Błażej Pawluk, Dawid Dębkowski

**ShopDB**

**Spis treści:**

1. Opis celu aplikacji
2. Charakterystyka rodzajów użytkowników oraz wymagania funkcjonalne względem każdego z nich
3. Opis bazy danych
   1. opis wyodrębnionych encji oraz związków między nimi
   2. normalizacja bazy danych
   3. uzasadnienie przechowywania redundantnych danych
   4. opisy tabel, klucze, powiązania między tabelami
   5. diagramy
   6. ograniczenia, triggery, procedury
4. Prawa dostępu użytkowników do poszczególnych komponentów bazy danych

**1. Opis celu aplikacji**

Aplikacja ShopDB jest symulacją internetowego sklepu odzieżowego. Pozwala   
na przeglądanie oraz kupowanie produktów dostępnych w sklepie. Ponadto jest   
ona również skierowana do pracowników sklepu – z poziomu aplikacji można dodawać nowe produkty, zarządzać stanem magazynu lub zamówieniami klientów.

**2. Charakterystyka rodzajów użytkowników oraz wymagania funkcjonalne względem każdego z nich.**

Przed wejściem do aplikacji należy się zalogować lub jeśli nie posiada się jeszcze konta   
to zarejestrować. Do każdego loginu przypisany jest jeden z typów użytkowników: klient, sprzedawca lub magazynier.

* **KLIENT** – klient sklepu. Rejestracja – stworzenie konta, z poziomu aplikacji tworzy konto tego typu. Klient ma możliwość przeglądania katalogu produktów (przeszukiwania bazy zgodnie z ustalonymi filtrami i opcjami sortowania), dodawania produktów o konkretnych rozmiarach do koszyka, a następnie   
  do złożenia zamówienia. Takie zamówienie może on anulować lub opłacić,   
  po czym zostanie ono do niego dostarczone (po skompletowaniu go przez magazyniera). Po otrzymaniu produktów klient może zgłosić zwrot. Ponadto może on dodawać do bazy swoje dane (bez których nie złoży zamówienia), przeglądać   
  i edytować je, a także przeglądać swoje wcześniejsze zakupy.
* **SPZEDAWCA** – pracownik sklepu. Po zalogowaniu się na swoje konto   
  ma on możliwość przeglądania produktów, edytowania ich oraz dodawania nowych. Ponadto może dodawać nowe typy produktów oraz kolory   
  do uwzględnienia w filtrach, zmieniać ceny produktów i ustawiać zniżki.
* **MAGAZYNIER** – pracownik magazynu. Po zalogowaniu się na swoje konto   
  ma on możliwość przeglądania produktów w magazynie, może je dodawać lub edytować ich ilość. Do jego obowiązków – funkcjonalności w aplikacji, należy również kompletowanie zamówień klientów oraz rozpatrywanie zwrotów.

**3. Opis bazy danych**

1. **Opis wyodrębnionych encji oraz związków między nimi:**

Użytkownicy – zawiera wszystkie dane o użytkownikach: loginy i hasła, typy użytkownika oraz, jeżeli są klientem, dane do kontaktu, dostawy oraz wybory użytkownika takie jak RODO, akceptację regulaminu lub cookies.

Produkty – zawiera wszystkie informacje o produktach dostępnych w sklepie: nazwy, kategorie, typy, kolory, cenę, zdjęcia, rozmiary oraz stany magazynu

Zamówienia – zawiera wszystkie informacje o zamówieniach: kto je złożył, co zawierają, obecny status, faktury, wartość oraz logi o zmianach statusu zamówienia

Encja zamówienia „łączy” użytkowników z produktami – użytkownicy mają zamówienia złożone z produktów.

1. **Normalizacja bazy danych**

Wersja początkowa:

* **U** (użytkownicy) = (login, hasło, typ\_użytkownika, email, numer telefonu, typ\_klienta, imię, nazwisko, nazwa\_firmy, NIP, adres (ulica, numer\_domu, numer\_mieszkania, kod\_pocztowy, miasto), RODO, regulamin, cookies),   
  **FU** = {login → hasło, typ\_użytkownika; email → typ\_klienta, imię, nazwisko, nazwa\_firmy, NIP, adres, RODO, regulamin, cookies;   
  numer\_telefonu → typ\_klienta, imię, nazwisko, nazwa\_firmy, NIP, adres, RODO, regulamin, cookies}
* **P** (produkty) = (nazwa, kategoria, typ, kolor, kod\_koloru, cena, zniżka, zdjęcia (lista), magazyn (rozmiar, stan\_magazynu, zarezerwowane)),   
  **FP** = {nazwa, kategoria, typ, kolor → cena, zniżka, zdjęcia, magazyn,   
  kolor → kod\_koloru, kod\_koloru → kolor}
* **Z** (zamówienia) = (id\_zamówienia, klient, pozycje (lista), status, faktura (id\_faktury, NIP, nazwa\_firmy, adres(ulica, numer\_domu, numer\_mieszkania, kod\_pocztowy, miasto)), wartość), **FZ** = {id\_zamówienia → klient, pozycje, status, faktura, wartość}
* **L** (logi zamówienia) = (id\_zamówienia, poprzedni\_status, nowy\_status, czas\_zmiany)

NF1 – każdy atrybut ma wartość atomową oraz nie istnieją powtarzające się krotki (istnieje klucz):

* **U** = (login, hasło, typ\_użytkownika, email, numer\_telefonu, typ\_klienta, imię, nazwisko, nazwa\_firmy, NIP, id\_adresu, RODO, regulamin, cookies),   
  **FU** = {login → hasło, typ\_użytkownika; email → typ\_klienta, imię, nazwisko, nazwa\_firmy, NIP, id\_adresu, RODO, regulamin, cookies;   
  numer\_telefonu → typ\_klienta, imię, nazwisko, nazwa\_firmy, NIP, id\_adresu, RODO, regulamin, cookies; imię, nazwisko},   
  **klucz**: (login, email)
* **A** (adresy) = (id\_adresu, ulica, numer\_domu, numer\_mieszkania, kod\_pocztowy, miasto),   
  **FA** = {id\_adresu → ulica, numer\_domu, numer\_mieszkania, kod\_pocztowy, miasto},   
  **klucz**: (id\_adresu)
* **P** = (nazwa, kategoria, typ, kolor, kod\_koloru, cena, zniżka),   
  **FP** = {nazwa, kategoria, typ, kolor → cena, zniżka, kolor → kod\_koloru,   
  kod\_koloru → kolor},   
  **klucz**: (nazwa, kategoria, typ, kolor)
* **ZD** (zdjęcia) = (produkt, zdjęcie),   
  **klucz**: (produkt, zdjęcie)
* **M** (magazyn) = (produkt, rozmiar, stan\_magazynu, zarezerwowane),   
  **FM** = {produkt, rozmiar → stan\_magazynu, zarezerwowane},   
  **klucz**: (produkt, rozmiar)
* **Z** = (id\_zamówienia, klient, status, id\_faktury, wartość),   
  **FZ** = {id\_zamówienia → klient, status, id\_faktury, wartość},   
  **klucz**: (id\_zamówienia)
* **PZ** (pozycje zamówienia) = (id\_zamówienia, produkt, rozmiar, ilość),   
  **FPZ** = {id\_zamówienia, produkt, rozmiar → ilość},   
  **klucz**: (id\_zamówienia, produkt, rozmiar)
* **F** (faktura) = (id\_faktury, NIP, nazwa\_firmy, id\_adresu),   
  **FF** = {id\_faktury → NIP, nazwa\_firmy, id\_adresu},   
  **klucz**: (id\_faktury)
* **L** = (id\_zamówienia, poprzedni\_status, nowy\_status, czas\_zmiany),   
  **FL** = {id\_zamówienia, poprzedni\_status → nowy\_status, czas\_zmiany},   
  **klucz**: (id\_zamówienia, poprzedni\_status)

NF2 – jest w NF1 oraz wszystkie atrybuty niekluczowe są w pełni funkcyjnie zależne od klucza

* **U** = (login, hasło, typ\_użytkownika, email, numer\_telefonu, typ\_klienta, imię, nazwisko, nazwa\_firmy, NIP, id\_adresu, RODO, regulamin, cookies),   
  **FU** = {login → hasło, typ\_użytkownika, email → numer\_telefonu, typ\_klienta, imię, nazwisko, nazwa\_firmy, NIP, id\_adresu, RODO, regulamin, cookies },   
  **klucz**: (login, email)
* **A** = (id\_adresu, ulica, numer\_domu, numer\_mieszkania, kod\_pocztowy, miasto), **FA** = {id\_adresu → ulica, numer\_domu, numer\_mieszkania, kod\_pocztowy, miasto},   
  **klucz**: (id\_adresu)
* **P** = (nazwa, kategoria, typ, kolor, cena, zniżka),   
  **FP** = {nazwa, kategoria, typ, kolor → cena, zniżka },   
  **klucz**: (nazwa, kategoria, typ, kolor)
* **K** (kolory) = (kolor, kod\_koloru),   
  **FK** = {kolor → kod\_koloru, kod\_koloru → kolor},  
  **klucz**: (kolor)
* **ZD** = (produkt, zdjęcie),   
  **klucz**: (produkt, zdjęcie)
* **M** = (produkt, rozmiar, stan\_magazynu, zarezerwowane),   
  **FM** = {produkt, rozmiar → stan\_magazynu, zarezerwowane},   
  **klucz**: (produkt, rozmiar)
* **Z** = (id\_zamówienia, klient, status, id\_faktury, wartość),   
  **FZ** = {id\_zamówienia → klient, status, id\_faktury, wartość},   
  **klucz**: (id\_zamówienia)
* **PZ** = (id\_zamówienia, produkt, rozmiar, ilość),   
  **FPZ** = {id\_zamówienia, produkt, rozmiar → ilość},   
  **klucz**: (id\_zamówienia, produkt, rozmiar)
* **F** = (id\_faktury, NIP, nazwa\_firmy, id\_adresu),   
  **FF** = {id\_faktury → NIP, nazwa\_firmy, id\_adresu},   
  **klucz**: (id\_faktury)
* **L** = (id\_zamówienia, poprzedni\_status, nowy\_status, czas\_zmiany),   
  **FL** = {id\_zamówienia, poprzedni\_status → nowy\_status, czas\_zmiany},   
  **klucz**: (id\_zamówienia, poprzedni\_status)

NF3 – jest w NF2 oraz wszystkie niekluczowe atrybuty zależą tylko od pełnego klucza minimalnego

* **U** = (login, hasło, typ\_użytkownika),   
  **FU** = {login → hasło, typ\_użytkownika},   
  **klucz:** (login)
* **C** (klienci) = (email, numer\_telefonu, typ\_klienta, imię, nazwisko, nazwa\_firmy, NIP, id\_adresu, RODO, regulamin, cookies),   
  **FC**  = {email → numer\_telefonu, typ\_klienta, imię, nazwisko, nazwa\_firmy, NIP, id\_adresu, RODO, regulamin, cookies, numer\_telefonu → email, typ\_klienta, imię, nazwisko, nazwa\_firmy, NIP, id\_adresu, RODO, regulamin, cookies},   
  **klucz**: (email, numer\_telefonu)
* **A** = (id\_adresu, ulica, numer\_domu, numer\_mieszkania, kod\_pocztowy, miasto), **FA** = {id\_adresu → ulica, numer\_domu, numer\_mieszkania, kod\_pocztowy, miasto},  
   **klucz**: (id\_adresu)
* **P** = (nazwa, kategoria, typ, kolor, cena, zniżka),   
  **FP** = {nazwa, kategoria, typ, kolor → cena, zniżka },   
  **klucz**: (nazwa, kategoria, typ, kolor)
* **K** (kolory) = (kolor, kod\_koloru),   
  **FK** = {kolor → kod\_koloru, kod\_koloru → kolor}  
  **klucz**: (kolor)
* **ZD** = (produkt, zdjęcie),
* **klucz**: (produkt, zdjęcie)
* **M** = (produkt, rozmiar, stan\_magazynu, zarezerwowane),   
  **FM** = {produkt, rozmiar → stan\_magazynu, zarezerwowane},   
  **klucz**: (produkt, rozmiar)
* **Z** = (id\_zamówienia, klient, status, id\_faktury, wartość),   
  **FZ** = {id\_zamówienia → klient, status, id\_faktury, wartość},   
  **klucz**: (id\_zamówienia)
* **PZ** = (id\_zamówienia, produkt, rozmiar, ilość),   
  **FPZ** = {id\_zamówienia, produkt, rozmiar → ilość},   
  **klucz**: (id\_zamówienia, produkt, rozmiar)
* **F** = (id\_faktury, NIP, nazwa\_firmy, id\_adresu),   
  **FF** = {id\_faktury → NIP, nazwa\_firmy, id\_adresu},   
  **klucz**: (id\_faktury)
* **L** = (id\_zamówienia, poprzedni\_status, nowy\_status, czas\_zmiany),   
  **FL** = {id\_zamówienia, poprzedni\_status → nowy\_status, czas\_zmiany},   
  **klucz**: (id\_zamówienia, poprzedni\_status)

1. **Uzasadnienie przechowywania redundantnych danych**

Pomimo, że każda relacja zawiera klucz, do każdej dodany został atrybut id, przykładowo **U** = (id\_użytkownika, login, hasło, typ\_użytkownika). Zostało to wprowadzone, aby przeszukiwanie danych było efektywniejsze (łatwiej sprawdzać pola typu int niż   
np. varchar(255)) oraz do łatwiejszego łączenia tabel kluczami obcymi. Ponadto w relacji **C** został dodany atrybut id\_użytkownika, aby połączyć tabele.

Pomimo, że przy przeszukiwaniu faktur można bez problemu otrzymać dane klienta składającego zamówienie (id\_faktury w **Z**, klient w **Z**, który łączy się z **C**, które zawiera dane), to są one dodatkowo przechowywane w relacji **F**. Zostało to zaprojektowane w taki sposób, aby przy zmianie danych klientów nadal znać dane, na które w przeszłości została wystawiona faktura. Ponadto w tabeli z zamówieniami poza id\_faktury dodane jest również pole *faktura* typu boolean przechowujące informację czy klient chce fakturę. Jest to spowodowane tym, że faktura wystawiana jest dopiero przy dostarczaniu zamówienia, a chęć otrzymania faktury wyrazić należy już przy składaniu zamówienia.

Dodana również została jedna tabela słownikowa, która przechowuje typy produktów, aby zapobiec tworzeniu wielu typów, które oznaczają ten sam przedmiot oraz aby ułatwić funkcjonalność wyświetlania wszystkich typów produktów przykładowo na liście rozwijalnej przy filtrowaniu katalogu produktów.

1. **Opisy tabel, klucze, powiązania między tabelami**

**Użytkownicy** (  
id użytkownika: int – klucz główny,  
login: varchar(255) – klucz kandydacki,  
hasło: varchar(255),  
typ klienta: enum(„klient”, „sprzedawca”, „magazynier”))   
przechowuje dane do logowania do aplikacji

**Klienci** (  
id klienta: int – klucz główny,  
id użytkownika: int – klucz obcy (Użytkownicy),  
typ klienta: enum(„indywidualny”, „firma”),   
imię: varchar(255),  
nazwisko: varchar(255),  
nazwa firmy: varchar(255),  
email: varchar(255) – klucz kandydancki,  
numer telefonu: varchar(15) – klucz kandydacki,  
id adresu: int – klucz obcy (Adresy),  
NIP: char(10),  
RODO: boolean,  
regulamin: boolean,  
cookies: boolean)  
przechowuje dane klientów

**Adresy** (  
id adresu: int – klucz główny,  
ulica: varchar(255),  
numer domu: int,  
numer mieszkania: int,   
miasto: varchar(255),  
kod pocztowy: char(6))  
przechowuje wszystkie adresy

klucz kandydacki

**Produkty** (  
id produktu: int – klucz główny,  
nazwa: varchar(255),   
kategoria: enum(„mężczyzna”, „kobieta”, „chłopak”, „dziewczyna”),  
id typu: int – klucz obcy (Typy produktów),  
id koloru: int – klucz obcy (Kolory produktów),  
cena: decimal(4, 2),  
przecena: int)  
katalog wszystkich produktów

klucz kandydacki

**Typy produktów** (  
id typu: int – klucz główny,  
typ: varchar(255) – klucz kandydacki)  
tabela słownikowa uzasadniona w 3. c)

**Kolory produktów** (  
id koloru: int – klucz główny,  
kolor: varchar(255) – klucz kandydacki,  
kod koloru: char(7)- klucz kandydacki)  
tabela z listą kolorów wraz z ich kodem RGB do łatwego wyświetlania listy kolorów w aplikacji

**Zdjęcia** (  
id zdjęcia: int – klucz główny,  
id produktu: int – klucz obcy (Produkty),  
zdjęcie: blob)  
baza wszystkich zdjęć wraz z dopasowaniem ich do produktów

klucz kandydacki

**Magazyn** (  
id magazynowe: int – klucz główny,  
id produktu: int – klucz obcy (Produkty),  
rozmiar: enum(„XS”, „S”, „M”, „L”, „XL”),  
ilość: int,  
zarezerwowane: int)  
przechowuje informacje o dostępności danych produktów w danym rozmiarze w magazynie

klucz kandydacki

**Zamówienia** (  
id zamówienia: int – klucz główny,  
id klienta: int – klucz obcy (Klienci),  
faktura: boolean,  
id faktury: int – klucz obcy (Faktury),  
status: enum(„koszyk”, „złożone”, „opłacone”, „anulowane”, „zrealizowane”, „zgłoszony zwrot”, „zwrócone”),  
wartość: decimal(10, 2))  
przechowuje informacje o wszystkich zamówieniach i ich statusach, w przypadku statusu „koszyk” jest ono nie złożone i służy klientowi do wygodnego tworzenia zamówienia

**Pozycje zamówienia** (  
id pozycji: int - klucz główny,  
id zamówienia: int – klucz obcy (Zamówienia),  
id magazynowe: int – klucz obcy (Magazyn),  
ilość: int)  
przechowuje informacje o konkretnych pozycjach zamówienia, odwołuje się bezpośrednio do id magazynowego – do zamówienia dodawany jest konkretny rozmiar produktu

klucz kandydacki

**Faktury** (  
id faktury: int – klucz główny,  
NIP: char(10),  
nazwa firmy: varchar(255),  
id adresu: int)  
przechowuje dane konkretnych faktur

**Logi zamówienia** (  
id logu: int – klucz główny,  
id zamównienia: int – klucz obcy (Zamówienia),  
poprzedni status: enum(„koszyk”, „złożone”, „opłacone”, „anulowane”, „zrealizowane”, „zgłoszony zwrot”, „zwrócone”),  
nowy status: enum(„koszyk”, „złożone”, „opłacone”, „anulowane”, „zrealizowane”, „zgłoszony zwrot”, „zwrócone”),  
data zmiany: timestamp)  
przechowuje wszystkie zmiany statusu zamówień

klucz kandydacki

Powiązania między tabelami dokładniej pokazane na diagramie 1 w podpunkcie e)

1. **Diagramy**

Obraz zawierający tekst, zrzut ekranu

Opis wygenerowany automatycznie  
Diagram 1: diagram związków encji bazy danych

Obraz zawierający tekst, zrzut ekranu, diagram, Czcionka

Opis wygenerowany automatycznie  
Diagram 2: diagram stanów zamówienia

1. **Ograniczenia, triggery, procedury**

* **Ograniczenia:**
  + Wszystkie atrybuty lub grupy atrybutów będące kluczami kandydackimi muszą być unikatowe. Takie sprawdzenia są dokonywane poprzez dodanie constrainta *unique(atrybut)* lub *unique key(grupa atrybutów)*
  + Atrybuty *email, numer telefonu, NIP* z relacji *Klienci* oraz *kod pocztowy*z relacji *Adresy* są sprawdzane pod względem formatu, przykładowo *kod pocztowy* musi być w formacie *„xx-xxx”*
  + Na atrybuty *zarezerwowane* z *Magazyn* oraz *wartość* z *Zamówienia* ustawiona jest domyślna wartość na 0
  + Dodatkowo do zabezpieczenia integralności bazy danych dodane   
    są również sprawdzenia: *Magazyn.ilość ≥ 0, Magazyn.ilość ≥ Magazyn.zarezerwowane, Pozycje zamówienia.ilość ≥ 0*
* **Triggery:**
  + 3 triggery (*after insert, after update, after delete*) aktualizujące wartość zamówieniapo każdej dodanej/edytowanej/usuniętej pozycji zamówienia (tylko gdy zamówienie ma status *„koszyk”*)
  + Automatyczne usuwanie pozycji zamówienia, jeśli ustawiona ilość produktów w danej pozycji wynosi 0
  + Automatyczne tworzenie logów zamówienia
* **Procedury:** wszystkie akcje na bazie danych wykonywane przez użytkowników aplikacji są wykonywane przez procedury, które wszystkie operacje wykonują   
  w sposób transakcyjny. W ten sposób baza jest zabezpieczona przed nieoczekiwanymi danymi, zapytaniami oraz chroniona jest jej integralność.
  + *auth\_user* – logowanie użytkowników
  + *add\_client* – zarejestrowanie nowego klienta (dodaje do *Użytkownicy, Klienci* oraz tworzy nowe zamówienie dla klienta o statusie „*koszyk*”)
  + *change\_acc\_info\_individual, change\_acc\_info\_company, change\_address* – edytowanie danych klienta, w przypadku adresu, jeżeli podany adres jest już w bazie danych to jest on „podłączany” do klienta, inaczej dodawany jest nowy – nigdy nie usuwamy adresów dodanych raz do bazy
  + *add\_order\_pos, edit\_order\_pos, remove\_order\_pos* – dodawanie/zmiana ilości/usuwanie produktów w koszyku, sprawdzane jest czy żądana ilość jest dostępna w magazynie
  + *place\_order, pay\_order, cancel\_order,* *complete\_order, report\_return, consider\_return* – zmiana statusu zamówienia, przy każdej zmianie sprawdzane są odpowiednio warunki oraz aktualizowane wartości zgodnie z diagramem 2 z punktu e)
  + *add\_type, edit\_type, remove\_type, add\_color, edit\_color, remove\_color* – dodawanie/edytowanie/usuwanie typu/koloru produktów, w bazie pod id=1 są domyślne wartości („inny”), które są ustawiane dla wszystkich produktów, których kolor został usunięty, krotek tych nie można usunąć
  + *add\_product, edit\_product* – dodawanie/edytowanie produktów   
    do katalogu, nie można usuwać produktów z poziomu bazy, aby mieć dostęp do produktów z dawnych zamówień, które nie są kontynuowane
  + *add\_photo, remove\_photo* – dodawanie/usuwanie zdjęć dla produktów
  + *change\_price, change\_discount* – zmiana ceny/zniżki, automatycznie aktualizuje wartości koszyków, które posiadają produkt o zmienionej cenie
  + *add\_warehouse, edit\_warehouse* – dodawanie/zmiana ilości produktów   
    w magazynie, nie można usuwać z tego samego powodu, co przy produktach
  + *show\_client\_info, show\_client\_orders, show\_order\_details, show\_order\_logs, show\_invoice, show\_products, show\_product\_details, show\_paid\_orders, show\_reported\_returns* – procedury służące   
    do wyświetlania odpowiednich danych odpowiednim użytkownikom, korzystają z prepared statementów oraz widoków, aby ograniczyć użytkownikom dostęp do danych do minimum niezbędnego do wygodnego korzystania z aplikacji

**4. Prawa dostępu użytkowników do poszczególnych komponentów bazy danych**

Użytkownicy aplikacji mają minimalny dostęp do bazy danych. Wszystkie operacje wstawiania, edytowania, usuwania lub wyświetlania są przez nich wykonywane poprzez procedury. Mają zatem oni tylko uprawnienia do wykonywania procedur zgodnych z ich typem użytkownika (przy typie użytkownika podane są nazwy procedur z punktu 3. f:

* **KLIENT**: *auth\_user, add\_client, change\_acc\_info\_individual, change\_acc\_info\_company, change\_address, add\_order\_pos, edit\_order\_pos, remove\_order\_pos, place\_order, pay\_order, cancel\_order, report\_return, show\_client\_info, show\_client\_orders, show\_order\_details, show\_order\_logs, show\_invoice, show\_products, show\_product\_details*
* **SPRZEDAWCA**: *auth\_user, add\_type, edit\_type, remove\_type, add\_color, edit\_color, remove\_color, add\_product, edit\_product, add\_photo, remove\_photo, change\_price, change\_discount, show\_products, show\_product\_details*
* **MAGAZYNIER**: *auth\_user , add\_warehouse, edit\_warehouse, cancel\_order, complete\_order, consider\_return, show\_order\_details, show\_order\_logs, show\_invoice, show\_products, show\_product\_details, show\_paid\_orders, show\_reported\_returns*