихідні дані до проекту (роботи) Розробити програмне забезпечення для продажу майстер-класів. Для розробки використати фрейсворк Angular. Програмне забезпечення має містити платіжну систему та таймінг на який людина заказала майстер-клас.

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

Стан проблемної області

Інформаційне забезпечення

Програмне забезпечення

Висновки

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

Підготовка матеріалу до доповіді .

6. Консультанти розділів проекту (роботи)

7. Дата видачі завдання 23.11.2022р.

КАЛЕНДАРНИЙ ПЛАН

№, з/п	Етапи бакалаврської роботи	Термін виконання етапів роботи	Примітка
1.	Огляд літератури згідно досліджуваної теми. Збір необхідних матеріалів.	23.11-20.12	виконано
2.	Постановка задачі і її формалізація	20.12-13.01	виконано
3.	Виконання вхідного етапу технології	13.01-14.04	виконано
4.	Реалізація головних класів проекту	14.04-20.05	виконано
5.	Виконання етапу відлагодження проєкту	20.05.-25.05.	виконано
6.	Виконання етапу впровадження та випуску бета-версії.	25.05.-02.06.	виконано
7.	Оформлення записки до дипломного проекту.	02.06.-10.06.	виконано

Студент _____ Комаров В. В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ проф.Шабатура Ю.В.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Дипломна робота містить 43 сторінок пояснювальної записки, 8 рисунків, 15 використаних джерел, 1 додаток.

Дипломна робота присвячена розробці веб-орієнтованої системи для продажу майстер-класів у випіканні та створені десертів. При створенні використано фреймворк Angular. База даних створювалась за допомогою Firebase Hosting - це статичний та динамічний вебхостин для розміщення різнотипних БД. В проєкті передбачено реалізацію адаптивності на різних платформах та діагоналях. Також враховано безконтактний розрахунок за допомогою платіжної системи.

Ключові слова: Angular, Firebase Hosting, платіжна система, програмне забезпечення.

ABSTRACT

The thesis contains 43 pages of explanatory notes, 8 pictures, 15 sources used, 1 appendix.

The thesis is devoted to the development of a web-oriented system for the sale of master classes in baking and creating desserts. The Angular framework was used during creation. The database was created using Firebase Hosting - it is a static and dynamic web host for hosting various types of databases. The project provides for the implementation of adaptability on different platforms and diagonals. Contactless payment using a payment system is also taken into account.

Keywords: Angular, Firebase Hosting, payment system, software.

ТЕХНІЧНЕ ЗАВДАННЯ

Розробити веб-орієнтовану систему для продажу майстер-класів у випіканні та створені десертів. При реалізації використати фреймворк Angular. Базу даних створити за допомогою Firebase Hosting (статичний та динамічний вебхостин для розміщення різнотипних БД). В проєкті передбачити реалізацію адаптивності на різних платформах та діагоналях. Також врахувати безконтактний розрахунок за допомогою платіжної системи.

ЗМІСТ

ВСТУП 7
ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ 8
РОЗДІЛ 1. СТАН ПРОБЛЕМНОЇ ОБЛАСТІ 9
1.1. Огляд проблемної області. 9
1.2. Огляд програмного забезпечення конкурентів 10
РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ТА МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ 14
2.1. Angular 14
РОЗДІЛ 3. ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ 20
3.1. Налаштування середовища. 20
3.2. Налаштування шаблону HTML та таблиці стилів 21
3.3. Створення макетних продуктів 22
3.4. Створення веб-орієнтованої системи 24
3.5. Тестування 39
ВИСНОВКИ 43
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 44
ДОДАТКИ 46

ВСТУП

Майстер-клас – це поглиблений та інтенсивний досвід навчання, зосереджений на просунутих, експертних або проміжних навичках, які викладають досвідчені інструктори. Інструктор дасть контекст навичкам, які ви будете практикувати, а також поради та методи, що використовуються в професійних умовах.

Оркестри, танцювальні школи, громадські організації тощо пропонують майстер-класи, щоб надати учням можливість освіти, яка недоступна у місцевих вчителів.

Крім того, найкращий спосіб покращити свої навички – спостерігати, як ті, хто нагорі, мають досвід, роблять те, що вони роблять, а потім намагаються вчитися у них. Це може бути онлайн, особисто або багато комбінацій обох. Майстри-художники та експерти проводять ці заняття або особисто, або віртуально, залежно від обставин.

Крім того, віртуальні майстер-класи включають відеоуроки у форматі HD, практичні інструменти та спільноту студентів. Віртуальні майстер-класи - один з кращих способів вчитися у профі.

Об’єктом дослідження використання Angular для створення торгівлі майстеркласами.

Метою роботи створення активного сервісу для залучення клієнтів до майстеркласів.

Предметом дослідження є використання фреймворку Angular для торгівлі.

Практичне значення гнучкий інструмент до торгівлі часом в мережі інтернет.

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

API – Application Programming Interface, інтерфейс програмування застосунків.

CLI – Command Line Interface, інтерфейс командного рядка.

HTML – HyperText Markup Language, мова розмітки гіпертекстових документів.

CSS – Cascading Style Sheets, каскадні таблиці стилів для оформлення веб-сторінок.

JS – JavaScript, мова програмування для створення інтерактивних веб-застосунків.

SPA – Single Page Application, односторінковий веб-застосунок.

PWA – Progressive Web Application, прогресивний веб-застосунок.

MVC – Model-View-Controller, архітектурний шаблон побудови програмного забезпечення.

MVVM – Model-View-ViewModel, архітектурний шаблон побудови користувацьких інтерфейсів.

POJO – Plain Old JavaScript Object, звичайний об’єкт JavaScript без додаткових залежностей.

SEO – Search Engine Optimization, оптимізація веб-сайту для пошукових систем.

РОЗДІЛ 1. СТАН ПРОБЛЕМНОЇ ОБЛАСТІ

1.1. Огляд проблемної області.

Платформи для створення онлайн-курсів можна розділити на дві категорії:

- Ринки курсів
- Програмне забезпечення для створення курсів

На ринку ваш курс є частиною каталогу, і у вас є можливість налаштувати цільову сторінку курсу, але не багато іншого, крім змісту курсу. Більшість маркетплейсів дозволяють публікувати курс безкоштовно, але беруть частку продажів курсів. Основна перевага: ринки курсів надають вам існуючу студентську базу, тому, якщо у вас ще немає великої присутності в Інтернеті, ви можете спочатку побалуватися ними [6].

З іншого боку, програмне забезпечення для створення онлайн-курсів пропонує набагато більше варіантів налаштування. Ви можете створювати фірмові цільові сторінки, вибирати один із кількох форматів вмісту під час створення курсу та отримувати необхідні інструменти для маркетингу свого курсу. Зазвичай вони стягують фіксовану щомісячну плату, а деякі платформи також стягують комісію за транзакцію [4].

Не дивно, що немає універсального рішення для створення прибуткового курсу. Ваша ідеальна платформа для створення курсів унікальна для ваших потреб і цілей. Чи є онлайн-курси вашим хлібом з маслом? Тоді вам потрібна платформа, яка допоможе вам охопити максимальну кількість студентів. Чи захоплює вас викладання та взаємодія зі студентами? Тоді творець курсу з інтерактивними інструментами буде добре працювати. Ви створюєте курс для залучення існуючої аудиторії? Тоді вам знадобиться інструмент із надійними маркетинговими функціями [5].

Хоча кожна платформа має свою унікальну точку продажу, я оцінив програмне забезпечення для онлайн-курсів нижче за певними критеріями:

Підтримувані формати вмісту, зокрема відео, аудіо, PDF-файли та зображення [8]:

- Функції редагування та параметри налаштування;
- Чи є вони реально доступними для малого та середнього бізнесу;
- Підтримка оцінок: вікторини, іспити, сертифікати тощо;
- Маркетингові та платіжні особливості.

Пам'ятаючи про ці критерії, ми звузили його до 21 програми, яку варто протестувати, а потім кілька днів пішло на то, щоб їх вивчити та зробити поглиблений процес тестування. Ось як це виглядало [2]:

- Реєстрація облікового запису та завершення будь-яких запропонованих процедур адаптації
- Проходження основного робочого процесу для створення нового курсу, включаючи перегляд будь-яких шаблонів курсів, створення навчальної програми та додавання вмісту курсу в різних форматах
- Тестування інших основних функцій, таких як створення вікторин і тестів, розробка сертифікатів курсів, планування сеансів у прямому ефірі та налаштування крапельних розкладів
- Персоналізація загального дизайну, кольорів, шрифтів, логотипів, цільових сторінок тощо в кожному додатку
- Нарешті, розглядаючи платіжні та маркетингові функції, такі як трансляції електронною поштою, структури ціноутворення, створення спільноти та SEO.

1.2. Огляд програмного забезпечення конкурентів

1. Udemy – найкраще підходить для асортименту доступних курсів.

Udemy пропонує широкий вибір курсів, починаючи від програмування та науки про дані до дизайну, ілюстрації та особистого розвитку. Незалежно від того, що ви хочете вивчити, у Udemy є курс. Udemy пропонує понад 185 000 онлайн-відеокурсів, які викладають досвідчені інструктори. Ви можете навчатися у своєму власному темпі та у вільний час, а на платформу Udemy постійно додаються нові курси. З різними викладачами приходять різні стилі курсів, тому ви можете знайти стиль викладання, який найкраще підходить для вас. Крім того, якщо ви експерт у певній галузі, ви можете створити свій курс і заробляти гроші, навчаючи інших на Udemy.

Що негативно в Udemy, так це те, що вони постійно пропонують курси зі знижкою, тому чиста сума, яку ви заробляєте на своєму класі, може бути не такою привабливою. Крім того, це чудова платформа для студентів. Ключовими особливостями є:

- Довічний доступ до курсів гарантує, що ви зможете переглядати матеріал так часто, як вам потрібно
- Навчайтеся у власному темпі за гнучким графіком
- Отримайте сертифікат про закінчення, щоб продемонструвати свої досягнення
- Задавайте питання викладачам і отримуйте відгуки, які допоможуть вам краще вчитися

Знайти курси відносно легко, а платформа розроблена, щоб зробити навчання легким і веселим. Незалежно від того, на яку тему ви хочете дізнатися, у Udemy є курс для неї. Вартість залежить від обраного вами курсу.

Перевага Udemy полягає в тому, що вони пропонують різні знижки протягом року.

2. Skillshare - це онлайн-платформа навчання з акцентом на творчість та дизайн.

Якщо ви хочете навчитися бути більш творчими або вдосконалити свої дизайнерські навички, Skillshare - чудовий варіант. Завдяки мобільному додатку, онлайн-курсам та офлайн-заходам Skillshare дозволяє легко вивчати нові навички та спілкуватися з іншими творчими людьми, де б ви не були. Завдяки курсам із призначеними їм проектами та спільноті творчих однодумців, з якими можна зв'язатися, Skillshare є чудовою платформою для тих, хто хоче вдосконалити свої творчі навички в різних сферах. Ключовими особливостями євляється:

- Отримайте змістовне членство, яке допоможе вам вести більш творче життя.
- Онлайн-навчання, розроблене для реального життя, дає вам гнучкість та підтримку, необхідну для досягнення ваших цілей.
- Навчайтеся звідусіль за допомогою мобільного додатка, який робить ваші курси доступнішими.

Рекомендації щодо продуктів, як правило, на місці і показують вам кілька чудових варіантів, які ви, можливо, не розглядали інакше. Є різні плани, які слід розглянути. Skillshare for Teams дозволяє збиратися разом і отримувати доступ до потрібного вмісту для груп людей. Корпоративні подарункові картки також показують співробітникам, що ви цінуєте їх розвиток. Нарешті, є також кілька безкоштовних курсів, які слід розглянути.

3. EdX - це найкраща онлайн-платформа навчання для навчальних закладів світу, що пропонує високоякісні курси з різних предметів.

Вивчення найкращих курсів, які можуть наблизити вас до кар'єрних цілей, не виходячи з дому, є великою перевагою, яку може запропонувати ця платформа. У рамках своєї основної місії EdX прагне забезпечити трансформацію як частину навчального процесу. Заснований на передових дослідженнях

когнітивної науки, EdX розроблений, щоб допомогти людям навчатися більш ефективно та досягати своїх цілей. Ключовими особливостями є:

- Випробуйте курси світового класу від найкращих інструкторів
- Навчайтеся у власному темпі за гнучким графіком
- Отримайте сертифікат про закінчення, щоб продемонструвати свої досягнення
- Від інформатики до бізнесу та менеджменту, інженерії та багато іншого, ви можете знайти відповідні курси з EdX.

Бачити прогрес курсу та те, де ви стоїте порівняно з однолітками – це чудовий спосіб залишатися мотивованим під час навчання на EdX. У зв'язку з цим існує також часова шкала та інформація про таблицю лідерів, яка дуже корисна. Це залежить від курсу, який ви вирішите обрати.

РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ТА МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

2.1. Angular

Сьогодні компанії прагнуть до надефективної присутності в Інтернеті. Отже, потреба в надійній структурі веб-розробки з найсучаснішими функціями є найвищою за весь час.

Angular - це дуже популярна платформа для веб-розробки, яка пропонує багатий досвід користувача, швидке реагування та можливість обслуговування коду. Згідно з опитуванням «Stack Overflow», це четвертий найбільш використовуваний зовнішній веб-фреймворк [2].

Цей інструмент був створений Google ще в 2009 році для допомоги в веб-розробці. Це одна з найбільш затребуваних фреймворків JavaScript, яка спрямована на те, щоб зробити інтерфейсну розробку набагато простішою та доступною [9].

Angular - це фреймворк із відкритим вихідним кодом, розроблений Google для створення динамічних сучасних веб-програм. Він використовує мову програмування TypeScript на основі JavaScript, щоб усунути непотрібний код і забезпечити легші та швидші програми [3].

Angular допомагає створювати інтерактивні та динамічні односторінкові програми (SPA) за допомогою своїх переконливих функцій, які включають шаблони, двостороннє зв’язування, модульність, обробку RESTful API, впровадження залежностей і обробку AJAX [13].

Дизайнери можуть використовувати HTML як мову шаблонів і навіть розширювати синтаксис HTML, щоб легко передавати компоненти програми [7]. Вам також не потрібно покладатися на сторонні бібліотеки для створення динамічних програм за допомогою Angular.

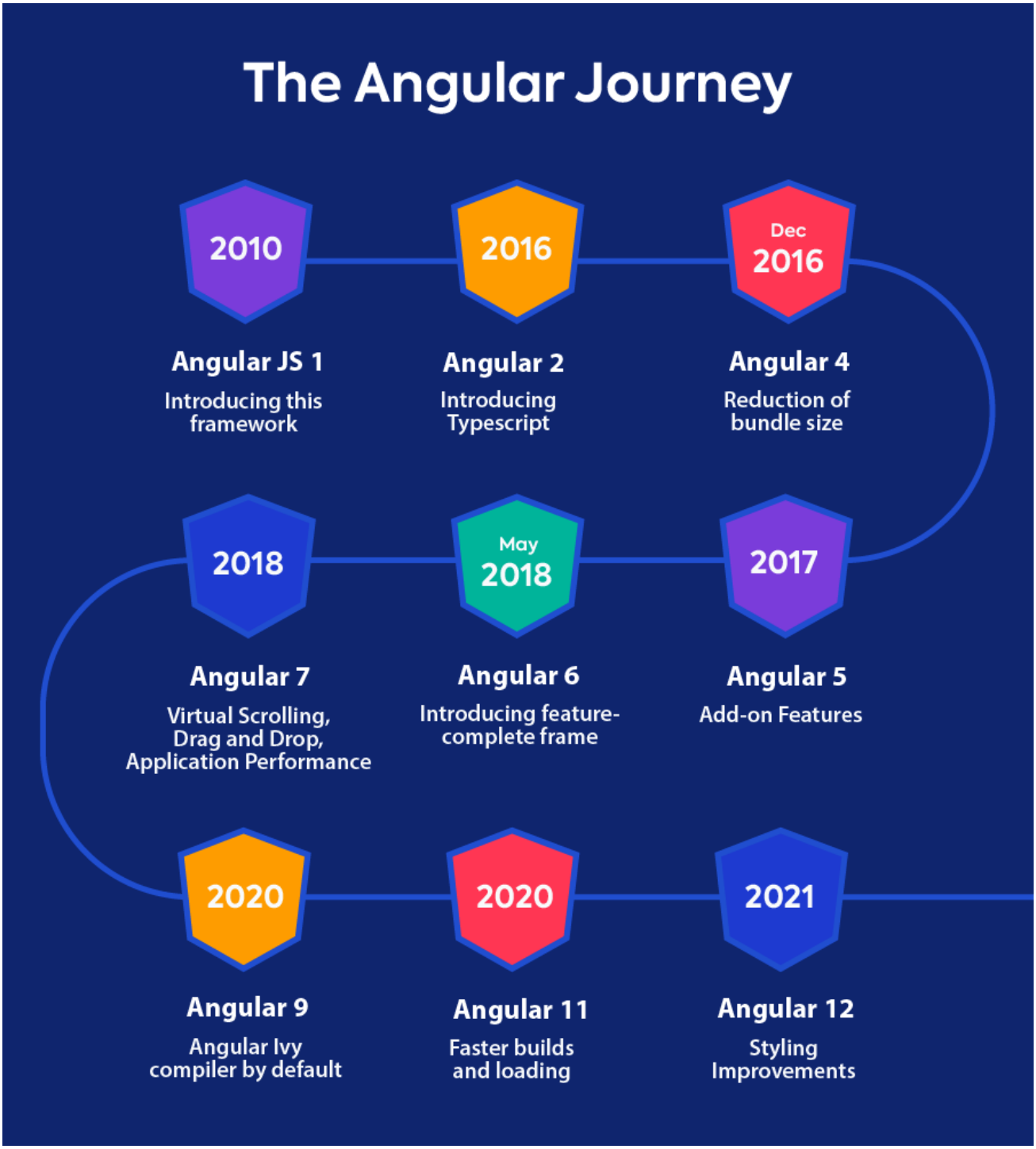

Рис. 2.1. Еволюція фрейморку

8 перевірених причин використовувати Angular

Тепер, коли ви маєте базове уявлення про Angular та його особливості, давайте обговоримо, чому Angular є чудовим вибором для ваших проектів розробки.

1. Підтримується Google

Однією з найбільших переваг Angular є те, що він підтримується Google. Google пропонує свою довгострокову підтримку (LTS) для Angular, яка проливає світло на плани Google дотримуватися фреймворку та подальшого масштабування екосистеми Angular [10].

Програми Google також використовують Angular, і їх команда досить оптимістично налаштована щодо його стабільності. Інші розробники Angular також отримують чудову можливість навчатися у сертифікованих професіоналів Angular від Google.

2.TypeScript

Додатки Angular створюються з використанням мови TypeScript, верхнього індексу для JavaScript, який забезпечує більш високий рівень безпеки, оскільки підтримує типи (примітиви та інтерфейси). Це допомагає виявляти та усувати помилки на ранніх стадіях процесу під час написання коду або виконання завдань обслуговування [12].

На відміну від CoffeeScript або Dart, TypeScript не є окремою мовою. За допомогою TypeScript ви можете легко взяти наявний код ES5 або ES2015+ JS, і він скомпілює його на основі того, що ви налаштовуєте. Він повністю підтримує основні функції ES2015 і ES2016/ES2017, такі як декоратори або async/await.

Ви також можете безпосередньо налагоджувати код TypeScript у браузері чи редакторі, якщо у вас є відповідні файли карт, створені під час збирання. Ця мова забезпечує покращену навігацію, рефакторинг і автодоповнення. Ви навіть можете відмовитися від його вбудованих функцій, коли це необхідно.

3. Декларативний UI

Angular використовує HTML для визначення інтерфейсу користувача програми. HTML, порівняно з JavaScript, є менш заплутаною мовою [11]. Це також декларативна та інтуїтивно зрозуміла мова з такими директивами, як ng-app, ng-model, ng-repeat і forms control.

З його допомогою вам не потрібно витрачати час на хід програми та вирішувати, що завантажується першим. Просто визначте, що вам потрібно, і Angular подбає про це.

4. POJO

З Angular вам не потрібні додаткові функції отримання та встановлення. Це пояснюється тим, що кожен об’єкт, який використовує Angular, є POJO (Plain Old JavaScript Object), який дозволяє маніпулювати об’єктами, надаючи всі звичайні функції JavaScript. Ви можете видаляти або додавати властивості до об’єктів, а також зациклювати ці об’єкти, коли потрібно [14].

5. PWA і SPA

Angular Progressive Web Application (PWA) – це економічне рішення, яке дозволяє веб-сайтам працювати як мобільні програми . Це зменшує залежність від мережі, що значно покращує взаємодію з веб-сайтом.

Кешування в PWA працює ефективно та зберігає пропускну здатність, коли це можливо. Це мінімізує ризики надання застарілого вмісту. Крім того, оскільки це веб-сайт, його можна оптимізувати для SEO.

Angular також сприяє розробці односторінкових додатків (SPA), які надають можливості рендеринга на стороні сервера, що підвищує рейтинг SEO. Це також допомагає швидко завантажити першу сторінку та підвищити ефективність веб-сайту на мобільних і малопотужних пристроях.

6. Спрощений шаблон MVC

Фреймворк Angular вбудовано в оригінальну архітектурну програму MVC (Model-View-Controller). Однак це не за встановленими стандартами [15]. Angular не просить розробників розділити додаток на різні компоненти MVC і створити код, який міг би їх об’єднати.

Швидше, він лише просить розділити додаток і піклується про все інше. Отже, структури дизайну Angular і MVVM (Model-View-View-Model) досить схожі.

Angular забезпечує легку розробку, оскільки усуває потребу в непотрібному коді. Він має спрощену архітектуру MVC, що робить непотрібним написання геттерів і сеттерів. Директивами може керувати інша команда, оскільки вони не є частиною коду програми. Загалом, розробникам обіцяють менше кодування, а також легші та швидші програми.

7. Модульна структура

Angular організовує код у відра , будь то компоненти, директиви, канали чи служби. Ті, хто знайомий з Angular, називають ці відра модулями. Модулі спрощують організацію функціональних можливостей програми, розділяючи її на функції та блоки, які можна багаторазово використовувати. Модулі також дозволяють відкладене завантаження, що відкриває шлях до завантаження функцій програми у фоновому режимі або на вимогу.

Angular робить досяжною метою розділити роботу між різними членами команди, забезпечуючи впорядкований код. Ви можете використовувати модулі якнайкраще, якщо добре їх розумієте. Розробники також можуть підвищити продуктивність за допомогою відповідних модулів.

8. Узгодженість коду та легке тестування

Кожна кодова база вимагає послідовного кодування. Це пояснюється тим, що неузгоджене кодування може збільшити ризики затримки запуску або підвищених витрат. З іншого боку, узгоджене кодування може полегшити використання вашого сайту та дозволити використання шаблонів або попередньо визначених фрагментів коду.

Кутовий каркас базується на компонентах, які починаються в одному стилі. Наприклад, кожен компонент розміщує код у класі компонентів або визначає декоратор @Component (метадані). Ці компоненти є невеликими

елементами інтерфейсу, незалежними один від одного, і пропонують вам кілька переваг, зокрема:

Багаторазове використання: компонентна структура Angular робить компоненти придатними для повторного використання в додатку. Ви можете створювати інтерфейс користувача (інтерфейс користувача) з рухомими частинами, забезпечуючи безперебійний процес розробки для розробників.

Спрощене модульне тестування: будучи незалежними один від одного, компоненти значно полегшують модульне тестування.

Покращена читабельність: послідовність у кодуванні робить читання коду шматком пирога для нових розробників у поточному проекті. Це підвищує продуктивність і загальну ефективність проекту.

Простота обслуговування: відокремлені компоненти можна замінити кращими реалізаціями. Простіше кажучи, це забезпечує ефективне обслуговування та оновлення коду.

Крім того, тестування в Angular надзвичайно просте. Модулі Angular.js містять частини програми, якими легко керувати. Завдяки поділу модулів ви можете завантажити необхідні служби, одночасно ефективно виконуючи автоматичне тестування. Вам навіть не потрібно запам'ятовувати порядок завантаження модулів, якщо ви дотримуєтеся принципу «один файл - один модуль».

РОЗДІЛ 3. ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1. Налаштування середовища.

Що було використано:

- Встановлений Node та NPM на комп'ютері;
- Обліковий запис Vercel;
- Обліковий запис Snipcart.

Перш за все, встановимо Angular CLI. Це потужний інструмент, який забезпечує автоматизацію багатьох розробницьких завдань. Для встановлення цього інструменту в терміналі потрібно ввести команду:

```
npm install -g @angular/cli
```

Після успішної установки, створюємо новий проект, використовуючи наведену нижче команду:

```
ng new snipcart-angular
```

Після цього отримаємо підказку, що стосується включення строгого режиму. Вибираємо опцію "так". Це дозволить активувати додаткові налаштування, що допомагатимуть виявляти помилки заздалегідь.

Тепер у нашому каталозі бачимо репозиторій свого проекту. Щоб увійти в нього, вводимо наступні команди:

```
cd snipcart-angular
```

Ця команда виконує зміну поточної робочої директорії в командному рядку до піддиректорії з назвою "snipcart-angular"

```
ng serve –open
```

Команда виконує запуск сервера розробки Angular і автоматично відкриває його у веб-браузері.

Використання команди ng serve дозволить нам створити програму, а додавши параметр --open, автоматично відкриється веб-браузер із відображенням нашого проекту за адресою http://localhost:4200/. Тепер повинна відобразитися сторінка загального шаблону Angular.

3.2. Налаштування шаблону HTML та таблиці стилів

Зараз, коли ми завершили загальні налаштування проекту, настав час перейти до його детальної настройки і налаштування.

У першу чергу, відкриваємо файл *app.component.ts* і вносимо зміни в значення властивості "title", змінивши його на "MasterClass Store".

Після цього відкриваємо файл *app.component.html* і заміняємо всі розділи шаблону на <**h1**>{{title}}</**h1**>, видаливши попередні вміст. Отриманий результат має виглядати таким чином:

```
<h1>{{title}}</h1>
<router-outlet></router-outlet>
```

Після виконання цих змін, браузер автоматично перезавантажиться, і ми побачимо тільки нову назву, без будь-якого іншого вмісту.

Для поліпшення стилю нашого проекту ми будемо використовувати компоненти **Angular Material Design**. Ці компоненти, розроблені командою **Material**, є реалізацією концепції **Material Design**. Вони нададуть нам можливість швидко створити стильний та перевірений дизайн для нашого інтернет-магазину.

Для встановлення **Material UI** використовуємо наступну команду:

```
ng add @angular/material
```

Після цього, ми можемо замінити файл *src/app.component.scss*.

3.3. Створення макетних продуктів

Зараз ми здійснимо просту дію і просто створимо файл *mock-products.ts* у кореневій папці нашого проекту:

```
import { Product } from './core/product';
import { Size } from './core/size';

export const PRODUCTS: Product[] = [
  {
    id: 1,
    name: 'Macarons MasterClass',
    imageUrls: ['../assets/master-class-prune.svg',
'../assets/master-class-cherry.svg', '../assets/master-
class-squash.svg'],
    price: 2400,
    duration: [
      { name: '30', color: '#5A188E' },
      { name: '60', color: '#F88532' },
      { name: '90', color: '#E91E63' },
    ],
    sizes: [Size.SMALL, Size.MEDIUM, Size.LARGE],
  },
  {
    id: 2,
    name: 'Cakes MasterClass',
    imageUrls: ['../assets/cake-lime.svg',
'../assets/cake-lettuce.svg', '../assets/cake-
cherry.svg'],
    price: 3800,
    duration: [
      { name: '30', color: '#00CACA' },
      { name: '60', color: '#80DC0B' },
      { name: '90', color: '#E91E63' },
    ],
    sizes: [Size.SMALL, Size.LARGE],
```

```
  },
];
```

Крім того, нам також знадобиться створити основну папку, яка буде містити TypeScript інтерфейс наших продуктів, а також інтерфейси для тривалості та розміру, які ми використовуватимемо для кращого задоволення потреб наших клієнтів у майстер-класах!

```
// core/product.ts

import { Duration } from "./duration";
import { Size } from "./size";

export interface Product {
    id: number;
    name: string;
    imageUrls: string[];
    price: number;
    duration: Duration[];
    sizes: Size[];
}
```

Описаний код визначає структуру даних для представлення товару в програмі. За допомогою цього інтерфейсу можна створювати об'єкти типу Product з відповідними властивостями. У наступних фрагментах визначається інтерфейс "Duration" та визначається перерахування (enum) "Size" (розмір).

```
// core/duration.ts

export interface Duration {
    name: string;
    color: string;
}
```

```
// core/size.ts
```

```
export enum Size {
  SMALL = "small",
  MEDIUM = "medium",
  LARGE = "large",
}
```

3.4. Створення веб-орієнтованої системи

Ми розпочинаємо створення головної сторінки нашого веб-сайту, де ми будемо відображати заголовок продукту та його варіанти. У командному рядку (терміналі) вводимо наступну команду:

```
ng generate component homepage
```

Додаємо властивості (props) до компонента головної сторінки, які будуть використовуватися для відображення назви та підзаголовка нашого додатку на веб-сайті. Для цього створимо дві властивості, одну для назви іншу для підзаголовка нашого додатку:

```
// homepage.component.ts

@Component({
  selector: 'app-homepage',
  templateUrl: './homepage.component.html',
  styleUrls: ['./homepage.component.scss']
})
export class HomepageComponent {
  title = 'The best pastry masterclasses in Ukraine';
  subtitle = 'Which one do you want?';
}
```

У наведеному фрагменті коду визначається компонент з назвою "HomepageComponent" для сторінки домашнього розділу (homepage). Цей компонент має декоратор @Component, який вказує налаштування компонента. В ньому вказані наступні налаштування:

- selector - селектор компонента, який може бути використаний для вставки цього компонента в шаблоні іншого компонента. У даному випадку, компонент можна вставити в шаблон за допомогою селектора app-homepage;
- templateUrl - шлях до зовнішнього шаблону, який буде використовуватися для відображення компонента. У даному випадку, шаблон знаходиться у файлі homepage.component.html;
- styleUrls - масив шляхів до зовнішніх файлів стилів, які будуть використовуватися для стилізації компонента. У даному випадку, стилі знаходяться у файлі homepage.component.scss.

Клас HomepageComponent містить дві властивості: title і subtitle. Ці значення можуть бути використані в шаблоні компонента для відображення відповідних текстів. Наприклад, title містить текст "The best pastry masterclasses in Ukraine", а subtitle містить текст "Which one do you want?". Наступним кроком є додавання HTML-шаблонування в відповідний файл.

```
<!-- homepage.component.html -->

<div class="header" fxLayout="column"
fxLayoutAlign="center center">
  <h1 class="jumbo">{{ title }}</h1>
  <h2>{{ subtitle }}</h2>
</div>
```

У наведеному фрагменті коду представлений шаблон (HTML-файл) компонента "HomepageComponent". Для відображення перегляду головної сторінки, додаємо наступний рядок до компонента:

```
<!-- app.component.html -->

<app-homepage></app-homepage>
```

У цьому фрагменті коду представлений шаблон (HTML-файл) компонента

"AppComponent". Тепер, після виконаних дій, ми повинні бачити відображення заголовка та підзаголовка! Далі перейдемо до створення продуктів для їх відображення.

Відображення продуктів: введення директив

Тепер, коли у нас є заголовок і декілька продуктів, ми відобразимо їх на нашому веб-сайті! Для цього ми створимо окремий компонент, що поліпшить його перевикористовуваність.

Для створення компонента "product-display" у терміналі, вводимо наступну команду за допомогою Angular CLI:

```
ng generate component products
```

Після успішного створення наших компонентів, ми можемо легко імпортувати необхідні продукти за допомогою Angular. Для цього додаємо наступні рядки до файлу "*product-display.component.ts*":

```
import { PRODUCTS } from '../mock-products';
```

Наведений фрагмент коду відбувається імпорт зовнішнього модуля або файлу з назвою "mock-products". Наступним кроком буде визначення атрибутів у класі компонента шляхом використання наступних рядків коду:

```
export class ProductsComponent implements OnInit {
  products = PRODUCTS;
}
```

У цьому фрагменті коду визначений клас ProductsComponent, який реалізує інтерфейс OnInit. Зараз ми готові створити компонент продукту, який надасть додаткову інформацію нашим клієнтам.

```
ng generate component product
```

Дана команда виконує генерацію нового компонента з назвою "product" за допомогою Angular CLI (Command Line Interface). Відкриваємо новостворений

файл product.component.ts і змінюємо його вміст на наступне:

```
import { Component, Input, OnInit } from '@angular/core';
import { Product } from '../core/product';

@Component({
  selector: 'app-product',
  templateUrl: './product.component.html',
  styleUrls: ['./product.component.scss']
})
export class ProductComponent implements OnInit {
  @Input() product: Product | undefined;
  imageUrl: string = "";

  ngOnInit() {
   this.imageUrl = this.product?.imageUrls[0] ?? '';
  }
}
```

У цьому фрагменті коду відбувається оголошення та визначення компонента "ProductComponent" в Angular. Доданий нами декоратор **@Input** дозволяє оголошувати властивості для введення. Це означає, що компонент тепер може отримувати значення безпосередньо від батьківського компонента.

При розгляді того, що ми щойно створили, можна помітити таке:

- Властивість "**product**" встановлена для об'єкта "**Product**", який був створений в модулі "**core**".
- Використовуватимемо властивість "**imageUrl**" для відображення зображень продукту.

За допомогою рядка **this.imageUrl = this.product?.imageUrls[0] ?? '';** ми присвоюємо властивості **imageUrl** значення першого зображення в масиві **imageUrls**, якщо воно існує. Це робимо у методі **ngOnInit**.

Метод **ngOnInit** в Angular є частиною життєвого циклу компонента і викликається після його ініціалізації. У нашому випадку, коли компонент

ініціалізований, властивості вводу, зокрема властивість **product**, заповнюються і стають доступними. Це дозволяє нам заповнити властивість **imageUrl**.

Тепер, коли властивості компонента визначені, ми можемо додати їх до HTML-коду (*product.component.html*):

```
<ul>
    <app-product *ngFor="let p of products"
[product]="p"></app-product>
</ul>
```

Створення список (ul) з використанням компонента "ProductComponent" для кожного елементу масиву products, передаючи кожен елемент як вхідний параметр product. Директива ***ngFor** дозволяє нам змінювати структуру DOM шляхом ітерації по елементах списку продуктів і створення вказаного HTML-вузла для кожного з них. У нашому випадку, ми використовуємо директиву ***ngFor** для створення вузла app-products для кожного елемента у списку продуктів.

Директиви, такі як ***ngFor**, є класами, які додають додаткову функціональність до елементів. У цьому контексті компоненти також є директивами, оскільки вони надають додаткову функціональність до стандартного HTML-шаблону.

Створення сторінки продукту: введення маршрутизації

Відкриваємо файл *app-routing.module.ts* і додаємо наступний код до масиву **routes**:

```
const routes: Routes = [
  {path: "**", component: HomepageComponent},
];
```

Визначає маршрутизацію для додатка, де шлях " ** " вказує на будь-який невизначений шлях і пов'язаний з ним компонент – HomepageComponent. Тепер ми можемо додати динамічний маршрут.

```
const routes: Routes = [
  {path: "product/:id", component: ProductPageComponent},
  {path: "**", component: HomepageComponent},
];
```

Цей код визначає маршрутизацію для нашого додатка. Він вказує, що якщо шлях починається з "product/:id", то буде використовуватись компонент ProductPageComponent для відображення сторінки продукту з відповідним ідентифікатором. У випадку, якщо шлях не співпадає з жодним іншим визначеним маршрутом, використовуватиметься компонент HomepageComponent для відображення домашньої сторінки.

Шлях "**" є маршрутом-шаблоном (wildcard route), який зазвичай використовується для сторінок 404. Важливо додати маршрут-шаблон останнім; в іншому випадку, він буде перекривати інші маршрути.

У файлі app.component.html, елемент <router-outlet></router-outlet> відображатиме компонент, пов'язаний з поточним маршрутом. Тому, якщо ми відкриємо адресу http://localhost:4200/ у веб-браузері, то побачимо нашу головну сторінку!

Створення компонента сторінок продуктів

Компоненти відповідають за відображення даних, але для покращення модульності програми, вони не повинні безпосередньо отримувати доступ до даних. Замість цього, компоненти повинні взаємодіяти з сервісами, які відповідають за доступ до даних програми.

Переробимо наш компонент продуктів таким чином, щоб він використовував сервіс для керування доступом до даних. Зараз ми будемо використовувати сервіс з мок-даними.

У терміналі введимо наступну команду:

```
ng generate service product
```

У новоствореному файлі *product.service.ts* додаємо наступний вміст:

```
import { Injectable } from '@angular/core';
import { PRODUCTS } from './mock-products';

@Injectable({
  providedIn: 'root'
})
export class ProductService {

  constructor() {}

  getProducts(): Product[] {
    return PRODUCTS;
  }
}
```

Оголошуємо сервіс ProductService, який надає функціональність для отримання списку продуктів. Метод **getProducts** просто повертає наші мок-дані. Пізніше ми модифікуємо його, щоб зробити його ще більш модульним. Наразі, у файлі *products.component.ts*, замінимо присвоєння мок-продуктів на виклик методу сервісу продукту:

```
export class ProductsComponent implements OnInit {
  products: Product[] = [];

  constructor(private productService: ProductService) {}

  getProducts(): void {
    this.products = this.productService.getProducts();
  }

  ngOnInit() {
    this.getProducts();
  }
```

```
}
```

Оголошуємо компонент ProductsComponent, який відповідає за відображення списку продуктів. Ось що ми написали вище:

1. Замінили значення **products** на пустий масив.
2. Впровадили **productService** у конструкторі.
3. Визначили метод **getProducts** у нашому компоненті, який відповідає за логіку продуктів.
4. Викликали цей метод у методі життєвого циклу **ngOnInit**.

Для сторінки продукту нам потрібна інформація про окремий продукт. Додамо метод getProduct до нашого сервісу продукту, щоб отримати ці дані:

```
// product.service.ts

getProduct(id: number): Observable<Product | undefined> {
   const product = PRODUCTS.find(product => product.id
=== id);
   return of(product);
}
```

Цей метод повертає об'єкт типу Observable. В Angular Observable використовується для обробки подій та асинхронного програмування. Ми використовуватимемо цей метод у нашому компоненті продукту для відображення вмісту продукту:

```
ng generate component product
```

```
// product.component.ts
import { Component, Input, OnInit } from '@angular/core';
import { Product } from '../core/product';

@Component({
  selector: 'app-product',
```

```
  templateUrl: './product.component.html',
  styleUrls: ['./product.component.scss']
})
export class ProductComponent implements OnInit {
  @Input() product: Product | undefined;
  imageUrl :string = "";

  ngOnInit() {
   this.imageUrl = this.product?.imageUrls[0] ?? '';
  }
}
```

Оголошуємо компонент ProductComponent, який відповідає за відображення окремого продукту. У файлі TypeScript ми додали вхідну властивість **product** для передачі продукту до компонента. Крім того, ми також додали властивість **imageUrl**, яку ми прив'язали до атрибуту **src** зображення компонента.

```
<!-- product.component.html -->

<div [routerLink]="'product/' + product?.id">
  <h2>{{ product?.name }}</h2>
  <img [src]="imageUrl" />
</div>
```

Цей код відповідає за розмітку (HTML) компонента ProductComponent. У HTML-файлі ми додали посилання маршрутизатора (router link) на сторінку продукту, яку ще не визначили. Далі ми це зробимо.

Додавання сторінок продуктів

Спочатку ми дозволимо користувачам вибрати тривалість та варіант продукту, який їм потрібен.

Наразі, окрім властивості imageUrl, яку ми додали до компонента продукту, також додаємо метод getProduct. Цей метод отримуватиме динамічний

параметр з нашого маршруту і використовуватиме його для виклику відповідного методу, який ми визначили у сервісі продукту:

```
// in product-page.component.ts
 imageUrl: string = '';
 product: Product | undefined;

 getProduct(): void {
    const id =
Number(this.route.snapshot.paramMap.get('id'));
    this.productService
      .getProduct(id)
      .subscribe((product) => (this.product = product));
  }
```

Отримання продукту зі специфічним ідентифікатором та встановлення відповідного зображення. Цей метод викликає метод **getProduct** у сервісі продукту і підписується на значення **Observable**. Коли значення повертається з методу **getProduct**, воно буде присвоєно властивості **product**.

Тепер, коли ми маємо всі необхідні дані, настав час відобразити назву продукту, URL зображення та ціну:

```
<h1>{{ product?.name }}</h1>
<img [src]="imageUrl" />
<p>Price: {{ product?.price }}</p>
```

На фрагменті відображення даних продукту на сторінці.

```
<h1>{{ product?.name }}</h1>
<img [src]="imageUrl" />
<p>Price: {{ product?.price }}</p>
<button
  class="snipcart-add-item"
  [attr.data-item-id]="product?.id"
  [attr.data-item-price]="product?.price"
  [attr.data-item-url]="'product/' + product?.id"
  [attr.data-item-image]="imageUrl"
  [attr.data-item-name]="product?.name">
```

```
  Add to cart
  </button>
```

Відображаємо дані продукту на сторінці і додаємо кнопку "Додати в кошик" з використанням Snipcart. При натисканні на кнопку "Купити", відкриється вікно оформлення замовлення Snipcart, де ви зможете переглянути і підтвердити свій продукт.

Сворення API для продукту за допомогою безсерверної функції Vercel

Атрибути, які ми додали до кнопки з класом "**snipcart-add-item**", визначають атрибути продукту (наприклад, ціну) під час оформлення замовлення. Оскільки компоненти Angular рендеряться динамічно під час виконання програми, HTML-пошуковик Snipcart не зможе знайти кнопку покупки під час пошуку HTML, оскільки вона ще не існує на етапі рендерингу HTML. Для вирішення цієї проблеми ми створимо невелике API, використовуючи безсерверну функцію Vercel. Це дозволить нам взаємодіяти з HTML-пошуковиком Snipcart та забезпечити правильну роботу кнопки покупки. JSON-пошуковик Snipcart буде здатен зчитати дані з нашого кінцевого пункту.

Vercel надає зручні можливості для реалізації безсерверних функцій без значних зусиль. В кореневій папці проекту просто створимо папку з назвою API. Усередині папки "API", яку ми створили раніше, створюємо ще одну папку з назвою "**products**". Потім всередині цієї папки створюємо файл з назвою "*[id].ts*". Дужки навколо "id" показують, що це динамічний параметр. Ми використовуємо його для отримання відповідного продукту і повернення його атрибутів, які впливають на ціноутворення.

Для цього додаємо такий вміст усередину файлу:

```
// <PROJECT'S ROOT>/api/products/[id].ts
import { VercelRequest, VercelResponse } from
'@vercel/node';
import { Product } from '../../src/app/core/product';
```

```
import { PRODUCTS } from '../../src/app/mock-products';

const findProduct = (id: number): Product | undefined =>
  PRODUCTS.find((product) => product.id === id);

export default function fetchProductInfo(
  req: VercelRequest,
  res: VercelResponse
) {
  const id = Number(req.query.id);
  const product = findProduct(id);
  res.statusCode = 200;
  res.send({
    id: id,
    name: product?.name,
    price: product?.price,
    url: `/products/${id}`,
  });
}
```

Цей код визначає функцію-обробник fetchProductInfo, яка відповідає за отримання інформації про продукт з вказаним ідентифікатором. Встанововимо модуль Vercel для визначень типів Node:

```
npm i -D @vercel/node
```

Ми можемо перевірити, що серверна функція працює локально, встановивши Vercel CLI. Потім введемо команду "**vercel .**" в кореневій папці нашого проекту, дотримуючись інструкцій для налаштування, а потім вводимо команду "**vercel dev**", щоб запустити локальне середовище.

Якщо перейдемо зараз за посиланням *https://localhost:3000/products/1*, отримаємо наступний файл:

```
{"id":1,"name":"Macarons
MasterClass","price":2400,"url":"/products/1"}
```

Об'єкт з інформацією про продукт, включаючи його ідентифікатор. Цей

JSON містить обов'язкові атрибути, необхідні для перевірки ціни в JSON-переглядачі Snipcart. Ці атрибути включають ідентифікатор (id), ціну та URL.

Налаштування власних витрат клієнтів з використанням користувацького поля в Snipcart

Додали деяку логіку для створення власних полів Snipcart. Це дозволить нашим користувачам обирати тривалість та розміри майстер-класу.

Спочатку ми створили дві функції, що повертають нам варіанти, розділені символом "|". Ми будемо використовувати їх для заповнення полів кнопки покупки з власними параметрами.

```
get flavorOptions(): string {
    return (
      this.product?.flavors?.map((flavor) =>
flavor.name).join('|') ?? ''
    );
  }

  get sizeOptions(): string {
    return this.product?.sizes?.join('|') ?? '';
  }
```

Код визначає два методи-аксесора (get) для отримання рядків зі списком варіантів тривалості (durationOptions) і розмірів (sizeOptions) продукту. Потім ми додаємо випадаючий список для вибору розміру продукту і компоненти фішок для вибору тривалості.

```
<!-- product-page.component.html -->
<mat-form-field appearance="fill">
  <mat-label>Size</mat-label>
  <mat-select
    (selectionChange)="
      updateSelectedProductAttributes(
        this.selectedAttributes?.flavor,
        $event.value
      )
```

```
    "
    required
  >
    <mat-option *ngFor="let size of product?.sizes"
[value]="size">
      {{ size }}
    </mat-option>
  </mat-select>
</mat-form-field>
<mat-chip-list aria-label="Duration selection">
  <mat-chip
    *ngFor="let duration of product?.duration"
    [style.background-color]="duration.color"
    (click)="
      updateSelectedProductAttributes(flavor,
this.selectedAttributes?.size)
    "
  >
    {{ duration.name }}
  </mat-chip>
</mat-chip-list>
```

Відображення форми на сторінці, де користувач може вибрати розмір та тривалість продукту, а також з певною логікою для відстеження обраних значень:

```
// core/selectedProductAttributes.ts

import { Duration } from "./duration";
import { Size } from "./size";

export interface SelectedProductAttributes {
    flavor: Duration | undefined;
    size: Size | undefined;
}
```

У фрагменті коду зображено визначення інтерфейс SelectedProductAttributes, який представляє об'єкт з вибраними атрибутами продукту.

```
// product-page.component.ts

export class ProductPageComponent implements OnInit {
  imageUrl: string = '';
  selectedAttributes: SelectedProductAttributes = {
    duration: undefined,
    size: undefined,
  };
  product: Product | undefined;
```

Визначаємо компонент ProductPageComponent, який представляє сторінку з детальною інформацією про продукт. Тепер нам лише потрібно включити атрибути для користувацького поля в кнопку покупки Snipcart.

```
<button
  class="snipcart-add-item"
  [attr.data-item-id]="product?.id"
  [attr.data-item-price]="product?.price"
  [attr.data-item-url]="'product/' + product?.id"
  [attr.data-item-image]="imageUrl"
  [attr.data-item-name]="product?.name"

  data-item-custom1-name="Duration"
  [attr.data-item-custom1-options]="durationOptions"
  [attr.data-item-custom1-
value]="this.selectedAttributes?.duration?.name"
  data-item-custom2-name="Size"
  [attr.data-item-custom2-options]="sizeOptions"
  [attr.data-item-custom2-
value]="this.selectedAttributes?.size"
>Add to cart</button>
```

Зображений код являється створенням кнопки "Add to cart" для додавання продукту до кошика.

3.5. Тестування

Рис. 3.1. Головна сторінка

Рис. 3.2. Корзина замовлень – «Порожня»

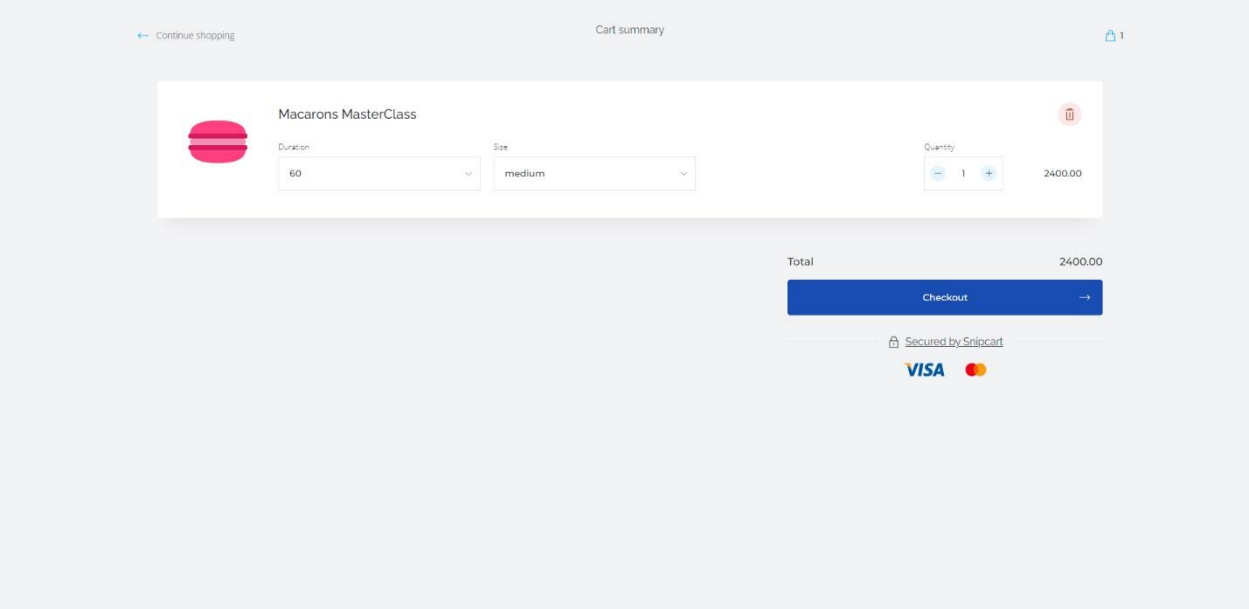

Рис. 3.3. Замовлення майстеркласу

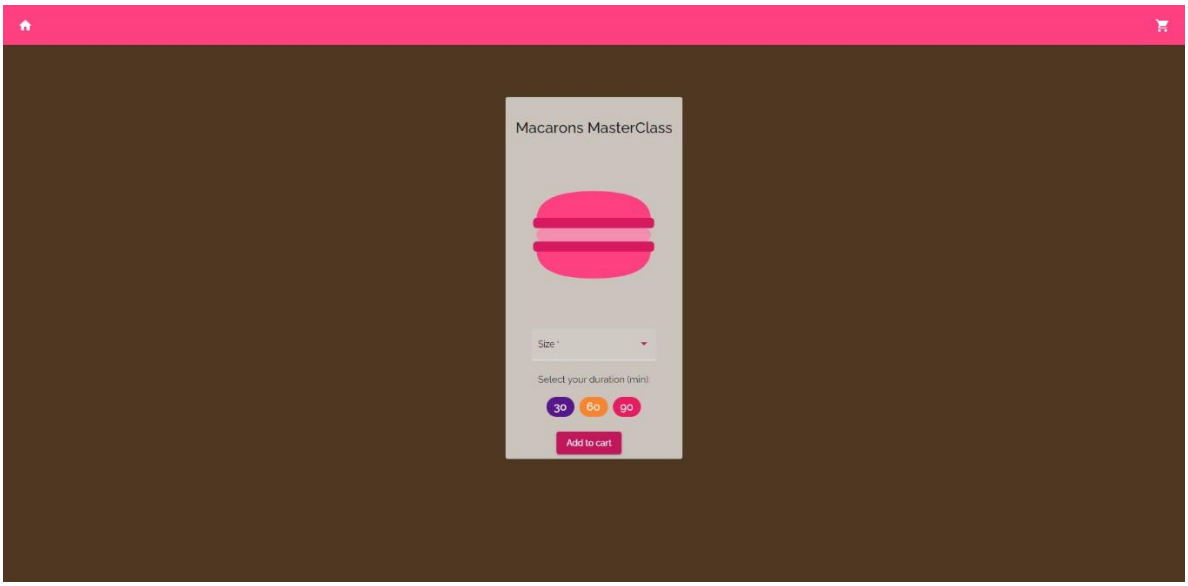

Рис. 3.4. Вибір часу майтеркласу

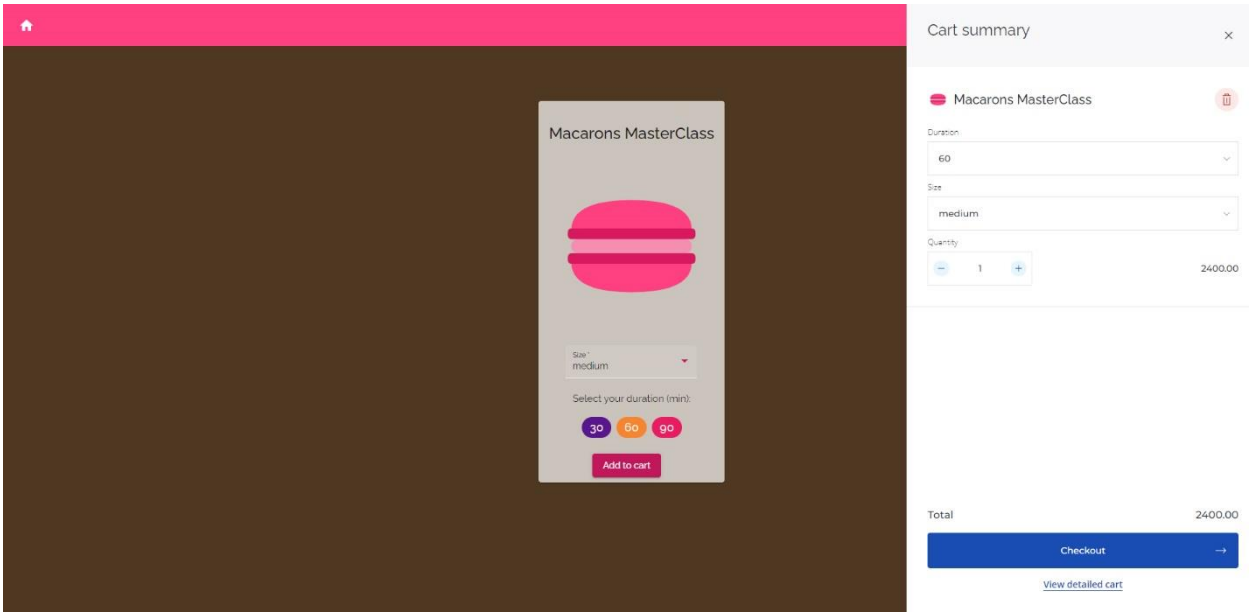

Рис. 3.5. В корзині товар

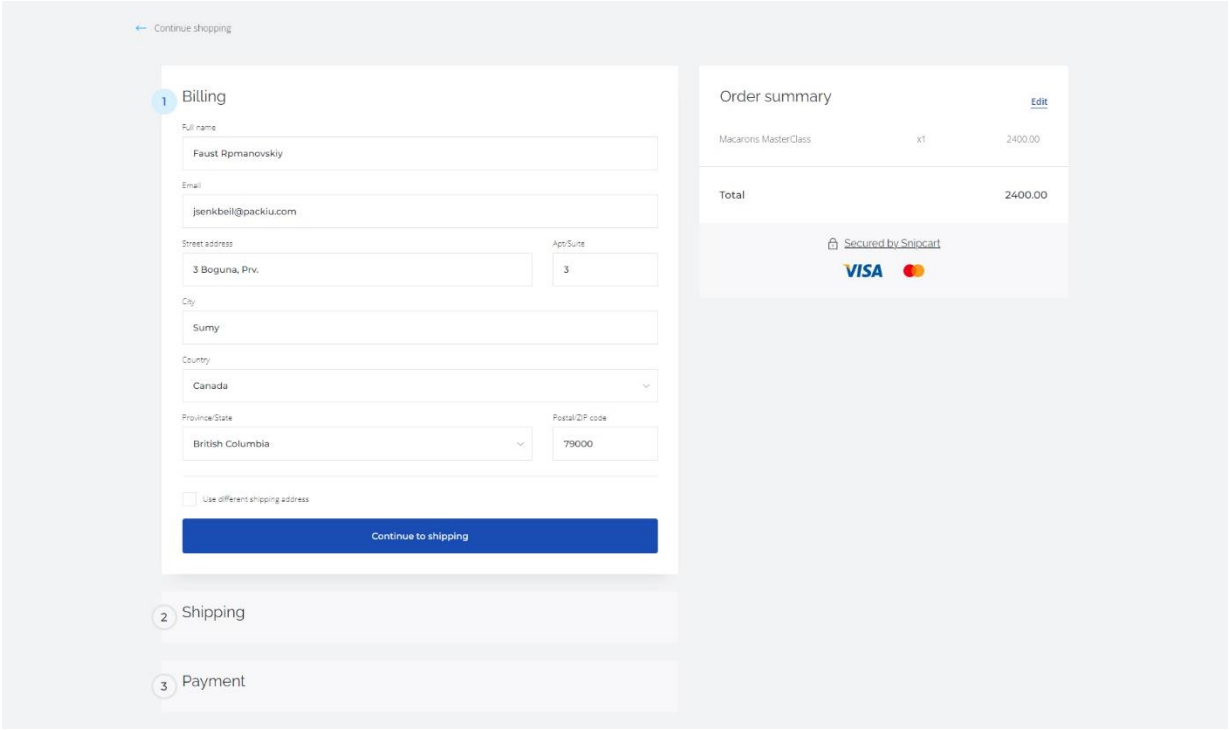

Рис. 3.6. Платіжна система

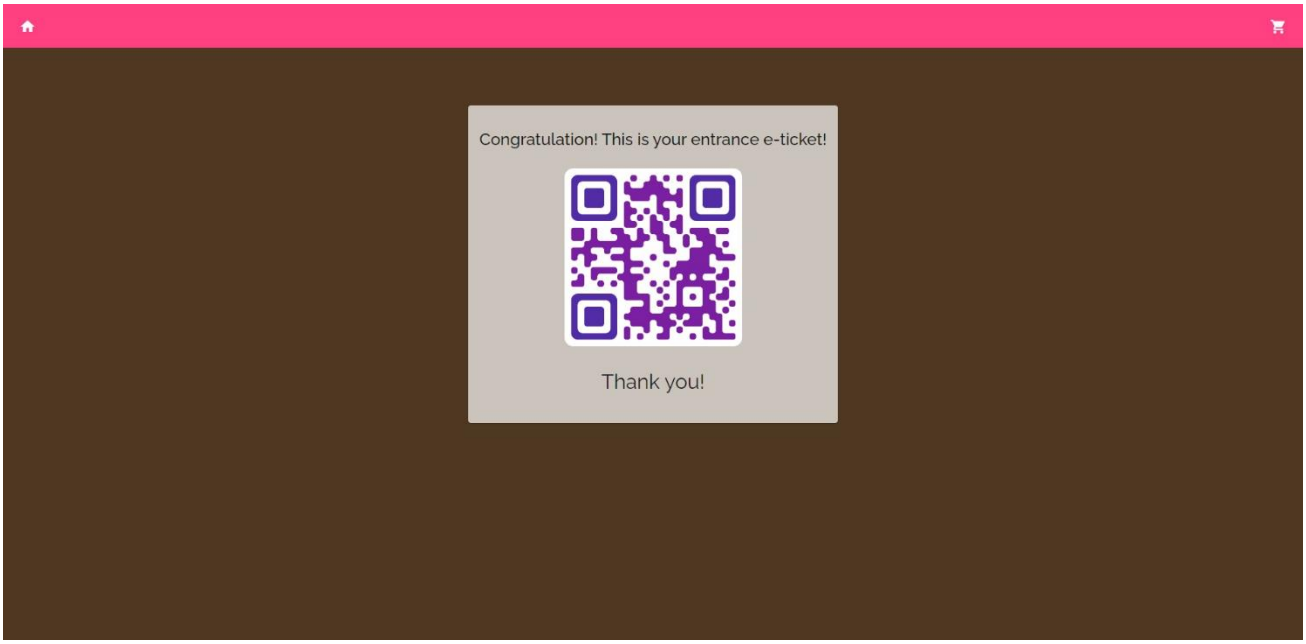

Рис. 3.7. Код для демонстрування при відвідуванні

ВИСНОВКИ

Розроблено веб-орієнтовану систему для продажу майстер-класів у випіканні та створені десертів. При реалізації використано фреймворк Angular. Базу даних створино за допомогою Firebase Hosting (статичний та динамічний вебхостин для розміщення різнотипних БД). В проєкті передбачено реалізацію адаптивності на різних платформах та діагоналях. Також враховано безконтактний розрахунок за допомогою платіжної системи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Adam Freeman: Pro Angular 2nd ed. Edition - NY, February 3, 2017.
2. Aristeidis Bampakos: Svelte 3 Up and Running: Learning Angular: A no-nonsense beginner's guide to building web applications with Angular 10 and TypeScript, 3rd Edition – Packt Publishing Ltd, September 7, 2020
3. Brad Dayley, Brendan Dayley, Caleb Dayley: Learning Angular, 2nd Edition – Addison-Wesley Professional, October 2017.
4. Bates T. Teaching in a Digital Age: Guidelines for Designing Teaching and Learning, Tony Bates Associates Ltd., 2019, 529 p.
5. Bonk C., Lee M. The World Is Open: How Web Technology Is Revolutionizing Education, Jossey-Bass, 2020, 512 p.
6. Clark R., Mayer R. E-Learning and the Science of Instruction, Wiley, 2020, 480 p.
7. Horton W. E-Learning by Design, Wiley, 2021, 640 p.
8. Anderson T. The Theory and Practice of Online Learning, Athabasca University Press, 2020, 472 p.
9. Dron J., Anderson T. How Education Works: Teaching, Technology, and Technique, Athabasca University Press, 2022, 384 p.
10. Salmon G. E-Moderating: The Key to Teaching and Learning Online, Routledge, 2020, 252 p.
11. Almarashdeh I. Sharing instructors’ experience of learning management system: A technology perspective of user satisfaction in distance learning, Computers in Human Behavior, 2020, 102 p.
12. Bierman G., Torgersen M., Cazzulino D. Understanding TypeScript, Packt Publishing, 2020, 410 p.
13. Sarrion E. Angular Projects: Build Modern Web Applications with Angular, Packt Publishing, 2020, 410 p.
14. Pereira F. Hands-On Full Stack Development with Angular, Packt Publishing, 2020, 450 p.

15. Angular Documentation. [Електронний ресурс] – Режим доступу до ресурсу: https://angular.io/docs (дата звернення: 15.04.2023).

ДОДАТКИ

```
import { Component, Input} from '@angular/core';

@Component({
  selector: 'app-homepage',
  templateUrl: './homepage.component.html',
  styleUrls: ['./homepage.component.scss']
})
export class HomepageComponent {
  title = 'The best pastry masterclasses in Ukraine';
  subtitle = 'Which one do you want?';
  }

import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-navbar',
  templateUrl: './navbar.component.html',
  styleUrls: ['./navbar.component.scss']
})
export class NavbarComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {
  }

}
```

```
import { Component, OnInit } from '@angular/core';
import { ActivatedRoute } from '@angular/router';
import { Duration } from '../core/durayion';
import { Product } from '../core/product';
import { SelectedProductAttributes } from
'../core/selectedProductAttributes';
import { Size } from '../core/size';
import { ProductService } from '../product.service';

@Component({
  selector: 'app-product-page',
  templateUrl: './product-page.component.html',
```

```
  styleUrls: ['./product-page.component.scss'],
})
export class ProductPageComponent implements OnInit {
  imageUrl: string = '';
  selectedAttributes: SelectedProductAttributes = {
    duration: undefined,
    size: undefined,
  };
  product: Product | undefined;

  constructor(
    private route: ActivatedRoute,
    private productService: ProductService
  ) {}

  getProduct(): void {
    const id = Number(this.route.snapshot.paramMap.get('id'));
    this.productService
      .getProduct(id)
      .subscribe((product) => (this.product = product));
  }

  get durationOptions(): string {
    return (
      this.product?.duration?.map((duration) =>
duration.name).join('|') ?? 'yolii'
    );
  }

  get sizeOptions(): string {
    return this.product?.sizes?.join('|') ?? 'yolii';
  }

  setImageUrl(duration: Duration): void {
    const durationImageUrl = this.product?.imageUrls.find((url) =>
      url.includes(duration.name)
    );
    if (!durationImageUrl) {
      throw Error(`No duration for ${duration.name} value`);
    }
    this.imageUrl = durationImageUrl;
  }

  ngOnInit() {
    this.getProduct();
```

```
    this.setSelectedAttributes(
      this.product?.duration[0],
      this.product?.sizes[0]
    );
    if (this.selectedAttributes?.duration) {
      this.setImageUrl(this.selectedAttributes.duration);
    }
  }

  public updateSelectedProductAttributes(duration: Duration |
undefined, size: Size | undefined) {
    this.setSelectedAttributes(duration ?? {name: "none", color:
"#DDD"}, size ?? Size.SMALL);
    if (this.selectedAttributes.duration) {
      this.setImageUrl(this.selectedAttributes.duration);
    }
  }

  private setSelectedAttributes(
    duration: Duration | undefined,
    size: Size | undefined
  ) {
    this.selectedAttributes = {
      duration: duration,
      size: size,
    };
  }
}
```