**Opis projektu sklepu internetowego sport3000**

**Cel:**

Celem projektu jest stworzenie internetowego sklepu specjalizującego się w sprzedaży sprzętu narciarskiego. Strona ma umożliwiać użytkownikom łatwe przeglądanie i zakup różnych marek sprzętu narciarskiego, takich jak narty, buty narciarskie, kijki, kaski i gogle. Sklep będzie dostarczał bogate informacje o produktach, oferował system opinii użytkowników oraz narzędzia do filtrowania produktów według marek, rozmiarów i cen. Dodatkowo, strona będzie zintegrowana z systemami płatności oraz systemem wysyłki.

**Zakres:**

* **Frontend:** Tworzenie interaktywnego interfejsu użytkownika za pomocą Vue.js, który umożliwia szybkie i płynne przeglądanie produktów, dodawanie ich do koszyka oraz składanie zamówień.
* **Backend:** Zbudowany w Spring Boot, obsługujący funkcjonalność sklepu, autoryzację użytkowników, przetwarzanie zamówień oraz integrację z bazą danych i zewnętrznymi systemami (np. system płatności online, zarządzanie stanem magazynowym).
* **Baza danych:** Przechowywanie danych o produktach, użytkownikach, zamówieniach oraz stanach magazynowych.
* **Integracje:** Z systemami płatności online oraz firmami kurierskimi.
* **Zarządzanie użytkownikami:** Możliwość zakładania kont, logowania się, przeglądania historii zamówień oraz zarządzania danymi konta.

**1. Analiza wymagań**

**1.1.** **Opis działania i schemat logiczny systemu**

**System będzie działał zgodnie z następującym schematem:**

1. Rejestracja i logowanie użytkowników: Użytkownicy będą mogli założyć konto lub zalogować się za pomocą istniejących kont. Po rejestracji będą mieli możliwość dodawania różnych adresów zamieszkania, zarządzania swoimi danymi oraz przeglądania historii zamówień.
2. Przeglądanie i wyszukiwanie produktów: Użytkownicy będą mogli przeglądać katalog produktów oraz korzystać z zaawansowanych filtrów według marki, typu sprzętu, ceny, rozmiaru oraz sortowania.
3. Dodawanie do koszyka i składanie zamówień: Po wybraniu produktów użytkownik będzie mógł dodać je do koszyka, dokonać płatności oraz wybrać metodę wysyłki.
4. Recenzje i oceny: Użytkownicy będą mogli oceniać i recenzować zakupione produkty, co poprawi jakość rekomendacji dla innych klientów.
5. Zarządzanie magazynem: Po złożeniu zamówienia stany magazynowe będą automatycznie aktualizowane.
6. Integracja z zewnętrznymi systemami: System będzie zintegrowany z usługami płatności (np. PayPal, Stripe) oraz firmami kurierskimi (np. DHL, UPS), co zapewni bezproblemowe płatności oraz wysyłki.

**1.2**. **Wymagania funkcjonalne**

1. R**ejestracja użytkownika i logowanie:**
   * Użytkownik musi mieć możliwość rejestracji konta, a następnie logowania się na stronie.
   * Możliwość dodania różnych adresów wysyłki oraz zarządzania danymi konta.
2. **Przeglądanie produktów i filtrowanie:**
   * Wyszukiwanie produktów za pomocą wyszukiwarki oraz zaawansowanych filtrów (marka, rozmiar, typ sprzętu, cena).
   * Możliwość sortowania produktów (po cenie, popularności, dostępności).
3. **Koszyk i składanie zamówienia:**
   * Użytkownik może dodawać produkty do koszyka, wybierać opcje wysyłki oraz metodę płatności.
   * Możliwość podsumowania zamówienia przed finalizacją.
4. **Zarządzanie magazynem:**
   * System automatycznie aktualizuje stany magazynowe po złożeniu zamówienia.
5. **Integracja z płatnościami i wysyłką:**
   * Płatności online poprzez integrację z PayPal, Stripe lub innymi dostawcami.
   * Integracja z firmami kurierskimi w celu zarządzania wysyłką.
6. **Recenzje i oceny produktów:**
   * Użytkownicy mogą dodawać recenzje zakupionych produktów, oceniać je oraz przeglądać opinie innych użytkowników.

**1.3. Wymagania niefunkcjonalne**

**1.3.1.** **Wykorzystywane technologie i narzędzia**

* **Frontend**: Vue.js 3.2, HTML5, CSS3, JavaScript (ES6+) – Vue.js w wersji 3.2 zapewni dynamiczny i interaktywny interfejs użytkownika. HTML5 i CSS3 zostaną użyte do budowy oraz stylizacji komponentów, a JavaScript (ES6+) umożliwi zaawansowaną logikę aplikacji.
* **Backend**: Spring Boot 3.1 (Java 17), REST API, Spring Security 6.0 – Spring Boot 3.1 wraz z Javą 17 zapewni elastyczność i bezpieczeństwo aplikacji backendowej. REST API będzie odpowiedzialne za komunikację między frontendem a backendem, a Spring Security 6.0 zapewni bezpieczną autoryzację i uwierzytelnianie użytkowników.
* **Baza danych**: MySQL 8.0 lub PostgreSQL 15.1, Hibernate ORM 5.6 – System zarządzania bazą danych, taki jak MySQL 8.0 lub PostgreSQL 15.1, będzie używany do przechowywania danych aplikacji, a Hibernate ORM 5.6 umożliwi efektywne zarządzanie i mapowanie obiektowo-relacyjne.
* **Integracje**: PayPal API v2, Stripe API v2020-08-27, DHL API v3, UPS API v1 – Zewnętrzne integracje umożliwią bezpieczne przetwarzanie płatności oraz efektywną obsługę logistyki i wysyłek. Integracje te będą obsługiwać transakcje oraz monitorowanie przesyłek.
* **Infrastruktura**: Serwery VPS, AWS (Amazon Web Services) lub Google Cloud Platform (GCP) – Hosting i infrastruktura chmurowa zapewnią elastyczność i skalowalność, a serwery VPS będą używane do przetwarzania aplikacji i zarządzania danymi w trybie produkcyjnym.

**1.3.2.** **Wymagania dotyczące rozmiaru bazy danych**

System musi być w stanie obsłużyć przynajmniej kilka tysięcy produktów, użytkowników oraz zamówień. Oczekuje się, że baza danych będzie się dynamicznie rozrastać w miarę wzrostu sprzedaży. Początkowo system powinien obsłużyć:

* **3,000 produktów.**
* **10,000 użytkowników.**
* **50,000 zamówień rocznie.**

**1.3.3.** **Wymagania dotyczące bezpieczeństwa systemu**

* Szyfrowanie danych użytkowników: Stosowanie SSL do szyfrowania danych przesyłanych między użytkownikami a serwerem.
* Ochrona danych osobowych: Zgodność z przepisami o ochronie danych (np. RODO), w tym prawa do bycia zapomnianym.
* Bezpieczna autoryzacja: Zastosowanie bezpiecznych mechanizmów autoryzacji użytkowników (OAuth 2.0, JWT).
* Bezpieczne przetwarzanie płatności: Integracja z zaufanymi systemami płatności zewnętrznych (np. PayPal, Stripe).

**1.4. Przyjęte założenia projektowe**

1. Skalowalność: System powinien być skalowalny, tak aby mógł obsługiwać rosnącą liczbę użytkowników, produktów i zamówień bez spadku wydajności.
2. Responsywność: Strona musi być w pełni responsywna i dobrze działać na urządzeniach mobilnych, aby zapewnić wygodę użytkowania na różnych rozdzielczościach ekranu.
3. Wydajność: Czas ładowania strony powinien być możliwie najkrótszy. Optymalizacja bazy danych oraz mechanizmy cache (np. Redis) są wymagane dla utrzymania wysokiej wydajności.
4. Utrzymanie: System musi być łatwy w utrzymaniu i umożliwiać łatwe wdrażanie aktualizacji. Wymagana jest dobra dokumentacja kodu oraz infrastruktury.
5. Elastyczność: System powinien być elastyczny pod względem integracji z nowymi systemami płatności i logistycznymi w przyszłości.

**2. SWOT Analysis**

**Siły (Strengths):**

* **Szeroka oferta produktów:** Bogaty wybór marek sprzętu narciarskiego, umożliwiający przyciągnięcie różnych grup klientów.
* **Niche marketing:** Skupienie się wyłącznie na sprzęcie narciarskim, co buduje wizerunek eksperta w tej dziedzinie.
* **Personalizacja doświadczeń:** Możliwość filtrowania produktów według specyficznych parametrów (np. rozmiar, marka, cena), co poprawia doświadczenie użytkownika.
* **Łatwość nawigacji:** Przyjazny UX stworzony z pomocą Vue.js, który zapewnia szybkie ładowanie i responsywność strony.
* **Znajomość technologii:** Duża ilość doświadczenia praktycznego we Vue.js, pozyskanego podczas pracy zarobkowej.

**Słabości (Weaknesses):**

* **Brak odzieży:** Brak pełnej oferty narciarskiej może zmniejszyć atrakcyjność sklepu dla klientów, którzy poszukują kompletnego wyposażenia.
* **Koszt utrzymania systemu:** Potrzeba integracji z różnymi systemami zewnętrznymi (płatności, wysyłka) zwiększa koszty utrzymania sklepu.
* **Potencjalna konkurencja:** Wysokie koszty marketingu i pozycjonowania w wyszukiwarkach mogą być wyzwaniem na bardzo konkurencyjnym rynku e-commerce.
* **Brak pozycjonowania:** Niedostosowanie witryny internetowej zgodnie z najlepszymi praktykami SEO.

**Szanse (Opportunities):**

* **Rośnie popularność zakupów online:** Zwiększająca się liczba klientów dokonujących zakupów online to szansa na wzrost sprzedaży.
* **Sezonowe promocje:** Możliwość organizowania sezonowych kampanii marketingowych (np. Black Friday, wyprzedaże po sezonie zimowym).
* **Integracje z platformami społecznościowymi:** Ułatwienie klientów w dotarciu do sklepu poprzez działania promocyjne w social media.

**Zagrożenia (Threats):**

* **Zmienne warunki rynkowe:** Konkurencja ze strony dużych firm, takich jak Amazon czy inne sklepy sportowe, które oferują darmową dostawę lub agresywne promocje.
* **Sezonowość:** Sklep narciarski ma naturalny cykl sprzedaży, gdzie poza sezonem zimowym liczba zamówień może spadać.
* **Zmiany w zachowaniach konsumentów:** Przesunięcie trendów w kierunku bardziej ekologicznego sprzętu lub nowoczesnych technologii, co wymaga ciągłego dostosowywania oferty
* **Ataki hakerskie:** Jeden z ataków polega na wstrzyknięciu złośliwych zapytań SQL do aplikacji, które mogą dać dostęp do bazy danych. Wymagana ochrona: walidacja zapytań, korzystanie z ORM (np. Hibernate). Kolejnym bardzo łatwym do wykonania jest atak DDoS (Distributed Denial of Service): Ataki polegające na zalewaniu serwera żądaniami, co powoduje spadek wydajności systemu. Wymagana ochrona: zastosowanie firewalli oraz monitoringu ruchu.

**3. Krytyczne czynniki sukcesu (CSFs - Critical Success Factors):**

* **Sprawne działanie platformy:** Utrzymanie płynnego działania strony, szybkiego ładowania oraz bezbłędnego przetwarzania zamówień i płatności.
* **Efektywny UX/UI:** Intuicyjna i szybka nawigacja po stronie, która minimalizuje czas potrzebny na zakup i zapewnia komfort użytkownikowi.
* **Bezpieczne transakcje:** Bezpieczeństwo danych użytkowników oraz integracja z wiarygodnymi dostawcami usług płatności.
* **Zarządzanie magazynem i logistyką:** Automatyczne śledzenie stanów magazynowych i szybkie dostawy dla zwiększenia zadowolenia klientów.
* **Marketing i pozycjonowanie:** Odpowiednie działania marketingowe, kampanie reklamowe w Google oraz mediach społecznościowych, aby zdobywać nowych klientów.

**4. Otoczenie podsystemu**

* **Frontend (Vue.js 3.2):** Vue.js 3.2 zapewni interaktywny interfejs użytkownika oraz dynamiczne ładowanie treści, umożliwiając tworzenie responsywnych aplikacji internetowych. Dzięki komponentowej architekturze Vue.js możliwa będzie łatwa integracja z backendem, co umożliwi bezproblemowe operacje, takie jak dodawanie produktów do koszyka i składanie zamówień. HTML5, CSS3, oraz JavaScript (ES6) będą wykorzystywane do budowy i stylizacji elementów strony.
* **Backend (Spring Boot 3.1):** Spring Boot 3.1 odpowiada za logikę biznesową, autoryzację użytkowników, przetwarzanie zamówień oraz połączenie z bazą danych. Dzięki Spring Boot możliwe jest łatwe tworzenie REST API oraz integracja z systemami zewnętrznymi. Aplikacja backendowa będzie oparta na Javie 17, która oferuje nowoczesne funkcje i długoterminowe wsparcie, zapewniając stabilność i wydajność rozwiązania.
* **Baza danych (MySQL 8.0 lub PostgreSQL 15.1):** Baza danych SQL, taka jak MySQL 8.0 lub PostgreSQL 15.1, zostanie wykorzystana do zarządzania informacjami o produktach, zamówieniach oraz użytkownikach. System zarządzania relacyjną bazą danych utrzyma integralność danych i efektywność działania aplikacji, wspierając przetwarzanie zapytań i transakcji na dużą skalę.
* **Zewnętrzne integracje:** Aplikacja będzie zintegrowana z systemami płatności online, takimi jak PayPal API (v2) i Stripe API (v2020-08-27), aby umożliwić bezpieczne przetwarzanie transakcji. Dodatkowo zostaną zaimplementowane integracje z usługami kurierskimi, np. DHL API (v3) i UPS API (v1), aby umożliwić śledzenie zamówień i optymalizację dostaw. MailChimp API (v3.0) zostanie zintegrowane w celu wsparcia działań marketingowych i zarządzania relacjami z klientami (CRM).

**5. Kosztorys projektu sklepu internetowego sport3000**

**Kosztorys został rozbity na kluczowe obszary projektu, uwzględniając rozwój, integracje zewnętrzne, infrastrukturę oraz marketing. Dzięki szczegółowej analizie łatwiej będzie zidentyfikować obszary wymagające większych nakładów finansowych oraz przewidzieć ogólny koszt projektu.**

**1. Koszty zespołu developerskiego**

1. **Backend Development (Spring Boot, Java)**
   * **Programowanie funkcji sklepu (rejestracja, logowanie, zarządzanie produktami, koszyk, zamówienia)**
   * **Integracja z bazą danych oraz systemami zewnętrznymi (płatności, dostawa)**
   * **Optymalizacja i testowanie backendu**
   * **Koszt: 20 000 - 30 000 PLN**
2. **Frontend Development (Vue.js)**
   * **Tworzenie interaktywnego UI, zgodnie z wytycznymi UX/UI**
   * **Integracja frontend z backendem (komunikacja przez API REST)**
   * **Implementacja funkcji filtrowania, sortowania, recenzji oraz koszyka**
   * **Testy responsywności i optymalizacja wydajności na różnych urządzeniach**
   * **Koszt: 15 000 - 25 000 PLN**
3. **Projektowanie UX/UI**
   * **Opracowanie layoutu i elementów interfejsu użytkownika**
   * **Testy użyteczności (np. A/B testy) i optymalizacja UX na podstawie opinii testujących**
   * **Koszt: 5 000 - 10 000 PLN**
4. **Zarządzanie projektem i testowanie**
   * **Planowanie pracy zespołu, monitorowanie postępu i budżetu**
   * **Testy manualne oraz automatyczne w celu zapewnienia bezbłędnego działania systemu**
   * **Koszt: 8 000 - 12 000 PLN**

**2. Koszty infrastruktury i utrzymania**

1. **Hosting serwera i infrastruktura chmurowa (np. AWS, Google Cloud)**
   * **Serwery dla backendu i bazy danych oraz przestrzeń dla plików statycznych (zdjęcia produktów, recenzje)**
   * **Rozliczenie miesięczne lub roczne (w zależności od dostawcy) w zależności od obciążenia serwera**
   * **Koszt: 5 000 - 10 000 PLN rocznie**
2. **Baza danych (np. MySQL lub PostgreSQL)**
   * **Koszt bazy danych w chmurze lub na serwerze własnym w zależności od wyboru infrastruktury**
   * **Koszt: 2 000 - 5 000 PLN rocznie**
3. **Licencje na narzędzia monitorujące**
   * **Monitorowanie wydajności serwera i bazy danych, bezpieczeństwa aplikacji (np. New Relic, Sentry)**
   * **Koszt: 1 000 - 3 000 PLN rocznie**

**3. Koszty integracji zewnętrznych**

1. **Systemy płatności online (np. PayPal, Stripe)**
   * **Koszt początkowej integracji oraz opłaty prowizyjne (zwykle 1-3% wartości transakcji)**
   * **Koszt integracji: 3 000 - 5 000 PLN**
   * **Prowizja: Zależna od wolumenu transakcji**
2. **Systemy wysyłkowe (np. DHL, UPS)**
   * **Koszt integracji API z systemem śledzenia i zarządzania wysyłkami**
   * **Koszt integracji: 2 000 - 4 000 PLN**

**4. Koszty marketingu i promocji**

1. **SEO i optymalizacja strony pod kątem wyszukiwarek**
   * **Optymalizacja treści, kampanie Google Ads, pozycjonowanie na wybrane słowa kluczowe**
   * **Koszt: 5 000 - 10 000 PLN**
2. **Reklama w mediach społecznościowych (np. Facebook, Instagram)**
   * **Kampanie reklamowe skierowane na klientów zainteresowanych sprzętem narciarskim**
   * **Koszt: 5 000 - 8 000 PLN miesięcznie**
3. **E-mail marketing i CRM**
   * **Narzędzia do zarządzania relacjami z klientem, automatyzacja wysyłki e-maili promocyjnych**
   * **Koszt: 1 000 - 2 000 PLN rocznie**

**Rezerwa na nieprzewidziane koszty**

**Koszt: 5 000 PLN**

**Łączny koszt projektu: 65 000 - 100 000 PLN**

**\**

**6. Wstępne założenia dotyczące funkcjonalności strony:**

1. **Rejestracja i logowanie użytkowników:**
   * Tworzenie konta, zarządzanie danymi konta wprowadzenie różnych adresów zamieszkania oraz odbiorców.
2. **Wyszukiwanie i filtrowanie produktów:**
   * Wyszukiwarka, filtry według marki, typu sprzętu, ceny, rozmiaru oraz sortowanie wyświetlanych produktów.
3. **Koszyk i proces składania zamówienia:**
   * Dodawanie do koszyka, podsumowanie zamówienia, wybór metody płatności i dostawy.
4. **Recenzje i oceny produktów:**
   * Użytkownicy mogą oceniać i recenzować zakupione produkty poprzez napisanie własnej opinii.
5. **Zarządzanie magazynem:**
   * Automatyczne aktualizowanie stanów magazynowych po złożeniu zamówienia.
6. **Integracja z systemami płatności i wysyłki.**

**Model relacyjny**

1. **Tabela: Users (użytkownicy)**
   * **user\_id** (INT, PK, AUTO\_INCREMENT): ID użytkownika.
   * **email** (VARCHAR(255), UNIQUE): E-mail użytkownika.
   * **password\_hash** (VARCHAR(255)): Zaszyfrowane hasło.
   * **first\_name** (VARCHAR(100)): Imię.
   * **last\_name** (VARCHAR(100)): Nazwisko.
   * **phone\_number** (VARCHAR(20)): Numer telefonu.
   * **created\_at** (DATETIME): Data utworzenia konta.
   * **updated\_at** (DATETIME): Data ostatniej aktualizacji.
   * **is\_active** (BOOLEAN): Status konta.
2. **Tabela: Addresses (adresy)**
   * **address\_id** (INT, PK, AUTO\_INCREMENT): ID adresu.
   * **user\_id** (INT, FK - Users.user\_id): ID użytkownika.
   * **street** (VARCHAR(255)): Ulica.
   * **city** (VARCHAR(100)): Miasto.
   * **state** (VARCHAR(100)): Województwo.
   * **postal\_code** (VARCHAR(20)): Kod pocztowy.
   * **country** (VARCHAR(100)): Kraj.
3. **Tabela: Products (produkty)**
   * **product\_id** (INT, PK, AUTO\_INCREMENT): ID produktu.
   * **name** (VARCHAR(255)): Nazwa produktu.
   * **description** (TEXT): Opis produktu.
   * **brand** (VARCHAR(100)): Marka.
   * **category** (VARCHAR(100)): Kategoria (np. narty, kijki).
   * **price** (DECIMAL(10, 2)): Cena.
   * **stock\_quantity** (INT): Ilość w magazynie.
   * **created\_at** (DATETIME): Data dodania produktu.
   * **updated\_at** (DATETIME): Data ostatniej aktualizacji.
4. **Tabela: Product\_Images (zdjęcia produktów)**
   * **image\_id** (INT, PK, AUTO\_INCREMENT): ID zdjęcia.
   * **product\_id** (INT, FK - Products.product\_id): ID produktu.
   * **image\_url** (VARCHAR(255)): URL zdjęcia.
   * **alt\_text** (VARCHAR(255)): Opis alternatywny dla zdjęcia.
5. **Tabela: Orders (zamówienia)**
   * **order\_id** (INT, PK, AUTO\_INCREMENT): ID zamówienia.
   * **user\_id** (INT, FK - Users.user\_id): ID użytkownika.
   * **order\_date** (DATETIME): Data złożenia zamówienia.
   * **total\_amount** (DECIMAL(10, 2)): Łączna kwota zamówienia.
   * **status** (ENUM: 'pending', 'paid', 'shipped', 'delivered', 'canceled'): Status zamówienia.
   * **shipping\_address\_id** (INT, FK - Addresses.address\_id): ID adresu wysyłki.
6. **Tabela: Order\_Items (pozycje zamówienia)**
   * **order\_item\_id** (INT, PK, AUTO\_INCREMENT): ID pozycji zamówienia.
   * **order\_id** (INT, FK - Orders.order\_id): ID zamówienia.
   * **product\_id** (INT, FK - Products.product\_id): ID produktu.
   * **quantity** (INT): Ilość zamówionego produktu.
   * **unit\_price** (DECIMAL(10, 2)): Cena jednostkowa.
   * **total\_price** (DECIMAL(10, 2)): Cena całkowita za tę pozycję.
7. **Tabela: Payments (płatności)**
   * **payment\_id** (INT, PK, AUTO\_INCREMENT): ID płatności.
   * **order\_id** (INT, FK - Orders.order\_id): ID zamówienia.
   * **payment\_date** (DATETIME): Data płatności.
   * **amount** (DECIMAL(10, 2)): Kwota płatności.
   * **payment\_method** (ENUM: 'credit\_card', 'paypal', 'stripe'): Metoda płatności.
   * **status** (ENUM: 'completed', 'failed', 'pending'): Status płatności.
8. **Tabela: Shipping (wysyłka)**
   * **shipping\_id** (INT, PK, AUTO\_INCREMENT): ID wysyłki.
   * **order\_id** (INT, FK - Orders.order\_id): ID zamówienia.
   * **tracking\_number** (VARCHAR(50), UNIQUE): Numer śledzenia przesyłki.
   * **carrier** (ENUM: 'DHL', 'UPS', 'InPost'): Dostawca wysyłki.
   * **shipping\_date** (DATETIME): Data wysyłki.
   * **status** (ENUM: 'in\_transit', 'delivered', 'pending', 'returned'): Status wysyłki.
9. **Tabela: Reviews (recenzje)**
   * **review\_id** (INT, PK, AUTO\_INCREMENT): ID recenzji.
   * **product\_id** (INT, FK - Products.product\_id): ID produktu.
   * **user\_id** (INT, FK - Users.user\_id): ID użytkownika.
   * **rating** (INT): Ocena (np. 1-5).
   * **comment** (TEXT): Komentarz użytkownika.
   * **created\_at** (DATETIME): Data dodania recenzji.
10. **Tabela: Categories (kategorie)**
    * **category\_id** (INT, PK, AUTO\_INCREMENT): ID kategorii.
    * **name** (VARCHAR(100), UNIQUE): Nazwa kategorii (np. narty, kaski).
    * **description** (TEXT): Opis kategorii.
11. **Tabela: Inventory\_Logs (logi magazynu)**
    * **log\_id** (INT, PK, AUTO\_INCREMENT): ID logu.
    * **product\_id** (INT, FK - Products.product\_id): ID produktu.
    * **change\_quantity** (INT): Zmiana ilości (dodatnia lub ujemna).
    * **date** (DATETIME): Data zmiany.
    * **reason** (ENUM: 'purchase', 'return', 'adjustment'): Powód zmiany.

**Relacje**

* **Users** do **Addresses**: 1 do N (jeden użytkownik może mieć wiele adresów).
* **Products** do **Product\_Images**: 1 do N (każdy produkt może mieć wiele zdjęć).
* **Orders** do **Order\_Items**: 1 do N (zamówienie może zawierać wiele pozycji).
* **Orders** do **Payments** i **Shipping**: 1 do 1 (jedno zamówienie powiązane jest z jedną płatnością i jedną wysyłką).
* **Products** do **Reviews**: 1 do N (jeden produkt może mieć wiele recenzji).
* **Categories** do **Products**: 1 do N (jedna kategoria może zawierać wiele produktów).