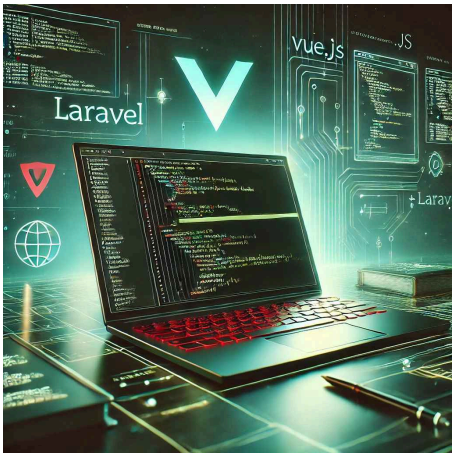
PRAKTYKI	Dokumentacja projektu
Autor	Dawid Olko
Kierunek, rok	Informatyka, III rok, st. stacjonarne (3,5-l)
Temat projektu	*Aplikacja e-komercyjna sklepu komputerowego z koszykiem*



App

04.02.2025r. - 12.02.2025r.

Spis treści

1. Narzędzia i technologie 3

2. Baza danych 6

3. GUI 12

4. Uruchomienie aplikacji 17

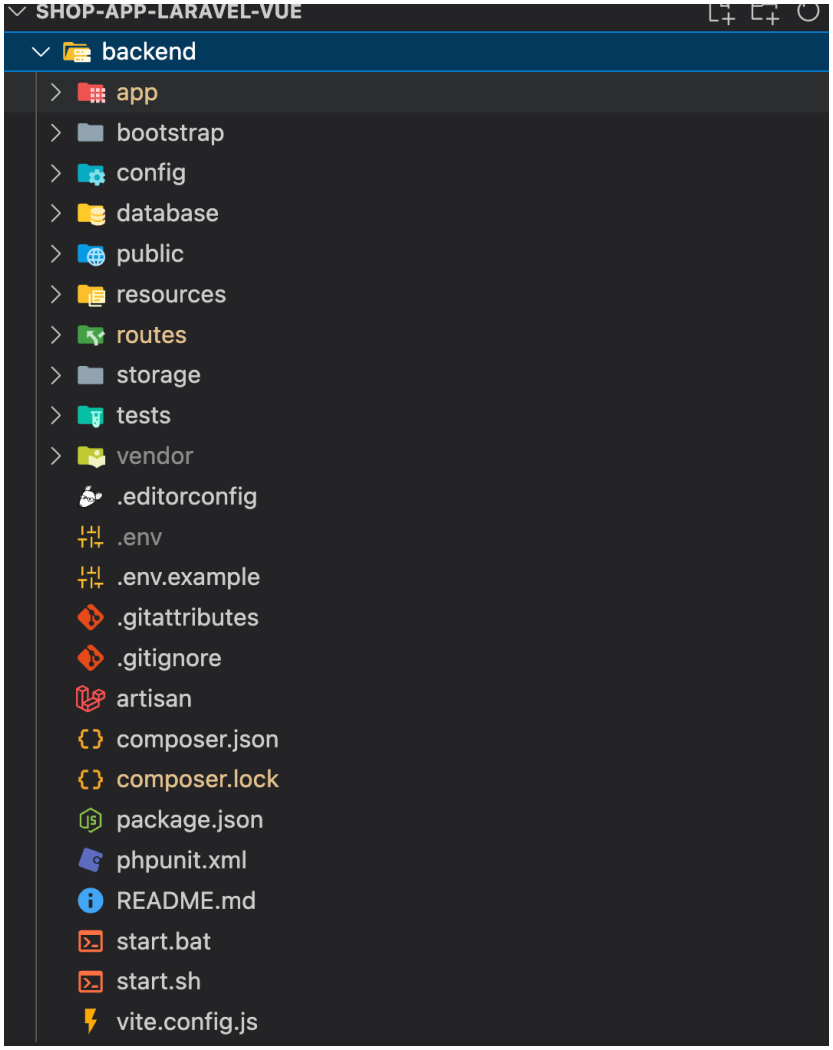
5. Podsumowanie 20

1. Narzędzia i technologie

1.1 Technologie użyte w projekcie

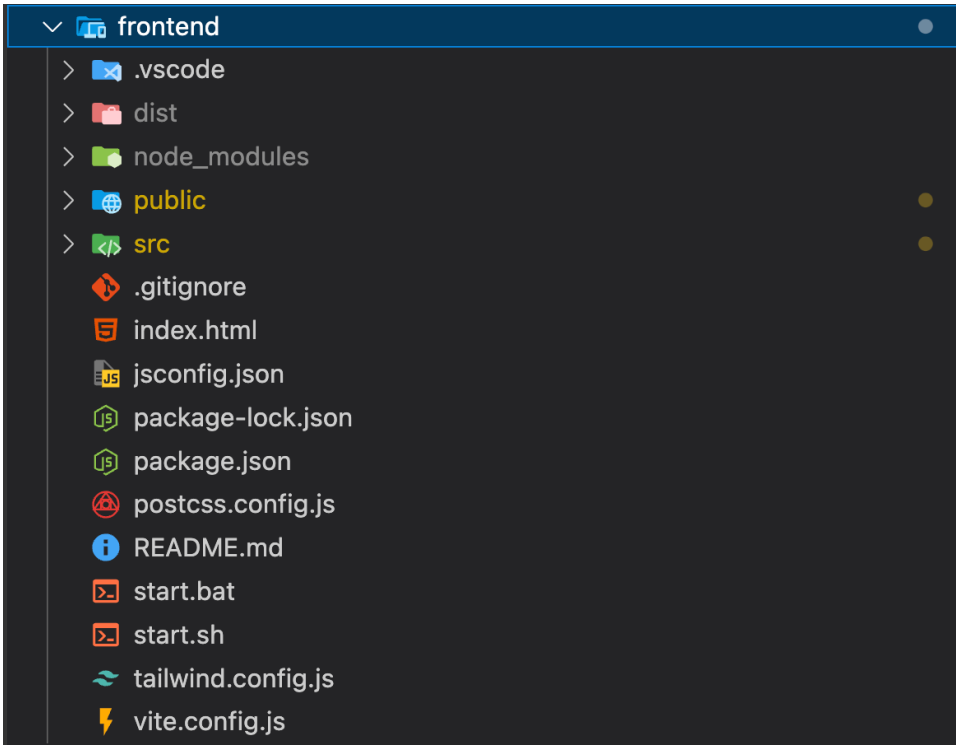
Projekt eCommerce ShopApp został zrealizowany przy użyciu nowoczesnych technologii webowych zarówno po stronie backendu, jak i frontendu.

Backend: Laravel (PHP)



- **Framework:** Laravel 11.4
- **Język programowania:** PHP 8+
- **Baza danych:** MySQL
- **ORM:** Eloquent
- **REST API:** Laravel (endpointy REST)
- **Autoryzacja:** Laravel Sanctum
- **Migracje bazy danych:** Laravel Migrations
- **Obsługa plików statycznych:** Laravel Storage
- **Obsługa zapytań HTTP:** Laravel HTTP Client

Frontend: Vue 3 + Vite



- **Framework:** Vue 3
- **Bundler:** Vite
- **Zarządzanie stanem:** Pinia
- **Routing:** Vue Router
- **Stylowanie:** Tailwind CSS (lub własne style oparte na klasach W3.css)
- **Ikony:** FontAwesome

Dodatkowe narzędzia i integracje:

- **System kontroli wersji:** Git
- **Repozytorium:** GitLab
- **Zarządzanie zależnościami:** Composer (backend), npm (frontend)
- **Narzędzia developerskie:** PHP Artisan, ESLint, Prettier

1.2 Struktura katalogów w projekcie

Projekt podzielony jest na trzy główne części:

- **Backend – Laravel:** Zawiera katalogi takie jak app/, database/, routes/ i inne pliki konfiguracyjne.
- **Frontend – Vue 3 + Vite:** Całość plików frontendu znajduje się w katalogu src/.
- **Baza danych:** Migracje oraz seedery są zarządzane przez Laravel Migrations i znajdują się w katalogu database/migrations/.

1.3 Narzędzia do zarządzania projektem

Projekt wykorzystuje szereg narzędzi usprawniających pracę i zarządzanie kodem:

- **System kontroli wersji:** Git, używany do śledzenia zmian w kodzie oraz wersjonowania.
- **Repozytorium:** GitLab – przechowywanie kodu źródłowego.
- **Zarządzanie zależnościami:** Composer dla backendu oraz npm dla frontendu.
- **Narzędzia CLI:** PHP Artisan do uruchamiania migracji, seedów oraz wykonywania innych zadań.

1.5 Wykorzystane biblioteki i zależności

Backend (Laravel):

- laravel/sanctum – autoryzacja użytkowników
- guzzlehttp/guzzle – obsługa zapytań HTTP
- Inne pakiety Laravel, np. debugbar, laravel/tinker

Frontend (Vue):

- vue-router – zarządzanie trasami
- pinia – zarządzanie stanem aplikacji
- axios – wykonywanie zapytań HTTP do API backendowego
- tailwindcss – nowoczesne stylowanie (lub własne style oparte na W3.css)

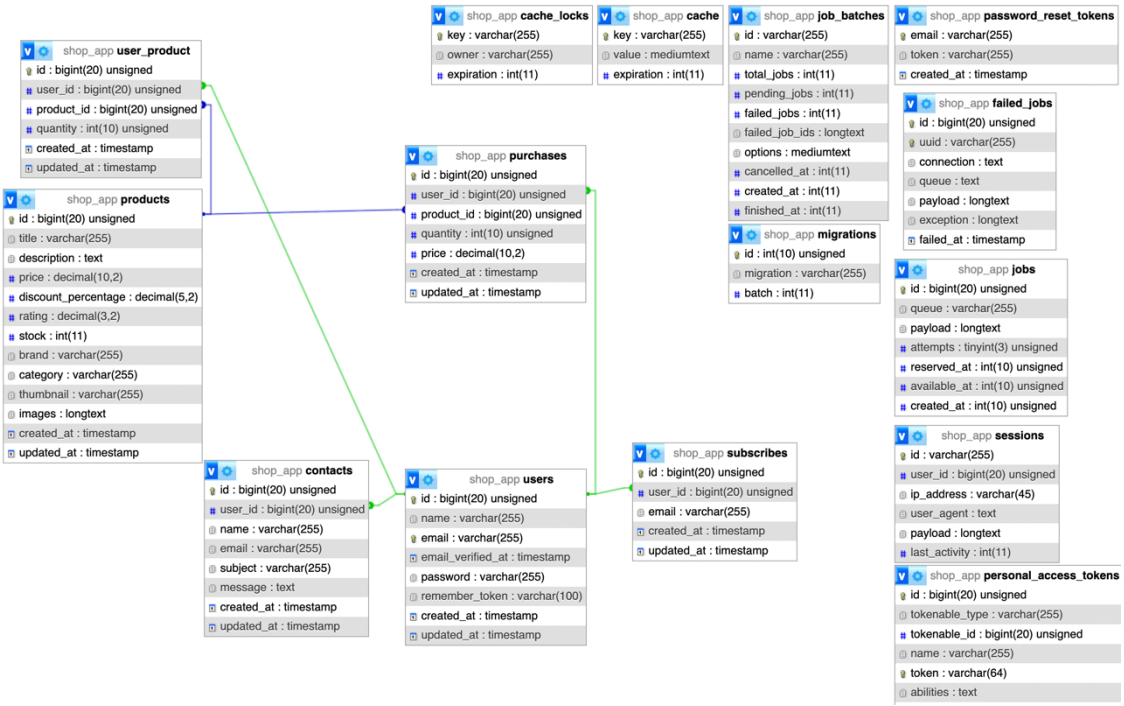
2. Baza danych

2.1 Struktura bazy danych

Projekt ShopApp korzysta z bazy danych MySQL, w której przechowywane są informacje o użytkownikach, produktach, koszyku, zakupionych produktach, subskrypcjach oraz kontaktach. Struktura bazy została zaprojektowana tak, aby umożliwić efektywne zarządzanie danymi i relacjami.

Główne tabele w bazie danych:

- **users** – przechowuje dane zarejestrowanych użytkowników.
- **products** – przechowuje dane produktów dostępnych w sklepie.
- **user_product** – tabela pivot, która przechowuje informacje o produktach dodanych do koszyka przez użytkowników.
- **purchases** – przechowuje dane dotyczące zakupionych produktów.
- **subscribes** – przechowuje informacje o subskrypcjach (formularz newslettera).
- **contacts** – przechowuje wiadomości wysyłane za pomocą formularza kontaktowego.
- **sessions** – przechowuje dane sesji użytkowników.
- **personal_access_tokens** – przechowuje tokeny autoryzacyjne dla użytkowników (obsługa Laravel Sanctum).



2.2 Opis tabel

Tabela: users (Użytkownicy)

Przechowuje dane zarejestrowanych użytkowników sklepu.

Kolumna	Typ danych	Opis
id	BIGINT UNSIGNED	Klucz główny
name	VARCHAR(255)	Imię i nazwisko użytkownika
email	VARCHAR(255)	Unikalny adres e-mail
password	VARCHAR(255)	Hasło użytkownika (hashowane)
email_verified_at	TIMESTAMP	Data weryfikacji adresu email (opcjonalne)
remember_token	VARCHAR(100)	Token zapamiętania użytkownika (opcjonalne)
created_at	TIMESTAMP	Data utworzenia rekordu
updated_at	TIMESTAMP	Data ostatniej aktualizacji rekordu

Tabela: products (Produkty)

Przechowuje informacje o produktach dostępnych w sklepie.

Kolumna	Typ danych	Opis
id	BIGINT UNSIGNED	Klucz główny
title	VARCHAR(255)	Nazwa produktu
description	TEXT	Opis produktu
price	DECIMAL(10,2)	Cena produktu
discount_percentage	DECIMAL(5,2)	Procent rabatu (jeśli dotyczy)
rating	DECIMAL(3,2)	Ocena produktu
stock	INT	Ilość produktu dostępna w magazynie
brand	VARCHAR(255)	Marka produktu
category	VARCHAR(255)	Kategoria produktu
thumbnail	VARCHAR(255)	Ścieżka do głównego zdjęcia produktu
images	JSON	Inne zdjęcia produktu zapisane jako JSON (tablica URLi)
created_at	TIMESTAMP	Data utworzenia rekordu
updated_at	TIMESTAMP	Data ostatniej aktualizacji rekordu

Tabela: user_product (Koszyk)

Tabela pośrednia przechowująca produkty dodane przez użytkowników do koszyka.

Kolumna	Typ danych	Opis
id	BIGINT UNSIGNED	Klucz główny
user_id	BIGINT UNSIGNED	Klucz obcy do tabeli users
product_id	BIGINT UNSIGNED	Klucz obcy do tabeli products
quantity	UNSIGNED INT	Ilość danego produktu w koszyku
created_at	TIMESTAMP	Data dodania produktu do koszyka
updated_at	TIMESTAMP	Data ostatniej aktualizacji rekordu

Tabela: purchases (Zakupy)

Przechowuje dane dotyczące dokonanych zakupów.

Kolumna	Typ danych	Opis
id	BIGINT UNSIGNED	Klucz główny
user_id	BIGINT UNSIGNED	Klucz obcy do tabeli users
product_id	BIGINT UNSIGNED	Klucz obcy do tabeli products
quantity	UNSIGNED INT	Ilość zakupionego produktu
price	DECIMAL(10,2)	Cena produktu w momencie zakupu (jednostkowa lub łączna)
created_at	TIMESTAMP	Data dokonania zakupu
updated_at	TIMESTAMP	Data ostatniej aktualizacji rekordu

Tabela: subscribes (Subskrypcje)

Przechowuje dane dotyczące subskrypcji newslettera.

Kolumna	Typ danych	Opis
id	BIGINT UNSIGNED	Klucz główny
user_id	BIGINT UNSIGNED	Klucz obcy do tabeli users (jeśli dotyczy)
email	VARCHAR(255)	Adres e-mail subskrybenta
created_at	TIMESTAMP	Data dodania subskrypcji
updated_at	TIMESTAMP	Data ostatniej aktualizacji rekordu

Tabela: contacts (Kontakty)

Przechowuje wiadomości wysłane przez użytkowników za pomocą formularza kontaktowego.

Kolumna	Typ danych	Opis
id	BIGINT UNSIGNED	Klucz główny
user_id	BIGINT UNSIGNED	Klucz obcy do tabeli users (jeśli dotyczy)
name	VARCHAR(255)	Imię nadawcy
email	VARCHAR(255)	Adres e-mail nadawcy
subject	VARCHAR(255)	Temat wiadomości
message	TEXT	Treść wiadomości
created_at	TIMESTAMP	Data wysłania wiadomości
updated_at	TIMESTAMP	Data ostatniej aktualizacji rekordu

Tabela: sessions

Przechowuje dane sesji użytkowników.

Kolumna	Typ danych	Opis
id	VARCHAR	Klucz główny (identyfikator sesji)
user_id	BIGINT UNSIGNED	Klucz obcy do tabeli users (opcjonalnie)
ip_address	VARCHAR(45)	Adres IP użytkownika
user_agent	TEXT	Informacje o przeglądarce użytkownika
payload	LONGTEXT	Dane sesji
last_activity	INT	Ostatnia aktywność (timestamp)

Tabela: personal_access_tokens

Przechowuje tokeny autoryzacyjne użytkowników.

Kolumna	Typ danych	Opis
id	BIGINT UNSIGNED	Klucz główny
tokenable_id	BIGINT UNSIGNED	Identyfikator modelu (użytkownika)
tokenable_type	VARCHAR	Typ modelu (np. App\Models\User)
name	VARCHAR(255)	Nazwa tokenu
token	VARCHAR(64)	Unikalny token (hashowany)
abilities	TEXT	Zdolności tokenu (np. uprawnienia)
last_used_at	TIMESTAMP	Data ostatniego użycia tokenu
expires_at	TIMESTAMP	Data wygaśnięcia tokenu (opcjonalnie)
created_at	TIMESTAMP	Data utworzenia rekordu
updated_at	TIMESTAMP	Data ostatniej aktualizacji rekordu

2.3 Relacje między tabelami

- **users – products**: Użytkownik może dodawać produkty do koszyka; relacja wiele-do-wielu realizowana jest przez tabelę user_product.
- **users – purchases**: Każdy użytkownik może dokonywać zakupów; zakupy zapisywane są w tabeli purchases (relacja jeden-do-wielu).
- **users – subscribes**: Użytkownik może subskrybować newsletter; relacja (opcjonalna) jeden-do-wielu.
- **users – contacts**: Użytkownik może wysyłać wiadomości kontaktowe; relacja (opcjonalna) jeden-do-wielu.
- **products – purchases**: Produkt może być częścią wielu zakupów; relacja jeden-do-wielu.
- **products – user_product**: Produkt może być dodany do koszyka przez wielu użytkowników (relacja wiele-do-wielu).
- **users – sessions** oraz **users – personal_access_tokens**: Standardowe relacje obsługujące sesje i tokeny autoryzacyjne.

2.4 Mechanizm zakupów i zbierania danych

W projekcie ShopApp kluczową funkcjonalnością jest możliwość zakupów przez użytkowników. Proces ten przebiega następująco:

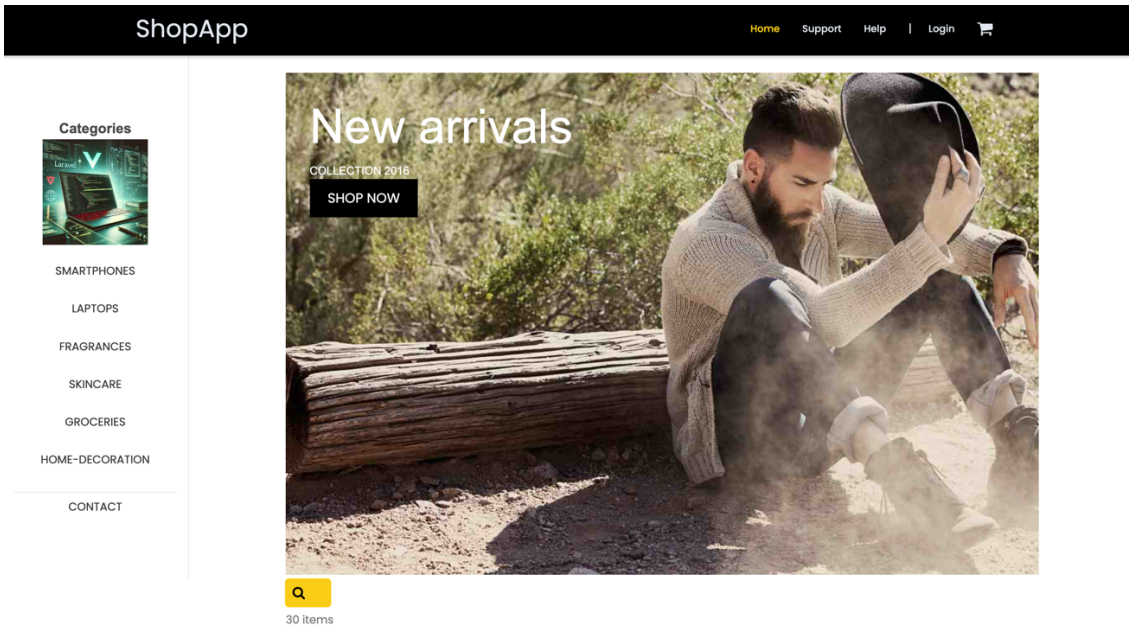
1. Użytkownik dodaje produkty do koszyka, a dane te są zapisywane w tabeli user_product.
2. Po kliknięciu przycisku "Buy Now" następuje wywołanie endpointu, który:
    - Sprawdza dostępność produktów (stan magazynowy).
    - Tworzy wpisy w tabeli purchases zawierające informacje o zakupionych produktach (ilość, cena, data zakupu).
    - Aktualizuje stan magazynowy produktów.
    - Czyści koszyk (usuwając wpisy z tabeli user_product).
3. Dane o zakupach są dostępne w panelu użytkownika, gdzie użytkownik może przeglądać historię swoich zamówień.
4. Formularze subskrypcji i kontaktu (obsługiwane przez tabele subscribes oraz contacts) pozwalają użytkownikom na zapisanie się do newslettera lub wysłanie wiadomości do obsługi sklepu.

3. GUI

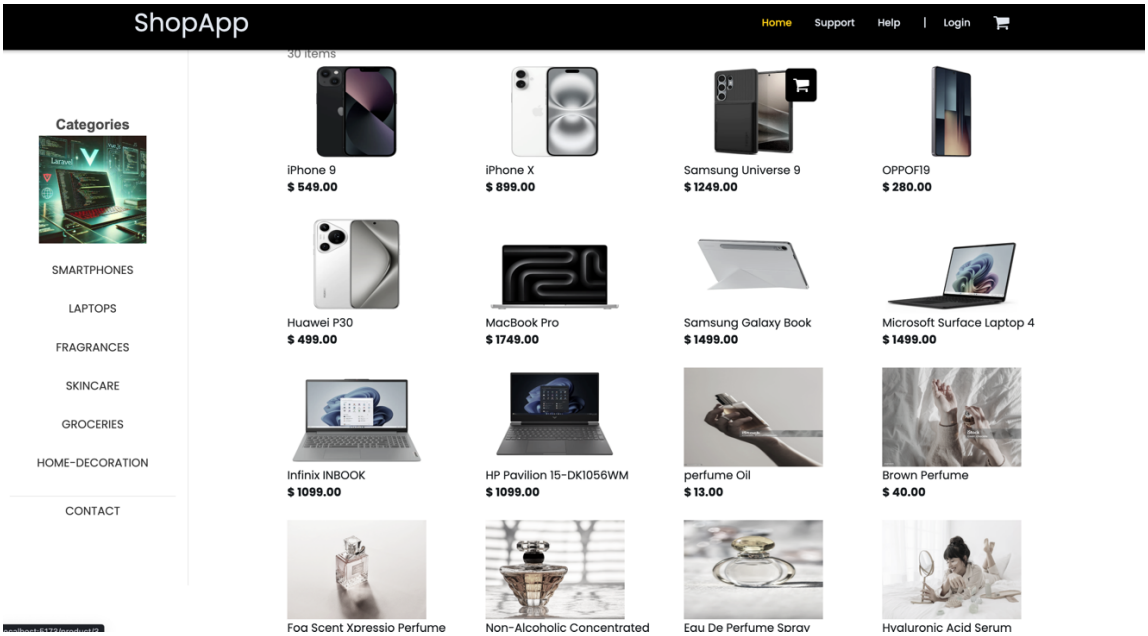
3.1 Przegląd interfejsu aplikacji

Aplikacja ShopApp posiada nowoczesny i intuicyjny interfejs użytkownika, który został zaprojektowany z myślą o łatwej nawigacji oraz wygodzie korzystania. Główne elementy interfejsu to:

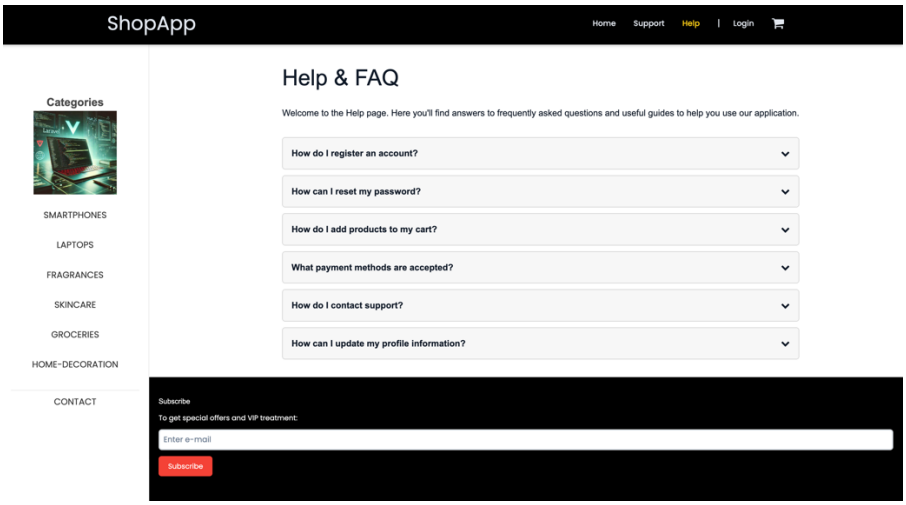
- **Strona główna** – zawiera duży baner z promocjami, pole wyszukiwania z ikoną lupy (stylizowaną na biało z czarnym tłem, zmieniającą kolor na pomarańczowy i skalującą się o 10% po najechaniu) oraz dynamicznie aktualizowany grid produktów.



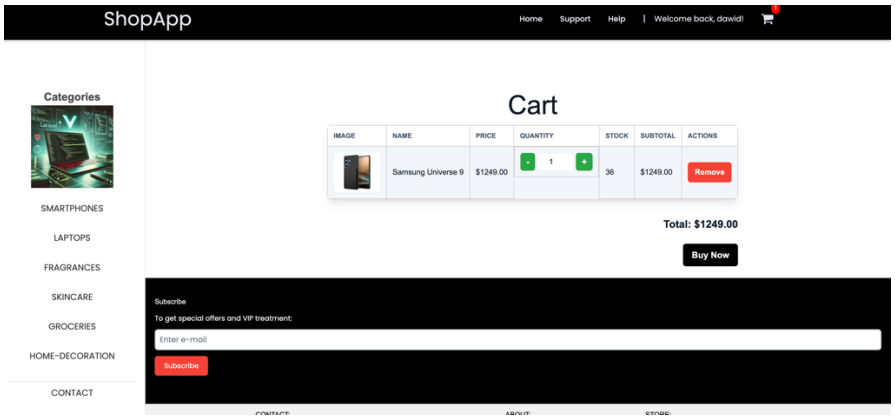
- **Panel główny (MainSection)** – prezentacja produktów w formie siatki, gdzie każdy produkt wyświetlany jest z obrazkiem, nazwą, ceną oraz przyciskiem dodawania do koszyka. Po wpisaniu frazy w pole wyszukiwania produkty filtrują się na bieżąco według nazwy, opisu, kategorii i marki.



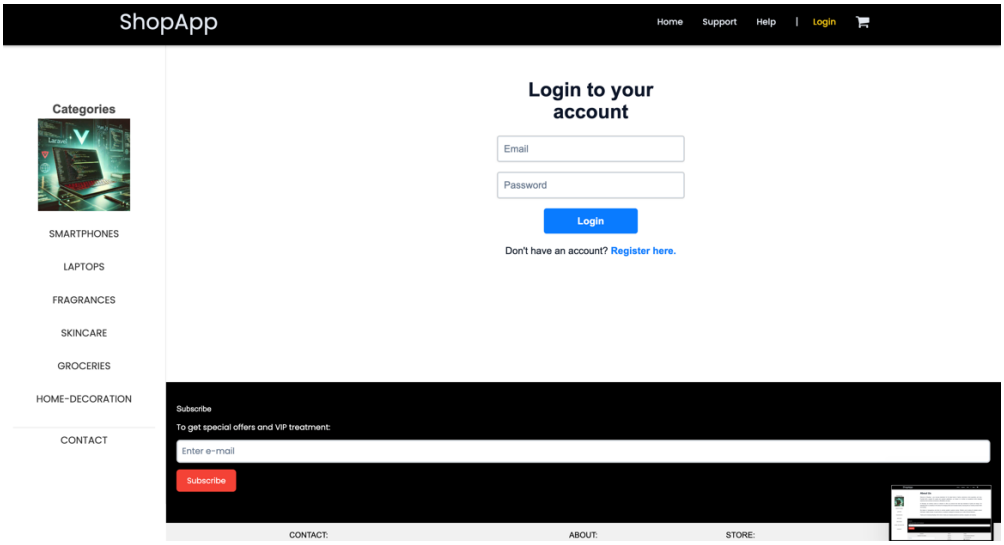
- **Widok szczegółowy produktu** – umożliwia obejrzenie produktu w powiększeniu, przeglądanie dodatkowych zdjęć w sliderze oraz przejście do zakupu.

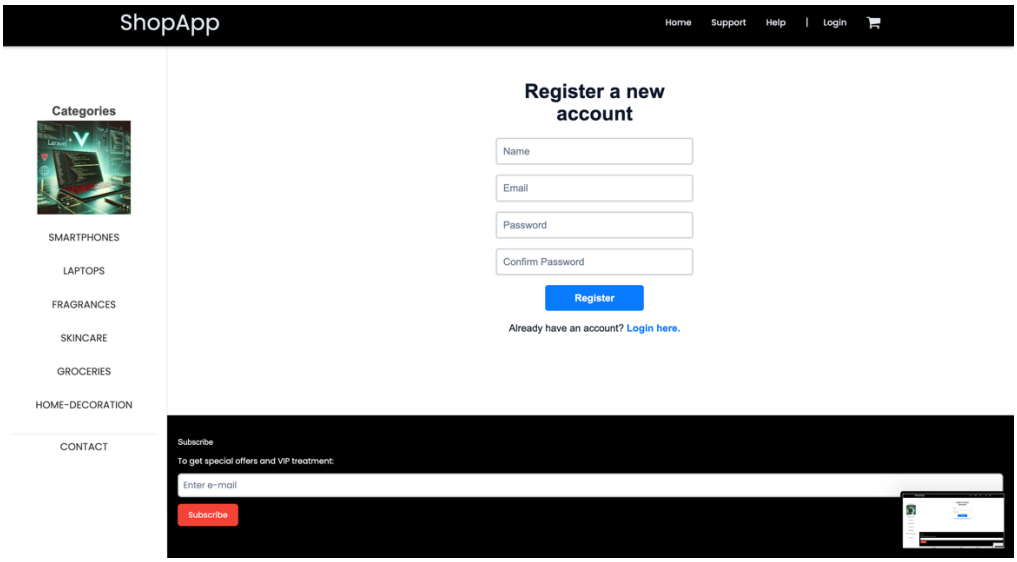


- **Koszyk** – wyświetla listę dodanych produktów, umożliwia zmianę ilości za pomocą przycisków +/- oraz usuwanie pozycji. Dodatkowo, użytkownik może zatwierdzić zakup, co przenosi dane z koszyka do historii zakupów.

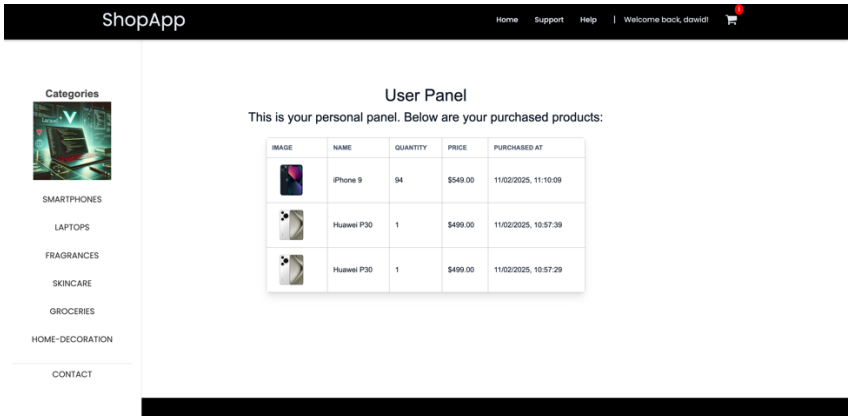


- **Widok logowania i rejestracji** – estetyczne formularze umożliwiające szybkie i bezproblemowe logowanie się oraz rejestrację.

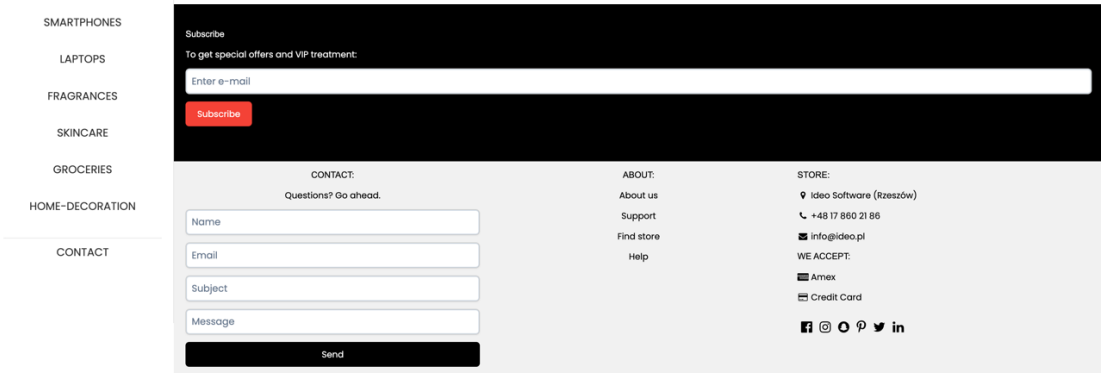


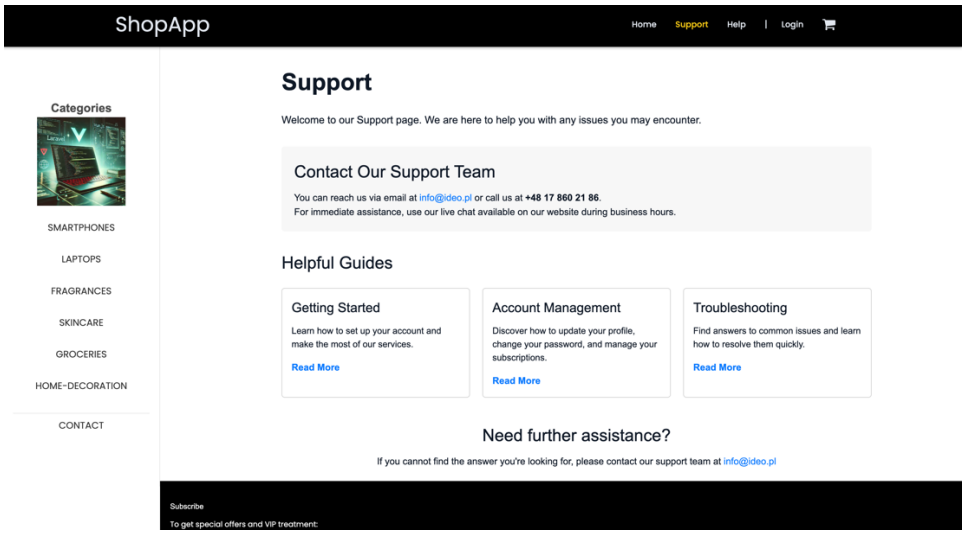


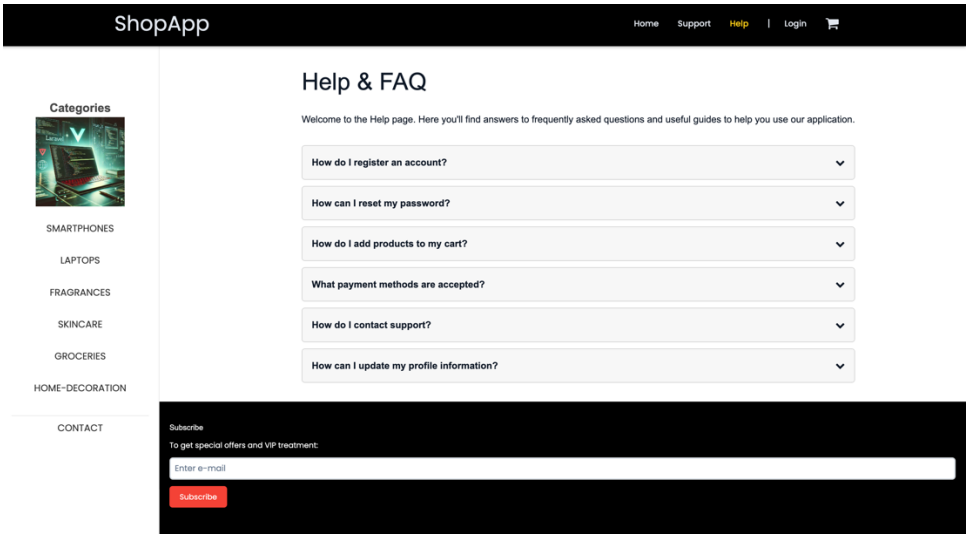
- **Panel użytkownika** – miejsce, w którym użytkownik może przeglądać historię swoich zakupów, zarządzać danymi konta, a także zmieniać ustawienia (np. aktualizować nazwę, hasło).



- **Strony pomocowe** – sekcje takie jak „About Us”, „Support” i „Contact” prezentują informacje o sklepie, kanały kontaktowe oraz szczegółowe przewodniki lub formularze umożliwiające kontakt z obsługą sklepu.







3.2 Rejestracja i logowanie

Proces rejestracji i logowania został zaprojektowany w sposób przejrzysty i intuicyjny.
Ekran rejestracji zawiera:

- Formularz z polami: Imię i nazwisko, adres e-mail, hasło oraz potwierdzenie hasła.
- Po poprawnym wypełnieniu formularza użytkownik otrzymuje potwierdzenie utworzenia konta oraz automatyczne zalogowanie.

Ekran logowania zawiera:

- Formularz z polami: adres e-mail i hasło.
- Po zalogowaniu użytkownik uzyskuje dostęp do funkcji takich jak dodawanie produktów do koszyka, zakupy oraz personalizacja ustawień konta.

3.3 Strona koszyka

Strona koszyka prezentuje produkty dodane przez użytkownika w formie tabeli.
Główne elementy widoku koszyka:

- Lista produktów wraz z obrazkiem, nazwą, ceną, ilością (z przyciskami zwiększania/zmniejszania) oraz stanem magazynowym.
- Wyświetlanie sumarycznej ceny zamówienia.
- Przyciski usuwania pozycji z koszyka oraz przycisk „Buy Now", który finalizuje zakup, przenosząc produkty z koszyka do historii zakupów.

3.4 Panel użytkownika

Panel użytkownika to miejsce, gdzie klient może:

- Przeglądać historię swoich zakupów – tabela zawiera informacje o zakupionych produktach, ilościach, cenach i datach zakupu.
- Zarządzać swoimi danymi – dostępny jest ekran ustawień, umożliwiający zmianę imienia oraz hasła.
- Przeglądać osobiste promocje oraz status konta.

3.5 Ekran ustawień użytkownika

Na stronie ustawień użytkownika dostępne są funkcje umożliwiające aktualizację danych osobowych.
Kluczowe elementy ekranu ustawień:

- Formularz do zmiany nazwy użytkownika.
- Opcja aktualizacji hasła (wraz z potwierdzeniem nowego hasła).
- Przycisk „Save Changes" potwierdzający wprowadzone zmiany, przy czym w przypadku błędów lub sukcesu wyświetlany jest odpowiedni komunikat modalny.

3.6 Responsywność i animacje

Interfejs ShopApp został zaprojektowany z myślą o pełnej responsywności, dzięki czemu aplikacja wygląda i działa poprawnie na urządzeniach o różnych rozdzielczościach.

- **Widok desktopowy** – menu, wyszukiwarka i inne elementy są wyświetlane w optymalnym układzie, a panel główny ma margines z lewej strony (300px) od sidebaru.
- **Widok mobilny** – układ dostosowany do mniejszych ekranów; menu nawigacyjne oraz przyciski są łatwo dostępne i przystosowane do interakcji dotykowej.
- **Animacje** – interfejs zawiera płynne animacje (np. przycisk lupy powiększa się lub zmniejsza przy hoverze, rozwijane sekcje i przyciski zmieniające kolor), co poprawia komfort korzystania z aplikacji.

4. Uruchomienie aplikacji

4.1 Wymagania systemowe

Aby uruchomić aplikację eCommerce ShopApp, należy spełnić następujące wymagania systemowe:

- *Backend (Laravel):*
  – PHP 8.0+
  – Composer 2.0+
  – MySQL 5.7+ lub MariaDB 10+
  – Node.js 16+ (dla narzędzi frontendowych)
- *Frontend (Vue 3 + Vite):*
  – Node.js 16+
  – npm 8+ lub yarn
  – Przeglądarka obsługująca nowoczesne standardy JavaScript (Chrome, Firefox, Edge)

4.2 Konfiguracja backendu (Laravel)

1. **Klonowanie repozytorium:**
   Pobierz kod źródłowy aplikacji z repozytorium:
   `git clone https://gitlab.ideo.pl/d.olko/shop-app-laravel-vue`
   Następnie przejdź do katalogu backend:
   `cd shop-app-laravel-vue /backend`

2. **Instalacja zależności PHP:**
   Zainstaluj zależności backendu przy użyciu Composera:
   `composer install`

3. **Konfiguracja pliku .env:**
   Skopiuj przykładowy plik konfiguracji środowiska:
   `cp .env.example .env`
   Następnie edytuj plik .env i ustaw poprawne wartości, np.:

```
DB_CONNECTION=mysql

DB_HOST=127.0.0.1

DB_PORT=3306

DB_DATABASE=shop_app

DB_USERNAME=twoj_uzytkownik

DB_PASSWORD=twoje_haslo
```

4. **Generowanie klucza aplikacji:**
   Wygeneruj klucz aplikacji za pomocą:
   `php artisan key:generate`

5. **Migracja bazy danych:**
   Aby utworzyć struktury tabel w bazie danych, wykonaj migrację (oraz seedowanie danych testowych, jeśli jest to wymagane):
   `php artisan migrate --seed`

6. **Uruchomienie serwera aplikacji:**
   Uruchom serwer developerski przy pomocy:

```
php artisan serve
```
Aplikacja będzie dostępna pod adresem: http://127.0.0.1:8000

4.3 Konfiguracja frontendu (Vue 3 + Vite)

1. **Przejście do katalogu frontendu:**
   ```
   cd ../frontend
   ```
2. **Instalacja zależności frontendowych:**
   Zainstaluj zależności, wykonując:
   ```
   npm install
   ```
3. **Uruchomienie aplikacji frontendowej:**
   Aby uruchomić aplikację w trybie deweloperskim, wpisz:
   ```
   npm run dev
   ```
   Aplikacja będzie dostępna pod adresem: http://localhost:5173
4. **Kompilacja produkcyjna:**
   Aby przygotować wersję produkcyjną aplikacji, wykonaj:
   ```
   npm run build
   ```

4.4 Testowanie API

Aby sprawdzić, czy backend działa poprawnie, można użyć Postman lub narzędzia curl. Na przykład, aby pobrać szczegóły produktu, wykonaj poniższe zapytanie:

```
curl -X GET "http://127.0.0.1:8000/api/products/1"
```

Przykładowa odpowiedź może wyglądać następująco:


```
{
  "id": 1,
  "title": "Product Example",
  "description": "Detailed description of the product.",
  "price": "99.99",
  "discount_percentage": "10.00",
  "rating": "4.50",
  "stock": 100,
  "brand": "BrandName",
  "category": "CategoryName",
  "thumbnail": "/storage/product1.jpg",
  "images": ["/storage/product1_1.jpg", "/storage/product1_2.jpg"],
  "created_at": "2025-02-04T09:11:06.000000Z",
  "updated_at": "2025-02-04T09:11:06.000000Z"
}
```

4.5 Uruchomienie aplikacji na serwerze produkcyjnym

Aby wdrożyć aplikację ShopApp na serwerze produkcyjnym, wykonaj następujące kroki:

1. Skonfiguruj środowisko Linux (np. Ubuntu 20.04+).
2. Zainstaluj serwer WWW (Nginx lub Apache) oraz PHP 8.0+.
3. Skopiuj pliki projektu na serwer i skonfiguruj plik .env z odpowiednimi danymi dostępowymi.
4. Wykonaj migracje bazy danych:

   ```
   php artisan migrate --seed
   ```

5. Skonfiguruj serwer tak, aby wskazywał na katalog public aplikacji Laravel.
6. Zbuduj frontend:

   ```
   cd frontend
   npm run build
   ```

5. Podsumowanie

Projekt aplikacji eCommerce ShopApp został zrealizowany przy użyciu nowoczesnych technologii backendowych i frontendowych, co umożliwiło stworzenie intuicyjnego, funkcjonalnego oraz responsywnego systemu zakupowego. Dzięki zastosowaniu Laravel jako backendu oraz Vue 3 z Vite na froncie, aplikacja łączy wydajność, skalowalność i przyjazny interfejs użytkownika, umożliwiając klientom łatwe przeglądanie produktów, dodawanie ich do koszyka, dokonywanie zakupów oraz zarządzanie swoim kontem.

5.1 Kluczowe funkcjonalności

Aplikacja umożliwia użytkownikom:

- Przeglądanie produktów z dynamicznym wyszukiwaniem – pole wyszukiwania z ikoną lupy umożliwia filtrowanie produktów w czasie rzeczywistym na podstawie nazwy, opisu, kategorii i marki.
- Rejestrację i logowanie – umożliwiające użytkownikom tworzenie konta oraz logowanie w celu korzystania z funkcji takich jak dodawanie produktów do koszyka, zakupy czy personalizacja ustawień.
- Dodawanie produktów do koszyka oraz finalizację zakupów – z możliwością zmiany ilości produktów oraz usuwania pozycji, a także przyciskiem "Buy Now", który przetwarza zakupy, zapisując dane w tabeli purchases i aktualizując stan magazynowy.
- Obsługę subskrypcji newslettera oraz wysyłanie formularza kontaktowego – formularze te są dostępne zarówno w stopce, jak i w dedykowanej stronie kontaktowej, a ich wysyłka wymaga zalogowania, co jest komunikowane użytkownikowi za pomocą modali.
- Zarządzanie kontem użytkownika – poprzez panel, w którym użytkownik może przeglądać historię swoich zakupów, edytować dane osobowe oraz zmieniać hasło.

5.2 Technologie wykorzystane w projekcie

W trakcie realizacji projektu ShopApp zastosowano następujące technologie:

- **Backend:** Laravel 11.4 (PHP 8+), MySQL, Eloquent ORM, Laravel Sanctum, Laravel Migrations.
- **Frontend:** Vue 3, Vite, Pinia, Vue Router, CSS (oraz elementy W3.css), FontAwesome.
- **Inne narzędzia:** Git, GitLab, Composer, npm.
- **API zewnętrzne:** Możliwe integracje z systemami płatności oraz zewnętrznymi usługami (np. dostawczymi).

5.3 Wnioski i dalsze możliwości rozwoju

Projekt ShopApp stanowi wydajny i intuicyjny system eCommerce, który umożliwia użytkownikom łatwe dokonywanie zakupów online. Kluczowe funkcjonalności, takie jak dynamiczne wyszukiwanie produktów, zaawansowana obsługa koszyka oraz panel użytkownika, zapewniają doskonałe doświadczenie zakupowe.
Możliwe dalsze rozszerzenia projektu obejmują:

- Rozbudowę funkcji wyszukiwania o zaawansowane filtry (np. zakres cenowy, oceny produktów).
- Integrację z systemami płatności online oraz usługami kurierskimi.
- Dodanie powiadomień push i alertów dotyczących promocji.
- Ulepszenia interfejsu mobilnego i wdrożenie dedykowanej aplikacji mobilnej.

5.4 Podsumowanie końcowe

Projekt ShopApp łączy nowoczesne technologie webowe i bazodanowe, tworząc skalowalny, responsywny i intuicyjny system eCommerce. Dzięki zastosowaniu Laravel i Vue 3, aplikacja oferuje użytkownikom bogate funkcjonalności, które można rozwijać i dostosowywać do przyszłych potrzeb rynkowych. Projekt został przetestowany pod kątem wydajności oraz użyteczności, a dalszy rozwój może obejmować kolejne usprawnienia, integracje oraz nowe funkcjonalności, które jeszcze bardziej poprawią doświadczenie zakupowe użytkowników.