Dokumentacja Projektowa: Moduł Sklepowy Dobrego Swiata

Bazy danych i systemy informacyjne

Krzysztof Zając

Styczeń 2026

1 Cel aplikacji i charakterystyka użytkowników

Celem projektu jest rozbudowa istniejącej aplikacji webowej o moduł sklepu internetowego. System backendowy zostanie zaimplementowany w technologii FastAPI (Python) z wykorzystaniem biblioteki SQLAlchemy jako ORM oraz bazy danych **MariaDB**.

1.1 Rodzaje użytkowników i uprawnienia

- **Klient**: Może przeglądać produkty, filtrować je według kategorii, składać zamówienia oraz wystawiać opinie. Ma dostęp do historii własnych zamówień.
- **Administrator**: Posiada pełne uprawnienia do zarządzania asortymentem (CRUD produktów i kategorii) oraz podglądu wszystkich zamówień w systemie.

2 Modelowanie Logiczne

Baza danych została zaprojektowana w oparciu o model relacyjny, zachowując III Formę Normalną (3NF).

2.1 Opis encji i pól (Tabele)

1. Tabela: `users` (Użytkownicy)
Przechowuje dane kont użytkowników. Hasła są przechowywane w formie zahashowanej.

Pole	Typ	Opis
id (PK)	Integer	Identyfikator użytkownika
email	Varchar(255)	Adres e-mail (Unique, Index)
hashed_password	Varchar(255)	Skrót kryptograficzny hasła
role	Enum	Rola: 'client' lub 'admin'

2. Tabela: `categories` (Kategorie)
Służy do grupowania produktów.

Pole	Typ	Opis
id (PK)	Integer	Identyfikator kategorii
name	Varchar(100)	Nazwa kategorii (Unique)

3. Tabela: `products` (Produkty)
Zawiera informacje o asortymencie i stanach magazynowych.

Pole	Typ	Opis
id (PK)	Integer	Identyfikator produktu
name	Varchar(255)	Nazwa produktu
price	Decimal(10,2)	Cena jednostkowa
stock_quantity	Integer	Ilość sztuk w magazynie
category_id (FK)	Integer	Powiązanie z tabelą categories

4. Tabela: `orders` (Zamówienia)
Nagłówek zamówienia zawierający dane o kliencie i statusie.

Pole	Typ	Opis
id (PK)	Integer	Numer zamówienia
user_id (FK)	Integer	Klient składający zamówienie
status	Varchar(50)	Status (np. 'pending', 'shipped')
created_at	Timestamp	Data złożenia zamówienia

5. Tabela: `order_items` (Pozycje zamówienia)
Tabela łącząca zamówienia z produktami (relacja wiele-do-wielu).

Pole	Typ	Opis
id (PK)	Integer	Identyfikator pozycji
order_id (FK)	Integer	Powiązanie z zamówieniem
product_id (FK)	Integer	Powiązanie z produktem
quantity	Integer	Zamówiona ilość
price_at_purchase	Decimal(10,2)	Cena w momencie zakupu

6. Tabela: `reviews` (Opinie)
Oceny produktów wystawiane przez użytkowników.

Pole	Typ	Opis
id (PK)	Integer	Identyfikator opinii
product_id (FK)	Integer	Oceniany produkt
user_id (FK)	Integer	Autor opinii
rating	Integer	Ocena w skali 1-5

2.2 Związki między encjami

- **Kategorie – Produkty (1:N)**: Jedna kategoria może zawierać wiele produktów.
- **Użytkownicy – Zamówienia (1:N)**: Jeden użytkownik może złożyć wiele zamówień.
- **Zamówienia – Produkty (N:M)**: Zrealizowane poprzez tabelę pośredniczącą `order_items`.
- **Produkty – Opinie (1:N)**: Jeden produkt może posiadać wiele opinii.

3 Logika Bazy Danych i Bezpieczeństwo

3.1 Planowane Triggery i Procedury

W celu zapewnienia spójności danych po stronie MariaDB, zaplanowano:

- **Trigger** `after_order_item_insert`: Po dodaniu rekordu do `order_items`, trigger automatycznie pomniejsza wartość `stock_quantity` w tabeli `products`.
- **Procedura** `CalculateUserTotal`: Procedura przyjmująca `user_id` i zwracająca łączną sumę wydaną przez klienta na wszystkie zamówienia.

4 Formalna Analiza Normalizacji (3NF)

Baza danych spełnia warunki Trzeciej Formy Normalnej (3NF).

4.1 Definicje i Zależności Funkcyjne (FD)

Niech R będzie relacją, a X, Y podzbiorami jej atrybutów. Zależność funkcyjną oznaczamy jako $X \to Y$.

4.1.1 Pierwsza Forma Normalna (1NF)

Wszystkie atrybuty w tabelach są atomowe (niepodzielne), a każda tabela posiada zdefiniowany klucz podstawowy[cite: 16]. Relacje nie zawierają powtarzających się grup atrybutów.

4.1.2 Druga Forma Normalna (2NF)

Relacja jest w 2NF, gdy jest w 1NF i każdy atrybut niekluczowy jest w pełni zależny funkcyjnie od całego klucza głównego.

- W tabelach `users`, `products`, `categories`, `orders` oraz `reviews` klucz główny jest jednoelementowy (id). W takim przypadku częściowa zależność od klucza jest niemożliwa.
- W tabeli `order_items`, mimo posiadania kluczy obcych, jako PK wybrano sztuczny klucz id, co zapewnia spełnienie 2NF.

4.1.3 Trzecia Forma Normalna (3NF)

Relacja spełnia 3NF, jeśli jest w 2NF i żaden atrybut niekluczowy nie jest zależny przechodnio od klucza głównego (brak zależności $PK \to X \to Y$).

Dowód na przykładzie tabeli `products`:

- Atrybuty: $A = \{id, name, price, stock\_quantity, category\_id\}$.
- Zależności: $id \to name, id \to price, id \to stock\_quantity, id \to category\_id$.
- Ponieważ $category\_id$ jest kluczem obcym prowadzącym do innej relacji, a atrybuty takie jak $name$ czy $price$ nie determinują siebie nawzajem (np. $name \not\to price$), brak jest zależności przechodnich.

4.2 Uzasadnienie dopuszczalnej redundancji

W tabeli `order_items` zdecydowano się na przechowywanie atrybutu `price_at_purchase`.

- **Uzasadnienie:** Choć cena znajduje się w tabeli `products`, ulega ona zmianom w czasie. Przechowywanie ceny w momencie zakupu jest niezbędne dla zachowania historycznej spójności finansowej zamówień i nie stanowi błędu redundancji, lecz wymóg spójności biznesowej.

4.3 Operacje CRUD i Transakcje

Wszystkie operacje składania zamówienia (dodanie rekordu do `orders` oraz wielu do `order_items`) będą wykonywane wewnątrz **transakcji**. Gwarantuje to, że jeśli zabraknie towaru w magazynie dla jednej z pozycji, całe zamówienie zostanie wycofane (Rollback).

4.4 Bezpieczeństwo

- Wykorzystanie mechanizmów SQLAlchemy (Bound Parameters) chroni aplikację przed atakami typu **SQL Injection**.
- Dostęp do bazy danych dla aplikacji FastAPI realizowany jest poprzez dedykowanego użytkownika technicznego z ograniczonymi uprawnieniami.