Dokumentacja projektu e-commerce

Nazwa projektu: **e-commerce**

Przedmiot: **Języki programowania JavaScript, studia niestacjonarne, III rok, V semestr**

Autorzy: **Anna Sroka, Barbara Sławińska, Justyna Szofińska, Paweł Staniul**

1. Specyfikacja

1.1. Opis aplikacji

Projekt e-commerce to aplikacja webowa dla sklepów internetowych, która umożliwia użytkownikom zakup produktów online. Składa się z dwóch głównych komponentów: frontendu zbudowanego w Angularze oraz backendu napisanego w Node.js. Aplikacja integruje się z bazą danych MySQL, co umożliwia skuteczne zarządzanie zamówieniami, produktami i danymi użytkowników.

Aplikacja ma na celu zapewnienie sprawnego i intuicyjnego procesu zakupowego, od przeglądania produktów przez użytkownika po zarządzanie zamówieniami i dostawą przez administrację sklepu. Dzięki zastosowaniu narzędzi Angular Material oraz Node.js, interfejs użytkownika jest przyjazny i responsywny, a backend API jest bezpieczny i skalowalny.

Główne cechy aplikacji obejmują:

- Rejestrację i logowanie użytkowników z funkcją danych w czasie rzeczywistym.
- Przeglądanie, sortowanie i filtrowanie katalogu produktów.
- Dodawanie produktów do koszyka oraz składanie zamówień.
- Monitorowanie statusu zamówień w historii zakupów.
- Zarządzanie produktami i użytkownikami w panelu administracyjnym.

1.2. User Stories

- Jako klient, chcę móc się zarejestrować i zalogować, aby móc dokonywać zakupów oraz śledzić historię swoich zamówień.
  - **Rejestracja:** Użytkownik może utworzyć konto, podając swoje dane osobowe oraz adres email.
  - **Logowanie:** Użytkownik może się zalogować używając swojego adresu email i hasła.

- Jako klient, chcę przeglądać produkty, aby wybrać te, które chcę kupić.
  - **Przeglądanie produktów:** Użytkownik może przeglądać listę produktów, filtrując je według kategorii, ceny lub innych parametrów.

- Jako klient, chcę dodawać produkty do koszyka, aby móc je później kupić.
  - **Zarządzanie koszykiem:** Użytkownik może dodawać produkty do koszyka, zmieniać ilość produktów w koszyku oraz usuwać je z koszyka.

- Jako klient, chcę składać zamówienie, aby otrzymać wybrane produkty.
  - **Składanie zamówienia:** Użytkownik po dodaniu produktów do koszyka może przejść do realizacji zamówienia, wybierając sposób dostawy oraz metodę płatności.

- Jako administrator, chcę zarządzać listą produktów, aby aktualizować informacje o produktach i dostępności.
  - **Zarządzanie produktami:** Administrator może dodawać nowe produkty, edytować istniejące (ceny, opisy, zdjęcia) oraz usuwać te, które nie są już dostępne.

- Jako administrator, chcę mieć możliwość przeglądania i zarządzania zamówieniami, aby kontrolować proces sprzedaży i dostaw.
  - **Zarządzanie zamówieniami:** Administrator ma dostęp do listy wszystkich zamówień, może zmieniać ich status (np. potwierdzone, wysłane, zrealizowane), oraz monitorować ich realizację.

1.3. Odbiorcy systemu

- **Klienci sklepu internetowego** - osoby szukające wygodnego i szybkiego sposobu na zakup produktów online, ceniące sobie szeroki wybór i łatwość dokonywania transakcji.
- **Administratorzy sklepu** - osoby odpowiedzialne za zarządzanie asortymentem produktów, procesami sprzedaży oraz obsługą klienta w sklepie internetowym.

1.4. Potencjale korzyści dla użytkowników

- **Wygodne zakupy online** - Użytkownicy mogą w łatwy sposób przeglądać i kupować produkty z dowolnego miejsca i o każdej porze, co jest wygodne i oszczędza czas.
- **Dostęp do historii zakupów i statusu zamówień** - Klienci mają możliwość śledzenia swoich zamówień i przeglądania historii zakupów, co ułatwia zarządzanie ich zakupami oraz planowanie przyszłych.
- **Ułatwione zarządzanie sklepem** - Administratorzy otrzymują narzędzia do efektywnego zarządzania produktami, zamówieniami i klientami, co może przyczynić się do lepszego dostosowania oferty do potrzeb rynku i zwiększenia sprzedaży.

W warstwie backend node.js wykorzystano zależności, tzw. dependencies:

- **bcryptjs:** Służy do hashowania haseł użytkowników, zwiększając bezpieczeństwo poprzez uniemożliwienie odczytu haseł w ich pierwotnej formie. Pozwala również na porównanie wprowadzonego hasła z hashem zapisanym w bazie danych.

- **body-parser:** Analizuje zawartość żądań HTTP, szczególnie w formacie JSON, i umożliwia aplikacji dostęp do danych przesyłanych w żądaniach `POST` lub `PUT`. Zapewnia łatwy dostęp do danych przesyłanych w formularzach i poprzez API.

- **cors:** Pozwala na konfigurowanie polityki CORS, umożliwiając aplikacji backendowej zezwalanie na dostęp do swoich zasobów z innych domen. Jest niezbędny do zabezpieczenia komunikacji między front-endem a back-endem.

- **dotenv:** Umożliwia zarządzanie konfiguracją aplikacji przez ładowanie zmiennych środowiskowych z pliku `env`. Zapewnia bezpieczeństwo danych poprzez oddzielenie ustawień od kodu źródłowego.

- **express:** Popularny framework do tworzenia aplikacji webowych w Node.js, zapewniający minimalistyczny i elastyczny zestaw narzędzi do zarządzania żądaniami HTTP i definiowania tras w aplikacji.

- **helmet:** Zestaw zabezpieczeń zwiększający bezpieczeństwo aplikacji webowych poprzez konfigurowanie odpowiednich nagłówków HTTP. Pomaga chronić przed typowymi zagrożeniami, jak ataki XSS.

- **jsonwebtoken:** Biblioteka do generowania i weryfikacji tokenów JWT (JSON Web Token), używanych głównie do autoryzacji i uwierzytelniania użytkowników w aplikacjach webowych.

- **multer:** Middleware do obsługi przesyłania plików, pozwalający na dodawanie załączników i zapisywanie ich na serwerze lub w bazie danych. Umożliwia zarządzanie typami plików oraz lokalizacją ich przechowywania.

- **mysql2:** Nowoczesny klient MySQL w Node.js, kompatybilny z biblioteką `mysql`, oferujący lepszą wydajność, asynchroniczne API i obsługę promes. Pozwala na łatwe zarządzanie połączeniami z bazą danych MySQL.

- **sequelize:** ORM (Object-Relational Mapping) pozwalający na łatwe tworzenie i zarządzanie modelami danych w bazach relacyjnych. Umożliwia pracę z różnymi typami baz danych, wspierając zarówno język SQL, jak i modele obiektowe w kodzie.