# Sklep internetowy (z modułem koszyk)

**Do zaliczenia nie jest wymagane zrobienie wszystkich wymienionych rzeczy. Robimy tyle ile potrafimy/możemy, ew. pytamy. Każdy może zrobić dużo więcej sugerując się featurami z przykładowego sklepu.**

Aplikacja może być rozbita na dowolna ilość komponentów. Aplikacja powinna wspierać routing w Angular. Do komunikacji z serwisami należy zdefiniować klasy/interfejs Product, Basket... z odpowiednimi polami. Wszystkie serwisy powinny być jawnie zarejestrowane w module (providers) z możliwością zmiany ich działania z pamięci na API (Dependency Injection, useClass). Style aplikacji mogą być globalne lub w komponencie, jednak powinny wyświetlać spójne UI.

Przykład dostępny pod adresem: https://colorlib.com/preview/theme/store/shop.html
Szablon strony (może być wykorzystany, ale nie musi): https://colorlib.com/wp/template/store/

Funkcje do zaimplementowania:

- Lista produktów pod adresem skonfigurowanym w Angular '/products'.
  - Lista zawiera nazwę, zdjęcie (adres url zdjęcia) i cenę produktu (liczba w klasie).
  - Dane mogą być podawane od API, pliku json lub w postaci obiektów w kodzie serwisu.
  - Metoda zwracająca dane powinna być umieszczona w Angular Service. Zwracać obiekt Observable<Product> lub Promise<Product>.
  - Przycisk dodaj do koszyka/kup.
  - Link z przejściem do szczegółów produktu.
  - Nazwa produktu powinna być skracana do 20 znaków z użyciem Pipe.
  - Cena produktu (liczba) powinna być formatowana do kwoty z waluta z użyciem Pipe.
  - Lista może mieć strony, **ale nie musi**.
  - Lista może mieć jakiś podstawowy filtr danych (nazwa, cena), **ale nie musi**.
  - Parametry takie jak numer strony i podstawowe filtry moga być parametrami routingu strony '/products' **(opcjonalne)**.

Przykład listy: https://colorlib.com/preview/theme/store/shop.html

- Produkt szczegółowy pod adresem skonfigurowanym w Angular 'products/:id', gdzie id to liczba lub unikalna nazwa.
  - Wyświetlamy nazwę produktu.
  - Wyświetlamy opis (może być HTML).
  - Wybór liczby elementów do koszyka (licznik).
  - Przycisk dodaj do koszyka/kup. Dodaj do koszyka pozostawia nas na szczegółach produktu. Wyświetla informacje o dodaniu do koszyka (toast lub message, dowolna biblioteka).
  - Liczba dodanych elementów do koszyka wyświetlana jest na górze strony jako oddzielny komponent BasketCounterComponent (ikonka koszyka i liczba).
  - Wejście na stronę produktu o nazwie, którego nie ma na liscie produktów powinno wrócić nas do listy produktów (np. adres 'products/lalalala').
  - Nad zdjęciem produktu możemy wyświetlać badge (promocja lub nowość). Powinien być zrealizowany za pomoca dyrektywy (losowa decyzja czy wyświetlać badge).

Przykład szczegółów produktu: https://colorlib.com/preview/theme/store/product-detail.html

- Koszyk z wybranymi produktami pod adresem skonfigurowanym w Angular '/cart'.
  - Wyświetla listę produktów dodanych do koszyka.
  - Zawiera nazwę, cenę, wybór ilości elementów, cenę razem.
  - Przycisk usuń z koszyka, usuwa produkt z listy.
  - Zmiana liczby elementów automatycznie przelicza koszyk.
  - Przycisk kup powoduje wyświetlenie okna modalnego z podziękowaniem za zakup i wyczyszczenie koszyka.

Przykład koszyka: https://colorlib.com/preview/theme/store/cart.html

Elementy z Angular:

- Routing
- Routing Guard do sprawdzania czy nazwa produktu istnieje
- Pipes do skracania nazwy I formatowania ceny
- ProductsApiService (do pobierania produktów, **jeśli potrzebne**)
- lub ProductsInMemoryService (do pobierania produktów z pamięci, **jeśli potrzebne**)
- BasketApiService (do operowania na koszyku z API z projektu w Node.js, **jeśli potrzebne**)
- lub BasketLocalStorageService (do operowania na koszyku w local storage przegladarki, **opcjonalne**)
- lub BasketInMemoryService (do operowania na koszyku w pamięci js, **jeśli potrzebne**)