**Katolicki Uniwersytet Lubelski Jana Pawła II**

**Wydział Nauk Przyrodniczych i Technicznych**

**Instytut** **Matematyki, Informatyki i Architektury Krajobrazu**

Informatyka, studia stacjonarne I stopnia

**Robert Kosieradzki**

Nr albumu 154860

**Wykorzystanie języka PHP w wersji 8.1.8 oraz systemu zarządzania bazą danych MySQL do stworzenia sklepu internetowego**

Praca licencjacka

napisana na seminarium **Bazy danych**  
pod kierunkiem

**dr Marcina Płonkowskiego**

**Lublin 2023**

**Spis treści**

[Wstęp 2](#_Toc127486294)

[1 Rozdział I. Bazy danych 3](#_Toc127486295)

[1.1 Teoria bazy danych 3](#_Toc127486296)

[1.1.1 podstawowe pojęcia //dopisaćKilka// 3](#_Toc127486297)

[1.2 Modele bazy danych 4](#_Toc127486298)

[1.2.1 model relacyjny 4](#_Toc127486299)

[1.2.2 model hipertekstowy 4](#_Toc127486300)

[1.2.3 model obiektów 5](#_Toc127486301)

[1.2.4 model hierarchiczny 5](#_Toc127486302)

[1.2.5 model sieciowy 5](#_Toc127486303)

[1.3 Normalizacja bazy danych 6](#_Toc127486304)

[1.3.1 Proces normalizacji danych i stopnie normalizacji 6](#_Toc127486305)

[1.4 Porównanie systemów bazodanowych 9](#_Toc127486306)

[1.4.1 Relacyjne systemy bazodanowe 10](#_Toc127486307)

[1.4.2 Nierelacyjne systemy bazodanowe 10](#_Toc127486308)

[1.4.3 Podstawowe komendy 11](#_Toc127486309)

[1.4.4 Dlaczego MySQL //dopisać jeszcze// 11](#_Toc127486310)

[2 Rozdział II. Tworzenie aplikacji internetowych 12](#_Toc127486311)

[2.1 Inne języki i ich opis 12](#_Toc127486312)

[2.2 Zabezpieczenia stosowane w backendzie 12](#_Toc127486313)

[2.3 Przygotowywanie aplikacji w PHP 12](#_Toc127486314)

[3 Rozdział III. Opis działania aplikacji sklepu internetowego 12](#_Toc127486315)

[3.1 Wygląd aplikacji i jej funkcjonalności 12](#_Toc127486316)

[3.2 Wygląd bazy danych sklepu 12](#_Toc127486317)

[3.2.1 Jak działa koszyk 12](#_Toc127486318)

[4 Bibliografia 12](#_Toc127486319)

# Wstęp

dotyczy projektu sklepu internetowego, który zostanie stworzony przy użyciu języka PHP w wersji 8.1.8 oraz systemu zarządzania bazą danych MySQL. Celem pracy jest zaprezentowanie możliwości i korzyści płynących z wykorzystania tych technologii w tworzeniu sklepów internetowych oraz przedstawienie konkretnego przykładu ich implementacji.

# Rozdział I. Bazy danych

## Teoria bazy danych

To dziedzina informatyki zajmująca się modelowaniem, projektowaniem, tworzeniem, implementacją i zarządzaniem bazami danych. Bazą danych jest zbiór danych, który jest przechowywany i przetwarzany w sposób umożliwiający efektywne wyszukiwanie, aktualizację i utrzymanie danych.

W teorii baz danych istnieje kilka modeli danych, takich jak model relacyjny, hipertekstowy, obiektowy, hierarchiczny i sieciowy. Model relacyjny jest najczęściej stosowanym modelem danych w bazach danych, ponieważ umożliwia łatwą implementację i zarządzanie danymi za pomocą języka SQL.

Ważnym elementem teorii baz danych jest również normalizacja baz danych, która polega na dzieleniu danych na tabele i tworzeniu odpowiednich relacji między nimi, aby zapewnić integralność i trwałość danych.

Podstawowe komponenty bazy danych to:

- Schemat bazy danych, który określa strukturę bazy danych,

- Instancja bazy danych, która jest konkretnym egzemplarzem bazy danych,

- Dane bazy danych, które są przechowywane w bazie danych,

- Programy aplikacji, które pozwalają na dostęp do danych z bazy danych.

### podstawowe pojęcia //dopisaćKilka//

Klucz główny ---

Klucz obcy ---

Encja jest tym, co musimy określić, aby reprezentować obiekt lub zestaw obiektów. Terminu byt używamy do opisania zarówno obiektów fizycznych, jak i niematerialnych. Przykładem encji może być obiekt taki jak ten: OSOBA, a cechami mogą być wzrost, rozmiar buta, waga. Te cechy są nazywane atrybutami encji.

Krotkę można zdefiniować w następujący sposób: Jeżeli tabela spełnia wymagania relacji (jest relacją), a jej kolumny są atrybutami, to krotka jest wierszem (rekordem). Jedna krotka przechowuje stałe wartości różnych typów danych i nie może być modyfikowana w innej krotce. Dlatego też typ kolejnych krotek tabeli, np. tytuł, ISBN, będzie stały, a ich zawartość będzie różna. Odczytanie krotki wymaga podania jej stałej indeksu.

Atrybuty są definiowane jako kolumny relacyjne z identyfikatorami (nazwami). W modelu relacyjnej bazy danych, gdy tabelę dwuwymiarową nazywamy relacją (gdy spełnia ona warunki relacji), to nazwane kolumny tej tabeli nazywane są atrybutami.[[1]](#footnote-1)

## Modele bazy danych

### model relacyjny

Model relacyjnej bazy danych to model oparty na relacjach między tabelami, gdzie każda tabela reprezentuje jednostkę (np. użytkownika, produkt, zamówienie) i ma unikalny klucz podstawowy. Relacje między tabelami oparte są na wspólnych kolumnach (kluczach obcych), które pozwalają łączyć dane z różnych tabel i tworzyć złożone zapytania. Model jest bardzo elastyczny i może łatwo zarządzać i przetwarzać duże zbiory danych.

Cztery kluczowe właściwości definiują transakcje w relacyjnych bazach danych: atomowość, spójność, izolacja i trwałość — często określane jako ACID (niepodzielność, spójność, izolacja i trwałość).

* Atomowość definiuje wszystkie elementy, które składają się na kompletną transakcję bazy danych.
* Spójność określa zasady utrzymywania elementów danych we właściwym stanie po transakcji.
* Izolacja oznacza, że ​​efekt transakcji jest niewidoczny dla innych, dopóki nie zostanie zatwierdzona, aby uniknąć nieporozumień.
* Trwałość oznacza, że zmiany w danych stają się trwałe po zatwierdzeniu transakcji.[[2]](#footnote-2)

### model hipertekstowy

Model hipertekstowy bazy danych opiera się na połączeniach pomiędzy dokumentami, zamiast relacji pomiędzy tabelami jak w przypadku modelu relacyjnego. W tym modelu, informacje są przechowywane w postaci hiperłączy, które łączą dokumenty ze sobą w sposób niehierarchiczny. Dzięki temu, użytkownicy mogą łatwo przeglądać i nawigować po danych, bez potrzeby korzystania z tradycyjnych zapytań SQL. Model hipertekstowy jest często używany w aplikacjach takich jak strony internetowe czy aplikacje do zarządzania wiedzą, gdzie ważne jest łatwe przeglądanie i dostępność informacji.

### model obiektów

Model obiektów bazy danych jest modelem, który opiera się na koncepcji obiektów jako podstawowych jednostek przechowywania danych. Każdy obiekt jest reprezentowany jako instancja klasy i posiada atrybuty i metody. W tym modelu, dane i ich relacje są opisywane w sposób bardziej naturalny i zbliżony do sposobu myślenia ludzkiego, co ułatwia ich zrozumienie i zarządzanie. Modele obiektowe często są stosowane w aplikacjach związanych z obiektami rzeczywistymi, takich jak systemy zarządzania nieruchomościami czy opieką zdrowotną. W przeciwieństwie do modelu relacyjnego, model obiektów jest bardziej elastyczny i pozwala na łatwe dostosowywanie do zmieniających się potrzeb biznesowych.

### model hierarchiczny

Model hierarchiczny bazy danych to model, w którym dane są przechowywane w postaci drzewa, gdzie każdy węzeł jest reprezentowany przez rekord, a każde połączenie pomiędzy węzłami odpowiada relacji pomiędzy danymi. W tym modelu dane są organizowane w sposób hierarchiczny, z jednym głównym węzłem i pod węzłami, co pozwala na łatwe zarządzanie danymi i uzyskiwanie informacji o nich. Model ten jest często stosowany w aplikacjach, w których dane są organizowane w sposób drzewiasty, takich jak systemy katalogów plików czy bazy danych katalogów produktów. W przeciwieństwie do modelu relacyjnego, model hierarchiczny jest mniej elastyczny i może być trudniejszy do modyfikacji w przypadku zmian potrzeb biznesowych.

### model sieciowy

Model sieciowy bazy danych to model, w którym dane są przechowywane jako obiekty i ich relacje. W tym modelu każdy obiekt jest reprezentowany przez węzeł, a każde połączenie pomiędzy obiektami odpowiada relacji. W przeciwieństwie do modelu hierarchicznego, w modelu sieciowym relacje między obiektami nie są ograniczone do jednej linii, ale mogą być bardziej złożone i skomplikowane. Model sieciowy jest często stosowany w aplikacjach, w których dane są powiązane w sposób sieciowy, takich jak systemy zarządzania kontaktami czy systemy zarządzania zasobami ludzkimi. W przeciwieństwie do modelu relacyjnego, model sieciowy jest bardziej elastyczny i pozwala na łatwiejsze reprezentowanie złożonych relacji między danymi.

## Normalizacja bazy danych

Normalizacja danych to proces, który polega na organizowaniu danych w bazie danych w taki sposób, aby zapobiec powstawaniu błędów i umożliwić łatwiejsze aktualizowanie i wyszukiwanie danych. Celem normalizacji jest usunięcie powtórzeń danych i zapewnienie jednoznaczności danych w bazie.

Normalizacja danych jest ważna, ponieważ bazy danych często są wykorzystywane do przechowywania i udostępniania dużych ilości danych. Powtórzenie danych i brak jednoznaczności danych może prowadzić do błędów i trudności z aktualizowaniem i wyszukiwaniem danych. Normalizacja danych pozwala uniknąć tych problemów i zapewnia, że baza danych będzie działać prawidłowo i bezproblemowo.[[3]](#footnote-3)

### Proces normalizacji danych i stopnie normalizacji

Normalizacja danych jest procesem, który polega na dzieleniu danych na mniejsze, bardziej znormalizowane tabele, takie jak tabele z informacjami o produktach, tabele z informacjami o klientach i tabele z informacjami o zamówieniach. Normalizacja danych jest kluczowa dla zapewnienia prawidłowego funkcjonowania bazy danych i jest często stosowana w projektowaniu baz danych. Normalizacja bazy danych polega na sprowadzeniu struktury tabelarycznej w bazie danych do postaci spełniającej założenia formy kanonicznej.

Normalizacja danych może wpłynąć na wydajność bazy danych. Im bardziej znormalizowane są dane, tym trudniej jest je wyszukiwać, co może spowodować spadek wydajności. Z drugiej strony, jeśli dane są dobrze znormalizowane, to można szybciej i dokładniej wyszukiwać po jednej kolumnie bez potrzeby przeszukiwania całej bazy danych. Im bardziej znormalizowane są dane, tym łatwiej jest je aktualizować, a także dodawać i usuwać dane, bez konieczności modyfikacji całej struktury bazy danych. Jednocześnie, nadmiernie znormalizowane dane mogą uniemożliwić lub utrudnić wprowadzanie zmian, na przykład w przypadku, gdy istnieje potrzeba dodania nowej kolumny do tabeli. Normalizacja danych może wpłynąć na bezpieczeństwo bazy danych. Ponadto, normalizacja danych może pomóc w uniknięciu sytuacji, w której dane są powielane w różnych miejscach bazy danych, co z kolei może zmniejszyć ryzyko utraty danych w przypadku awarii bazy danych. W każdym przypadku, odpowiednie dobranie stopnia normalizacji danych jest kluczowe dla uzyskania optymalnych wyników w zakresie wydajności, elastyczności i bezpieczeństwa bazy danych.

#### Pierwsza postać normalna (1NF)

Reguły:

* Jedna tabela opisuje jeden obiekt.
* Wartości atrybutów są atomowe(nie podzielne).
* Tabela nie zawiera powtarzających się informacji.
* Znaczenie kolumn nie zależy od ich kolejności.

Tabelę, która nie spełnia któregoś z powyższych warunków, nazywana jest zdenormalizowaną.[[4]](#footnote-4)

|  |  |  |  |
| --- | --- | --- | --- |
| Dish\_id | Dish\_name | price | ingredient |
| 1 | carbonara | 30 | [”sera pecorino”,”makaron”,…] |
| 2 | Spaghetti bolognese | 25 | [”mięso mielone”,”makaron”,…] |

Tabela Przykład zdenormalizowanej tabeli

W przykładowej Tabela 1 Przykład zdenormalizowanej tabeli składniki nie są atomowe. Po pierwsze, każda nazwa dania ma listę składników. Co więcej, niektóre składniki powtarzają się w kilku pozycjach. Jeśli chcesz zmodyfikować składnik, na przykład kurczak zamiast mięso mielone, musisz zmodyfikować wszystkie pozycje w tabeli. Wyszukiwanie składników użytych w tabeli o takiej strukturze również jest mocno utrudnione.

W tym przypadku rozwiązaniem jest użycie relacji wiele-do-wielu i podzielenie przykładowej tabeli na tabelę zawierającą elementy, tabelę zawierającą komponenty oraz tabelę pośrednią zawierającą informacje o tym, jakie komponenty zawiera dany element. Dzięki temu komponent staje się samodzielnym podmiotem w bazie danych, który posiada klucz podstawowy, dzięki czemu jest identyfikowalny i niepowtarzalny. Przykładową tabela uproszczona do pierwszej postaci normalnej wygląda następująco:

|  |  |  |
| --- | --- | --- |
| Dish\_id | Dish\_name | price |
| 1 | carbonara | 30 |
| 2 | Spaghetti bolognese | 25 |

Tabela Przykład tabeli znormalizowanej 1NF pozycji

|  |  |
| --- | --- |
| Ingredient\_id | ingredient |
| 1 | sera pecorino |
| 2 | makaron |
| 3 | mięso mielone |

Tabela Przykład tabeli znormalizowanej 1NF składników

|  |  |
| --- | --- |
| Position\_id | ingredient\_id |
| 1 | 1 |
| 1 | 2 |
| 2 | 2 |
| 2 | 3 |

Tabela Przykład tabeli znormalizowanej 1NF pozycji i składników [[5]](#footnote-5)

#### Druga postać normatywna (2NF)

Reguły:

* być w pierwszej postaci normalnej;
* nie posiadać kolumny, której wartości można wywnioskować z niektórych wartości klucza kandydującego (mogą być wnioskowane z całego klucza).

Polega na utworzeniu dodatkowej relacji, w której będą przechowywane częściowe dane z relacji modyfikującej.

Przedstawiona tabela zawiera jednak zbyt wiele nieistotnych danych.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Order\_id | First\_name | Last\_name | address | product | price | quantity |
| 1 | Jan | Kowalski | Piękna 4 | bułka | 0,33 | 2 |
| 10 | Tomasz | Nowak | Lipowa 20 | chleb | 5 | 1 |

Tabela Przykład zdenormalizowanej tabeli 2NF

Dane klienta zostały przeniesione do nowej tabeli jak i produkty z ceną.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Customer\_id | First\_name | Last\_name | address | City |
| 1 | Jan | Kowalski | Piękna 4 | Warszawa |
| 5 | Tomasz | Nowak | Lipowa 20 | Lublin |

Tabela Przykład tabeli znormalizowanej klienci 2NF

|  |  |  |
| --- | --- | --- |
| Product\_id | product | price |
| 1 | bułka | 0,33 |
| 3 | chleb | 5 |
| 5 | drożdżówka | 2 |

Tabela Przykład tabeli znormalizowanej produktów 2NF

|  |  |  |  |
| --- | --- | --- | --- |
| Order\_id | Customer\_id | Product\_id | Quantity |
| 1 | 1 | 1 | 2 |
| 10 | 5 | 3 | 1 |

Tabela Przykład tabeli znormalizowanej zamówień 2NF

Wszystkie dane niezwiązane ze zamówieniami są przenoszone do innych tabel, w zleceniach przechowywane są tylko klucze obce.

#### Trzecia postać normatywna (3NF)

Reguły:

* jest w drugiej postaci normalnej.
* każdy atrybut nienależący do klucza potencjalnego nie zależy funkcyjnie od innych atrybutów nienależących do klucza potencjalnego.

Dodatkowa zasada jest bardzo zbliżona do tej wprowadzonej w drugiej postaci normalnej. Czasem jest nazywana zakazem przechodniości. Nie pozwala ona, aby jakiś atrybut dał się wywnioskować z innego atrybutu. Nazwa wynika z tego, że jeśli mamy klucz potencjalny K oraz atrybuty A i B, to według tej reguły nie może istnieć zależność K → A → B. Czyli dowolny atrybut niekluczowy może zależeć tylko od całości klucza potencjalnego, nigdy zaś od innego atrybutu czy fragmentu.

#### Czwarta postać normatywna (4NF)

Reguły:

* Jest w trzeciej postaci normatywnej.

#### Piąta postać normatywna (5NF)

Reguły:

* Jest w czwartej postaci normatywnej.

## Porównanie systemów bazodanowych

Wyróżniamy głównie dwa systemy bazodanowe relacyjne i nierelacyjne. W relacyjnych systemach bazodanowych dane są przechowywane w tabelach z wierszami i kolumnami, podczas gdy w nierelacyjnych systemach bazodanowych dane mogą być przechowywane w różnych strukturach, takich jak dokumenty, grafy lub kolumny. W relacyjnych systemach bazodanowych najczęściej stosowanym językiem zapytań jest SQL, a w nierelacyjnych systemach bazodanowych stosowane są różne języki, takie jak MongoDB Query Language czy Apache Cassandra Query Language. Relacyjne systemy takie jak MySQL, PostgreSQL, Oracle Database oraz nierelacyjne takie jak MongoDB, Apache Cassandra.

### Relacyjne systemy bazodanowe

Relacyjne systemy bazodanowe (RDBMS) to rodzaj systemów bazodanowych, które przechowują dane w tabelach z wierszami i kolumnami. Te tabele są połączone ze sobą za pomocą kluczy obcych, co umożliwia łatwe łączenie danych z różnych tabel. Relacyjne systemy bazodanowe zapewniają wiele zalet, w tym:

* skalować zarówno poziomo (poprzez dodawanie nowych serwerów) jak i pionowo (poprzez zwiększanie zasobów serwera).
* bezpieczne i umożliwiają łatwe zarządzanie uprawnieniami dostępu do danych.
* relacyjne systemy bazodanowe stosują restrykcyjne reguły spójności danych, co zapobiega wprowadzeniu niepoprawnych lub sprzecznych danych do bazy.
* możliwość zastosowania transakcji które umożliwiają grupowanie kilku operacji na bazie danych w pojedynczą jednostkę działania, która jest albo wykonana w całości, albo w ogóle nie jest wykonana.

Wadą relacyjnych systemów bazodanowych jest zwykle mniejsza wydajność w porównaniu z nierelacyjnymi systemami bazodanowymi, szczególnie w przypadku dużej liczby zapytań i transakcji.

### Nierelacyjne systemy bazodanowe

Nierelacyjne systemy bazodanowe (NoSQL) to rodzaj systemów bazodanowych, które nie używają tabel, kolumn i kluczy obcych do przechowywania danych, ale stosują inne, bardziej elastyczne struktury, takie jak dokumenty, kolumny czy grafy. Zalety systemów nieralcyjnych bazodanowych to:

* skalowalność dzięki temu, że dane mogą być łatwo rozproszone na wiele serwerów, co umożliwia obsługę dużych ilości danych.
* elastyczne zmienianie schematu bazy danych, co umożliwia dostosowywanie struktury bazy do zmieniających się potrzeb aplikacji.
* bardziej wydajne w przypadku dużych ilości danych i zapytań.
* otwarte źródła większość nierelacyjnych systemów bazodanowych jest dostępna na licencji open source, co umożliwia ich modyfikację i dostosowanie do potrzeb użytkownika.

Wady nierelacyjnych systemów bazodanowych to zwykle mniejsza spójność danych w porównaniu z relacyjnymi systemami bazodanowymi, brak standardowych języków zapytań, co może wprowadzać trudności w programowaniu aplikacji oraz większa trudność w zapewnieniu bezpieczeństwa danych w skomplikowanych środowiskach z wieloma serwerami i węzłami.

### Podstawowe komendy

SELECT: służy do wyświetlania danych z bazy danych.

Przykład: SELECT \* FROM tabela;

INSERT: służy do dodawania nowych rekordów do bazy danych.

Przykład: INSERT INTO tabela (kolumna1, kolumna2) VALUES ('wartość1', 'wartość2');

UPDATE: służy do aktualizowania istniejących rekordów w bazie danych.

Przykład: UPDATE tabela SET kolumna1='nowa wartość' WHERE warunek;

DELETE: służy do usuwania rekordów z bazy danych.

Przykład: DELETE FROM tabela WHERE warunek;

CREATE: służy do tworzenia nowych tabel, baz danych i indeksów.

Przykład: CREATE TABLE tabela (kolumna1 typ1, kolumna2 typ2);

ALTER: służy do modyfikowania istniejących tabel, kolumn i indeksów.

Przykład: ALTER TABLE tabela ADD COLUMN kolumna typ;

DROP: służy do usuwania tabel, baz danych i indeksów.

Przykład: DROP TABLE tabela;

JOIN: służy do łączenia danych z różnych tabel na podstawie wspólnych kolumn.

Przykład: SELECT \* FROM tabela1 INNER JOIN tabela2 ON tabela1.kolumna=tabela2.kolumna;

GROUP BY: służy do grupowania danych na podstawie wartości w określonej kolumnie.

Przykład: SELECT kolumna, COUNT(\*) FROM tabela GROUP BY kolumna;

ORDER BY: służy do sortowania wyników zapytania w kolejności rosnącej lub malejącej.

Przykład: SELECT \* FROM tabela ORDER BY kolumna DESC;

### Dlaczego MySQL //dopisać jeszcze//

MySQL to popularny relacyjny system zarządzania bazami danych, który jest dostępny na licencji open source. MySQL jest powszechnie stosowany w aplikacjach internetowych, takich jak fora internetowe, sklepy internetowe czy systemy CMS. MySQL obsługuje wiele standardowych języków zapytań, takich jak SQL, a także wiele innych języków programowania, dzięki czemu można łatwo integrować go z różnymi aplikacjami. MySQL posiada także wiele narzędzi administracyjnych i do zarządzania bazą danych, takich jak MySQL Workbench czy MySQL Command Line Client. MySQL jest powszechnie stosowany w branży IT i jest obsługiwany przez wiele dostawców chmur obliczeniowych, takich jak Amazon Web Services, Google Cloud Platform czy Microsoft Azure.

# Rozdział II. Tworzenie aplikacji internetowych

Tworzenie aplikacji internetowych z użyciem PHP i MySQL może być stosunkowo proste, gdyż te technologie są dobrze udokumentowane i mają szerokie zastosowanie w tworzeniu aplikacji internetowych.

## Inne języki i ich opis

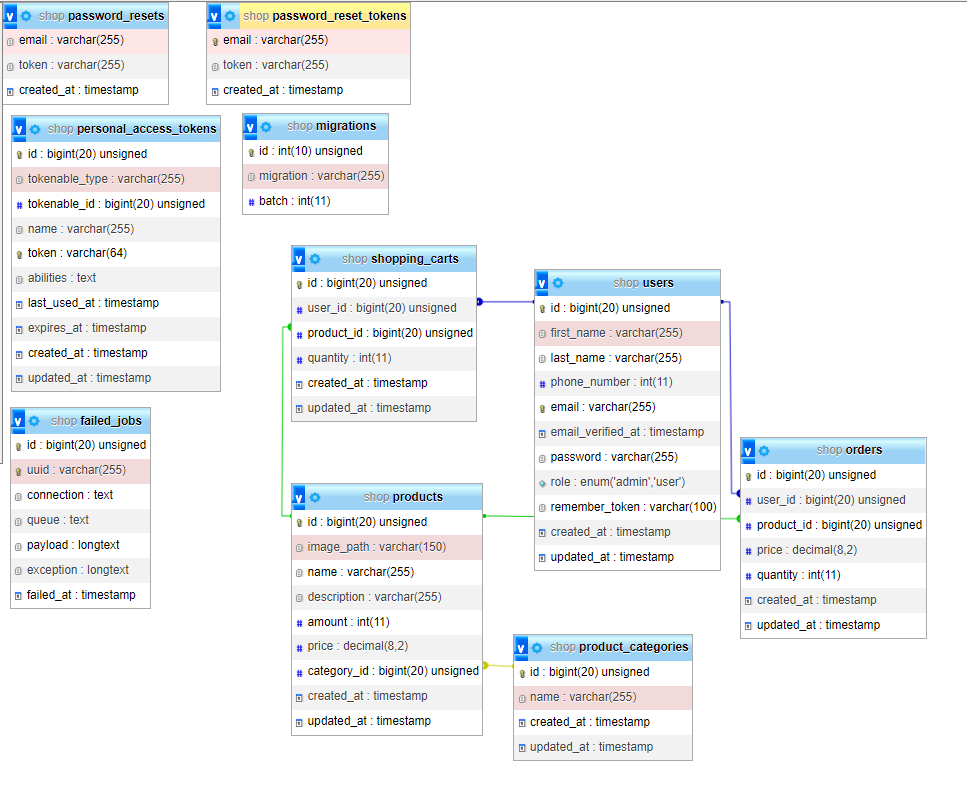
## Zabezpieczenia stosowane w backendzie

## Przygotowywanie aplikacji w PHP

# Rozdział III. Opis działania aplikacji sklepu internetowego

## Wygląd aplikacji i jej funkcjonalności

## Wygląd bazy danych sklepu



### Jak działa koszyk

# Bibliografia

[Tabela 1 Przykład zdenormalizowanej tabeli 6](#_Toc127483189)

[Tabela 2 Przykład tabeli znormalizowanej 1NF pozycji 7](#_Toc127483190)

[Tabela 3 Przykład tabeli znormalizowanej 1NF składników 7](#_Toc127483191)

[Tabela 4 Przykład tabeli znormalizowanej 1NF pozycji i skladnikow 7](#_Toc127483192)

[Tabela 5 Przykład zdenormalizowanej tabeli 2NF 8](#_Toc127483193)

[Tabela 6 Przykład tabeli znormalizowanej klienci 2NF 8](#_Toc127483194)

[Tabela 7 Przykład tabeli znormalizowanej produktów 2NF 8](#_Toc127483195)

[Tabela 8 Przykład tabeli znormalizowanej zamówień 2NF 8](#_Toc127483196)

1. https://www.lomilowka.pl/upload/file/SBD/1\_BazydanychPodstawy.pdf [↑](#footnote-ref-1)
2. https://www.oracle.com/pl/database/what-is-a-relational-database/ [↑](#footnote-ref-2)
3. https://encyklopedia.pwn.pl/haslo/normalizacja;3948277.html [↑](#footnote-ref-3)
4. https://zpe.gov.pl/a/przeczytaj/D7nxbjkYr [↑](#footnote-ref-4)
5. https://devszczepaniak.pl/postaci-normalne/ [↑](#footnote-ref-5)