Sklep Muzyczny

Mikołaj Laskowski

1. Wstęp 4
1.1. Cel projektu 4
1.2. Zakres projektu 4
1.3 Użytkownicy systemu 4
2. Opis ogólny 6
2.1. Przeznaczenie witryny 6
2.2. Główne funkcjonalności 6
2.3. Wymagania techniczne 6
2.4. Struktura katalogów i plików projektu 8
..... 8
3. Struktura strony internetowej 9
3.1. Strona główna (home.php) 9
3.2. Katalog produktów (katalog.php) 9
3.3. Strona produktu (produkt.php) 9
3.4. Koszyk (cart.php) 9
3.5. Proces zamówienia (checkout.php) 9
3.6. Panel użytkownika (profile.php) 9
3.7. Panel administracyjny (panel.php) 10
4. Elementy statyczne 11
4.1. HTML – struktura dokumentów 11
4.2. CSS – stylizacja 12
4.3. Biblioteki zewnętrzne 12
5. JavaScript 13
6. Elementy dynamiczne (PHP) 16
7. Baza danych 18
7.1. Opis tabel i pól 18
Opis tabel bazy danych sklepu muzycznego 18
7.4. Relacje między tabelami 22
7.3. Diagram ERD 24
7.5. Przykładowe zapytanie 25

8. Bezpieczeństwo 26
8.1. Walidacja danych 26
8.2. Obsługa sesji 28
8.3. Autoryzacja 28
9. Możliwości rozbudowy 29
9.1. Funkcjonalności 29
9.2. Techniczne 29
9.3. Biznesowe 29
10. Załączniki 30
10.1. Lista błędów/uwag i pomysłów na ulepszenia 30

1. Wstęp

1.1. Cel projektu

Stworzenie kompleksowego systemu sklepu muzycznego online, umożliwiającego sprzedaż i wypożyczanie instrumentów muzycznych, z pełną obsługą zamówień, płatności i zarządzania magazynem.

1.2. Zakres projektu

- System sprzedaży instrumentów muzycznych
- System wypożyczania instrumentów
- Panel administracyjny do zarządzania sklepem
- System ocen i recenzji produktów
- Zarządzanie zamówieniami i dostawami
- System kont użytkowników i pracowników

1.3 Użytkownicy systemu

1. **Klienci**
    a. Przeglądanie katalogu produktów
    b. Składanie zamówień
    c. Dodawanie ocen i recenzji
    d. Zarządzanie koszykiem klienta (tj. Dodawanie, usuwanie, zmienianie ilości produktów znajdujących się w koszyku)
    e. Możliwość stosowania kodów promocyjnych
    f. Zarządzanie swoim kontem użytkownika
    g. Dostęp do swojej historii zamówień oraz opinii
2. **Pracownicy**
    a. Obsługa zamówień - możliwość sprawdzenia szczegółów zamówienia, a także oznaczenia go (zamówienia) jako wysłane
    b. Obsługa dostaw - możliwość sprawdzenia szczegółów dostawy, a także oznaczenia jej (dostawy) jako dostarczonej (odebranej)
    c. Możliwość podejrzenia produktów dostępnych w systemie, a także ich stanu magazynowego
3. **Managerowie**
    a. Obsługa zamówień - możliwość sprawdzenia szczegółów zamówienia, a także oznaczenia go (zamówienia) jako wysłane
    b. Obsługa dostaw - możliwość sprawdzenia szczegółów dostawy, a także oznaczenia jej (dostawy) jako dostarczonej (odebranej); dodatkowo mogą tworzyć nowe dostawy (służą do zamawiania produktów do magazynu), a także edycji ich statusu na oczekiwaną, dostarczoną lub anulowaną

c. Możliwość podejrzenia produktów dostępnych w systemie, a także ich stanu magazynowego
d. Możliwość podejrzenia zatrudnionych (wprowadzonych do systemu) pracowników, a także powiązanych z nimi informacji

4. **Sekretarka**
    a. Obsługa wiadomości kontaktowych
5. **Informatyk**
    a. Pełny dostęp do widoków produktów, kategorii produktów, producentów oraz pracowników (z wyłączeniem możliwości zarządzania właścicielem)
    b. Dostęp do widoku kodów promocyjnych, a także do możliwości ich dodawania, edycji, wyłączania/włączania oraz ich usuwania
    c. Dostęp do widoku klientów, a także do możliwości ich edycji oraz podglądania
6. **Właściciel**
    a. Pełne zarządzanie systemem
    b. Dodatkowe tabelki widoczne i możliwe do edycji tylko przez właściciela np. Finanse oraz Stanowiska

1.3. Informacje do logowania różnych użytkowników serwisu

Rola Pracownika	Identyfikator Pracownika	Hasło Pracownika
Pracownik	0159	Hasłosiema123#
Manager	1594	Hasłosiema123#
Właściciel	7494	Hasłosiema123#
Informatyk	2690	Hasłosiema123#
Sekretarka	0969	Hasłosiema123#

Email Klienta	Hasło Klienta
herbert@example.com	Hasłosiema123#
g.braun@gmail.com	Hasłosiema123#

2. Opis ogólny

2.1. Przeznaczenie witryny

Platforma e-commerce dedykowana sprzedaży i wypożyczaniu instrumentów muzycznych, oferująca kompleksową obsługę klienta od wyboru produktu po realizację zamówienia.

2.2. Główne funkcjonalności

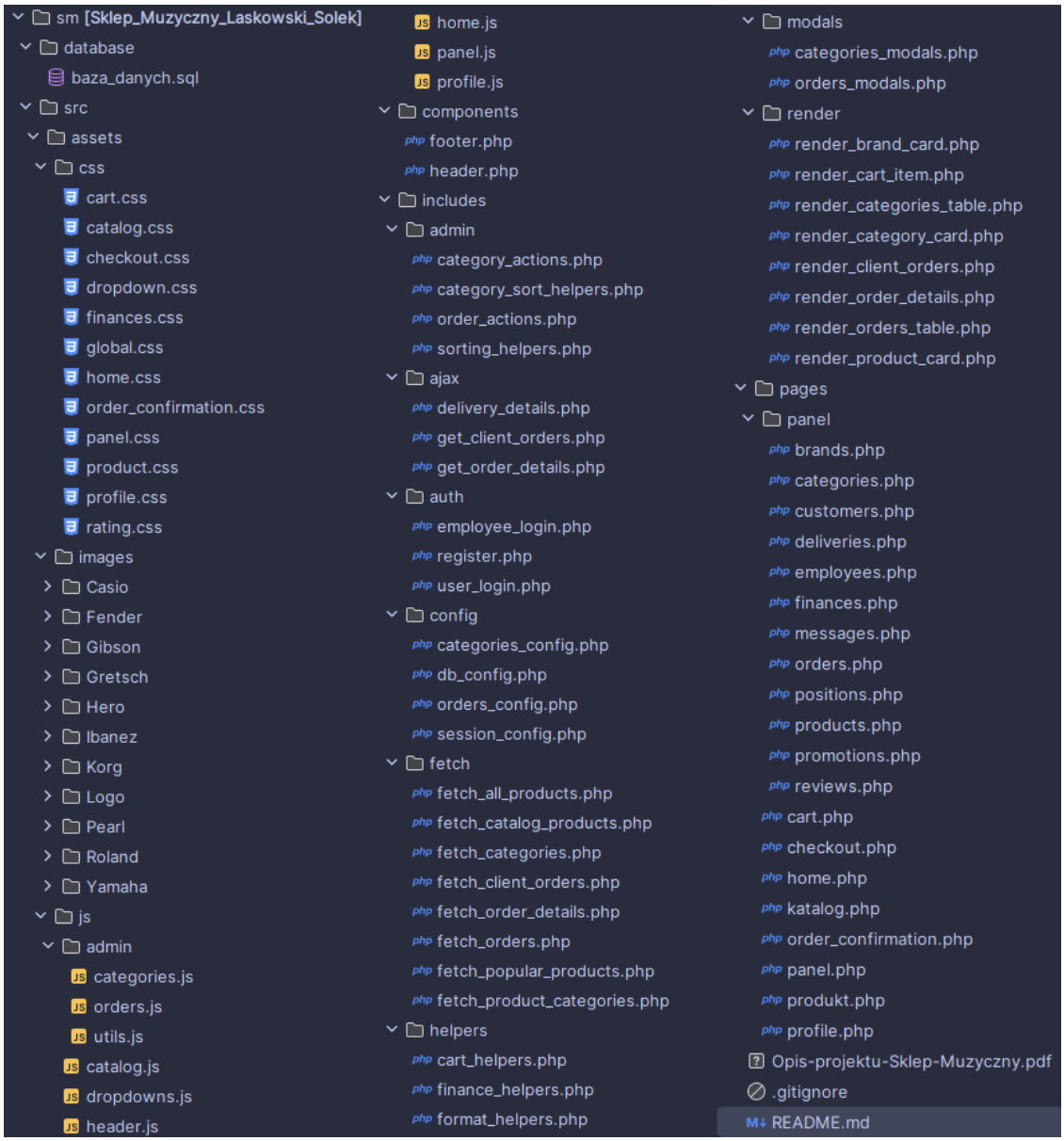
1. **System katalogowy**
    a. Zaawansowane filtrowanie produktów
    b. Kategoryzacja instrumentów
    c. System wyszukiwania
    d. Szczegółowe karty produktów
2. **System zamówień**
    a. Koszyk zakupowy
    b. Proces checkout
    c. Śledzenie statusu zamówienia
    d. Historia zamówień
3. **Panel administracyjny**
    a. Zarządzanie produktami
    b. Zarządzanie użytkownikami
    c. Zarządzanie kodami promocyjnymi
    d. Zarządzanie dostawami
    e. Zarządzanie pracownikami
    f. Zarządzanie stanowiskami
    g. Raporty finansowe
4. **System ocen i recenzji**
    a. Oceny produktów
    b. Komentarze użytkowników
    c. Ranking produktów

2.3. Wymagania techniczne

1. **Serwer**
    a. XAMPP 5.2.1^
    b. Apache 2.4^
    c. PHP 8.2.12^
    d. MariaDB 10.4.32^

2. **Klient**
    a. Jakakolwiek przeglądarka internetowa
    b. Javascript włączony
    c. Urządzenie o minimalnej rozdzielczości 375x667 (z uwagi na to, że nie ma mnejszych wyświetlaczy jest to minimalny wspierany rozmiar; teoretycznie mniejsze rozdzielczości są obsługiwane, ale do pewnego momentu)

2.4. Struktura katalogów i plików projektu



3. Struktura strony internetowej

3.1. Strona główna (home.php)

- Slider z kategoriami instrumentów
- Slider z producentami instrumentów
- Hero section
- Popularne produkty
- Formularz kontaktowy

3.2. Katalog produktów (katalog.php)

- Filtry zaawansowane
- Sortowanie produktów
- Lista produktów
- Paginacja
- Widok siatki/listy
- Wyszukiwarka produktów

3.3. Strona produktu (produkt.php)

- Galeria zdjęć
- Specyfikacja techniczna
- Cena produktu
- System ocen i recenzji
- Kategoria oraz producent instrumentu
- Dostępność produktu

3.4. Koszyk (cart.php)

- Lista produktów
- Podsumowanie zamówienia
- Kody rabatowe
- Aktualizacja ilości

3.5. Proces zamówienia (checkout.php)

- Dane dostawy
- Formularz adresu wysyłki
- Podsumowanie zamówienia
- Potwierdzenie zakupu

3.6. Panel użytkownika (profile.php)

- Dane osobowe oraz możliwość ich edycji

- Historia zamówień
- Historia wystawionych opinii
- Jeśli użytkownik nie jest zalogowany bądź nie posiada konta wyświetla się formularz logowania i rejestracji

3.7. Panel administracyjny (panel.php)

- Panel administratora serwisu z dostępnymi widokami:
  a. Produkty
  b. Kategorie produktów
  c. Producenci
  d. Oceny produktów
  e. Zamówienia
  f. Kody promocyjne
  g. Pracownicy
  h. Klienci
  i. Wiadomości
  j. Dostawy
  k. Finanse
  l. Stanowiska

4. Elementy statyczne

4.1. HTML – struktura dokumentów

Najciekawszym, a zarazem najciekawszym elementem HTML-a jest includowanie komponentów takich jak header czy footer.

```php
<?php
  include_once '../includes/helpers/cart_helpers.php';

  $current_page = basename($_SERVER['PHP_SELF']);
  $home_page = 'home.php';
  $cart_page = 'cart.php';
  $profile_page = 'profile.php';
  $admin_page = 'panel.php';

  $active_class = function ($page) use ($current_page) {
    return $current_page === $page ? 'active_subpage' : '';
  };
?>
<header class="header">
  <div class="logo">
    <img alt="Logo Sklepu Muzycznego" src="../assets/images/Logo/logo_muzyczny.png">
  </div>
  <form class="search-bar" role="search" method="get" action="katalog.php">
    <input type="text" name="search" aria-label="Wyszukiwarka instrumentów" class="se
    <button aria-label="Wyszukaj" class="search-button" type="submit">
      <i aria-hidden="true" class="fa-solid fa-magnifying-glass"></i>
    </button>
  </form>
  <nav class="tray">
    <button aria-label="Strona główna" class="tray-item <?= $active_class($home_page)
            title="Przejdź do strony głównej" type="button">
      <i aria-hidden="true" class="fa-solid fa-home"></i>
      <span>Główna</span>
    </button>
    <button aria-label="Koszyk" class="tray-item <?= $active_class($cart_page) ?>" ti
            type="button">
      <i aria-hidden="true" class="fa-solid fa-cart-shopping"></i>
      <span>Koszyk (<?= getTotalItemsInCart() ?>)</span>
    </button>
    <button aria-label="Profil użytkownika" class="tray-item <?= $active_class($profi
            title="Przejdź do swojego profilu" type="button">
      <i aria-hidden="true" class="fa-solid fa-user"></i>
      <span>Profil</span>
    </button>
    <?php
      if (isset($_SESSION['employee_id'])) {
```

Mając plik header.php możemy go zaimportować do pliku home.php! Niesamowita technologia, prawie jak w React-cie!

```php
<?php include '../components/header.php'; ?>
<main class="fade-in">
```

Gdy używasz `include`, uważaj, żeby nie załączać ponownie plików, które zostały już

dołączone wcześniej. Ścieżki do plików podawaj względem pliku, który wykonuje `include`.

4.2. CSS – stylizacja

- global.css - Style globalne
- home.css - Strona główna
- catalog.css - Katalog produktów
- product.css - Strona produktu
- cart.css - Koszyk
- checkout.css - Proces zamówienia
- panel.css - Panel administracyjny
- profile.css - Profil użytkownika
- finances.css - Dodatkowe style do widoku finansów
- rating.css - Dodatkowe style dla ocen produktów
- order_confirmations.css - Dodatkowe style dla potwierdzenia zamówienia
- dropdown.css - Dodatkowe style dla pól rozwijanych

Najtrudniejszym zagadnieniem związanym z CSS-em jest oczywiście centrowanie div-a, dlatego też załączam kod, który w tym pomoże.

```css
.tray-item {
  font-size: var(--font-xl);
  display: flex;
  align-items: center;
  justify-content: center;
  width: var(--card-size-xs-sm);
  height: var(--card-size-xs-sm);
  color: var(--text-color);
  border-radius: var(--radius-full);
  background-color: var(--secondary-bg);
}
```

Dochodzi tutaj do niesamowitego wyczynu, mianowicie justify-content centruje dzieci klasy .tray-item w osi X, a align-items centruje je w osi Y.

4.3. Biblioteki zewnętrzne

- Font Awesome – Ikonki

5. JavaScript

```javascript
/**
 * Inicjalizuje obsługę przewijania dla danej listy
 * @param {HTMLElement} list - Lista do przewijania
 * @param {HTMLElement} leftButton - Przycisk przewijania w lewo
 * @param {HTMLElement} rightButton - Przycisk przewijania w prawo
 */
Show usages  herb-chan
function initializeScrolling(list :HTMLElement , leftButton :HTMLElement , rightButton :HTMLElement )
  if (!list || !leftButton || !rightButton) return;

  leftButton.addEventListener( type: 'click',  listener: () :void  => {
    list.scrollBy( options: {
      left: -scrollAmount,
      behavior: 'smooth',
    });
  });

  rightButton.addEventListener( type: 'click',  listener: () :void  => {
    list.scrollBy( options: {
      left: scrollAmount,
      behavior: 'smooth',
    });
  });

  Show usages  herb-chan
  function updateScrollButtons() :void  {
    const isAtStart :boolean  = list.scrollLeft === 0;
    const isAtEnd :boolean  = list.scrollLeft + list.clientWidth >= list.scrollWidth;

    leftButton.style.opacity = isAtStart ? '0.5' : '1';
    rightButton.style.opacity = isAtEnd ? '0.5' : '1';
  }

  list.addEventListener( type: 'scroll', updateScrollButtons);
  window.addEventListener( type: 'resize', updateScrollButtons);
  updateScrollButtons();
```

Funkcja initializeScrolling odpowiada za przewijanie sekcji kategorii oraz producentów na stronie głównej.

```
document.addEventListener( type: 'DOMContentLoaded',  listener: () : void  => {
    // Obsługa formularza filtrów
    const filtersForm : HTMLElement  = document.getElementById( elementId: 'filters-form');
    const priceInputs : NodeListOf<Element>  = filtersForm.querySelectorAll( selectors: 'input[type="number"]');

    // Automatyczne wysyłanie formularza przy zmianie sortowania
    const sortSelect : Element  = filtersForm.querySelector( selectors: 'select[name="sort_by"]');
    sortSelect.addEventListener( type: 'change',  listener: () : void  => {
        filtersForm.submit();
    });

    // Walidacja zakresu cen
    priceInputs.forEach( callbackfn: input : Element  => {
        input.addEventListener( type: 'input',  listener: () : void  => {
            const minPrice : number  = parseFloat(priceInputs[0].value) || 0;
            const maxPrice : number  = parseFloat(priceInputs[1].value) || Infinity;

            if (maxPrice < minPrice) {
                priceInputs[1].setCustomValidity( error: 'Maksymalna cena musi być większa od minimalnej');
            } else {
                priceInputs[1].setCustomValidity( error: '');
            }
        });
    });

    // Obsługa resetowania filtrów
    const resetButton : Element  = document.querySelector( selectors: '.reset-filters-btn');
    resetButton.addEventListener( type: 'click',  listener: (e : Event ) : void  => {
        e.preventDefault();
        window.location.href = 'katalog.php';
    });

    // Animacja produktów przy załadowaniu
    const productsGrid : Element  = document.querySelector( selectors: '.products-grid');
    productsGrid.style.opacity = '0';

    setTimeout( handler: () : void  => {
        productsGrid.style.opacity = '1';
    },  timeout: 100);
});
```

Skrypt ten służy do obsługi filtrów w katalogu produktów.

```
/**
 * Filtruje tabelę na podstawie wartości z pola wyszukiwania
 *
 * @param {string} tableId - ID tabeli do filtrowania
 * @param {string} inputId - ID pola wyszukiwania
 * @param {number} columnIndex - Indeks kolumny, według której filtrujemy
 */
Show usages  herb-chan
function filterTable(tableId, inputId, columnIndex) : void  {
    const input : HTMLElement  = document.getElementById(inputId);
    const filter : string  = input.value.toUpperCase();
    const table : HTMLElement  = document.getElementById(tableId);
    const rows : HTMLCollectionOf<HTMLTableRowElement>  = table.getElementsByTagName( qualifiedName: 'tr');

    for (let i : number  = 1; i < rows.length; i++) { // Rozpoczynamy od 1, aby pominąć nagłówek
        const cell : HTMLTableCellElement  = rows[i].getElementsByTagName( qualifiedName: 'td')[columnIndex];
        if (cell) {
            const textValue : string  = cell.textContent || cell.innerText;
            if (textValue.toUpperCase().indexOf(filter) > -1) {
                rows[i].style.display = '';
            } else {
                rows[i].style.display = 'none';
            }
        }
    }
}
```

filterTable jest utility funkcją, która służy do filtrowania tabelek w widokach panelu administratora.

6. Elementy dynamiczne (PHP)

```php
function renderCategoryCard(array $category) : string
{
  /** @var mysqli $connection */
  global $connection;

  // Pobierz najpopularniejszy produkt z danej kategorii
  $sql = "
    SELECT
      iz.url,
      iz.alt_text
    FROM instrumenty i
    JOIN instrument_zdjecia iz ON i.id = iz.instrument_id AND iz.kolejnosc = 1
    LEFT JOIN zamowienie_szczegoly zs ON i.id = zs.instrument_id
    WHERE i.kategoria_id = {$category['id']}
    GROUP BY i.id, iz.url, iz.alt_text
    ORDER BY COUNT(zs.id) DESC
    LIMIT 1
  ";

  $result = mysqli_query($connection, $sql);
  $imageHtml = '<div aria-hidden="true" class="instrument-icon"></div>';

  if ($result && $image = mysqli_fetch_assoc($result)) {
    $imageUrl = "../assets/images/" . $image['url'];
    $imageHtml = "<div aria-hidden=\"true\" class=\"instrument-icon\">
      <img src=\"{$imageUrl}\" alt=\"{$image['alt_text']}\" />
    </div>";
  }

  return "
  <a href=\"katalog.php?category_id={$category['id']}\" class=\"instrument-card fade-in\" role=\"button\
    {$imageHtml}
    <span class=\"instrument-name\">{$category['nazwa']}</span>
  </a>
  ";
}
```

Funkcja renderCategoryCard służy do tworzenia karty kategorii dla przekazanej kategorii, używana na stronie głównej. Zwraca string z tagami html, który może zostać wyświetlony na stronie.

```php
function renderCartItem(mysqli $connection, array $product, string $type = 'buy') :
{
  $productId = $product['id'];
  $name = htmlspecialchars($product['nazwa']);
  $category = htmlspecialchars($product['nazwa_kategorii']);
  $imageUrl = "../assets/images/" . htmlspecialchars($product['url']);
  $altText = htmlspecialchars($product['alt_text']);
  $quantity = intval($product['quantity']);
  $price = formatPrice($product['cena_sprzedazy'], $quantity);

  // Pobierz stan magazynowy produktu
  $stock = 0;
  $query = "SELECT stan_magazynowy FROM instrumenty WHERE id = ?";
  $stmt = mysqli_prepare($connection, $query);
  mysqli_stmt_bind_param($stmt, types: 'i', &...vars: $productId);
  mysqli_stmt_execute($stmt);
  $result = mysqli_stmt_get_result($stmt);

  if ($result && mysqli_num_rows($result) > 0) {
    $productData = mysqli_fetch_assoc($result);
    $stock = intval($productData['stan_magazynowy']);
  }
  mysqli_free_result($result);

  // Przygotuj atrybuty dla inputa quantity
  $maxQuantity = max( value: 1, $stock); // Nigdy nie mniej niż 1
  $maxAttr = $quantity >= $stock ? 'max="'.$quantity.'"' : 'max="'.$stock.'"';
  $plusButtonDisabled = $quantity >= $stock ? 'disabled' : '';

  return "
  <li class=\"cart-item\">
    <img alt=\"{$altText}\" src=\"{$imageUrl}\">
    <div class=\"cart-item-details\">
      <div class=\"cart-item-product-details\">
        <div class=\"cart-item-text\">
          <div class=\"cart-item-name\">{$name}</div>
          <div class=\"cart-item-category\">{$category}</div>
          <div class=\"cart-item-category stock-info\">Dostępnych: {$stock}</div>
```

Funkcja renderCartItem służy do utworzenia karty produktu dodanego do koszyka, który został przekazany do funkcji, używany w podstronie koszyka. Zwraca string z tagami html, który może zostać wyświetlony na stronie.

7. Baza danych

7.1. Opis tabel i pól

Opis tabel bazy danych sklepu muzycznego

a. wiadomosci

- **Opis:** Wiadomości od klientów.
- **Pola:**
  - id
  - email
  - temat
  - tresc
  - data_wyslania
  - status

b. zamowienie_szczegoly

- **Opis:** Szczegóły zamówień.
- **Pola:**
  - id
  - zamowienie_id
  - instrument_id
  - ilosc
  - cena

c. instrumenty

- **Opis:** Przechowuje informacje o instrumentach muzycznych.
- **Pola:**
  - id
  - kod_produktu
  - nazwa
  - opis
  - cena_sprzedazy
  - cena_kupna
  - cena_wypozyczenia_dzien

    - stan_magazynowy
    - producent_id
    - kategoria_id

d. kategorie_instrumentow

- **Opis:** Kategorie instrumentów muzycznych.
- **Pola:**
    - id
    - nazwa

e. klienci

- **Opis:** Informacje o klientach sklepu.
- **Pola:**
    - id
    - uzytkownik_id

f. zamowienia

- **Opis:** Zamówienia złożone przez klientów.
- **Pola:**
    - id
    - klient_id
    - data_zamowienia
    - status
    - kod_promocyjny_id
    - adres_wysylki

g. wypozyczenia

- **Opis:** Wypożyczenia instrumentów.
- **Pola:**
    - id
    - klient_id
    - instrument_id
    - data_wypozyczenia
    - data_zwrotu
    - cena_wypozyczenia
    - status

h. pracownicy

- **Opis:** Informacje o pracownikach.
- **Pola:**
    - id
    - uzytkownik_id
    - stanowisko_id
    - data_zatrudnienia
    - identyfikator
    - data_zwolnienia

i. uzytkownicy

- **Opis:** Konta użytkowników systemu.
- **Pola:**
    - id
    - nazwa_uzytkownika
    - email
    - haslo
    - data_rejestracji
    - typ

j. instrument_oceny

- **Opis:** Oceny instrumentów.
- **Pola:**
    - id
    - instrument_id
    - user_id
    - ocena
    - komentarz
    - czy_edytowana
    - data_oceny
    - data_edycji

k. instrument_zdjecia

- **Opis:** Zdjęcia instrumentów.
- **Pola:**
    - id
    - instrument_id
    - url
    - alt_text
    - kolejnosc

l. dostawa_szczegoly

- **Opis:** Szczegóły dostaw instrumentów.
- **Pola:**
    - id
    - dostawa_id
    - instrument_id
    - ilosc
    - cena_zakupu
    - status

m. dostawy

- **Opis:** Dostawy od producentów.
- **Pola:**
    - id
    - data_zamowienia
    - data_dostawy
    - status
    - producent_id
    - pracownik_id

n. kody_promocyjne

- **Opis:** Kody promocyjne.
- **Pola:**
    - id
    - kod
    - znizka
    - data_rozpoczecia
    - data_zakonczenia
    - aktywna

o. koszyk

- **Opis:** Koszyki zakupowe.
- **Pola:**
    - id
    - klient_id

p. koszyk_szczegoly

- **Opis:** Pozycje w koszykach.

- **Pola:**
  - id
  - koszyk_id
  - instrument_id
  - ilosc
  - cena
  - typ
  - okres_wypozyczenia

q. stanowiska

- **Opis:** Stanowiska pracy.
- **Pola:**
  - id
  - nazwa
  - wynagrodzenie_miesieczne

7.4. Relacje między tabelami

Główne relacje

1. **instrumenty ↔ kategorie_instrumentow**
   a. `instrumenty.kategoria_id → kategorie_instrumentow.id`
   b. Każdy instrument należy do jednej kategorii
2. **instrumenty ↔ producenci**
   a. `instrumenty.producent_id → producenci.id`
   b. Każdy instrument jest produkowany przez jednego producenta
3. **uzytkownicy ↔ klienci**
   a. `klienci.uzytkownik_id → uzytkownicy.id`
   b. Każdy klient jest powiązany z jednym użytkownikiem
4. **uzytkownicy ↔ pracownicy**
   a. `pracownicy.uzytkownik_id → uzytkownicy.id`
   b. Każdy pracownik jest powiązany z jednym użytkownikiem
5. **pracownicy ↔ stanowiska**
   a. `pracownicy.stanowisko_id → stanowiska.id`
   b. Każdy pracownik ma przypisane jedno stanowisko

Relacje związane z zamówieniami

6. **klienci ↔ zamowienia**
   a. `zamowienia.klient_id → klienci.id`
   b. Każde zamówienie jest składane przez jednego klienta

7. **zamowienia ↔ zamowienie_szczegoly**
    a. `zamowienie_szczegoly.zamowienie_id → zamowienia.id`
    b. Każde zamówienie może mieć wiele pozycji szczegółowych
8. **instrumenty ↔ zamowienie_szczegoly**
    a. `zamowienie_szczegoly.instrument_id → instrumenty.id`
    b. Każda pozycja w zamówieniu dotyczy jednego instrumentu
9. **kody_promocyjne ↔ zamowienia**
    a. `zamowienia.kod_promocyjny_id → kody_promocyjne.id`
    b. Zamówienie może (ale nie musi) korzystać z jednego kodu promocyjnego

Relacje związane z koszykiem

10. **klienci ↔ koszyk**
    a. `koszyk.klient_id → klienci.id`
    b. Każdy klient ma jeden koszyk
11. **koszyk ↔ koszyk_szczegoly**
    a. `koszyk_szczegoly.koszyk_id → koszyk.id`
    b. Koszyk może zawierać wiele pozycji
12. **instrumenty ↔ koszyk_szczegoly**
    a. `koszyk_szczegoly.instrument_id → instrumenty.id`
    b. Każda pozycja w koszyku dotyczy jednego instrumentu

Relacje związane z dostawami

13. **dostawy ↔ dostawa_szczegoly**
    a. `dostawa_szczegoly.dostawa_id → dostawy.id`
    b. Każda dostawa może zawierać wiele pozycji
14. **instrumenty ↔ dostawa_szczegoly**
    a. `dostawa_szczegoly.instrument_id → instrumenty.id`
    b. Każda pozycja w dostawie dotyczy jednego instrumentu
15. **producenci ↔ dostawy**
    a. `dostawy.producent_id → producenci.id`
    b. Dostawa pochodzi od jednego producenta
16. **pracownicy ↔ dostawy**
    a. `dostawy.pracownik_id → pracownicy.id`
    b. Dostawa jest obsługiwana przez jednego pracownika

Relacje związane z wypożyczeniami

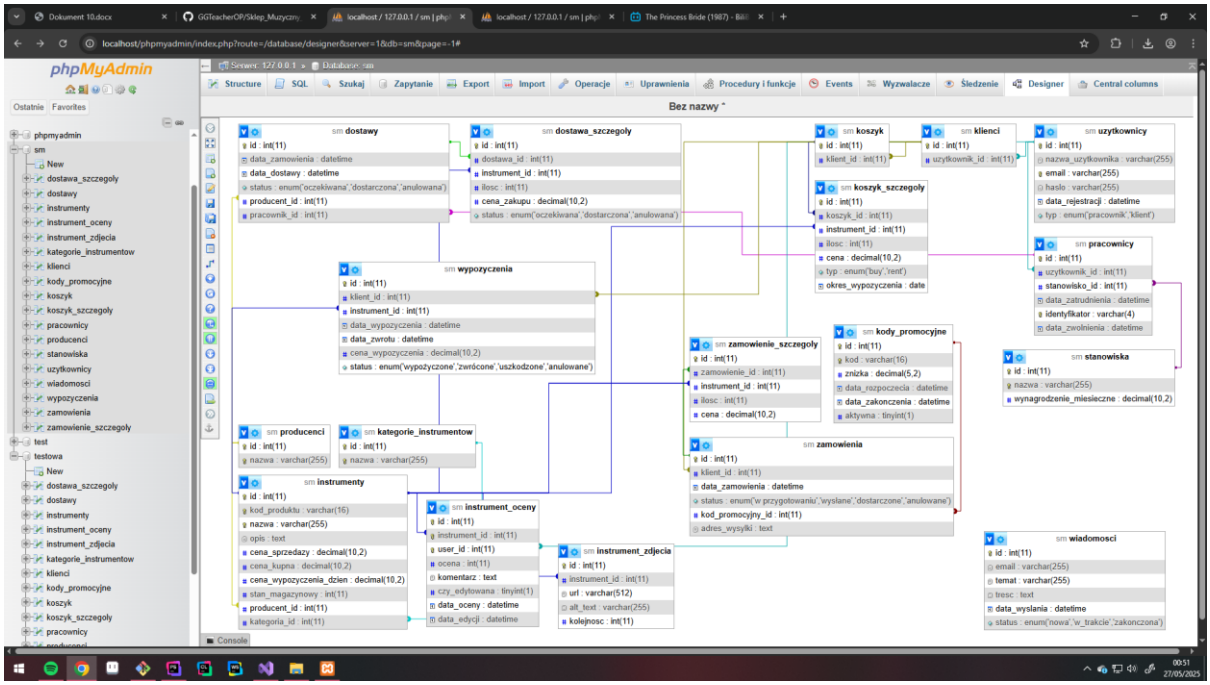
17. **klienci ↔ wypozyczenia**
    a. `wypozyczenia.klient_id → klienci.id`
    b. Wypożyczenie jest dokonywane przez jednego klienta
18. **instrumenty ↔ wypozyczenia**

a. `wypozyczenia.instrument_id` → `instrumenty.id`
b. Wypożyczenie dotyczy jednego instrumentu

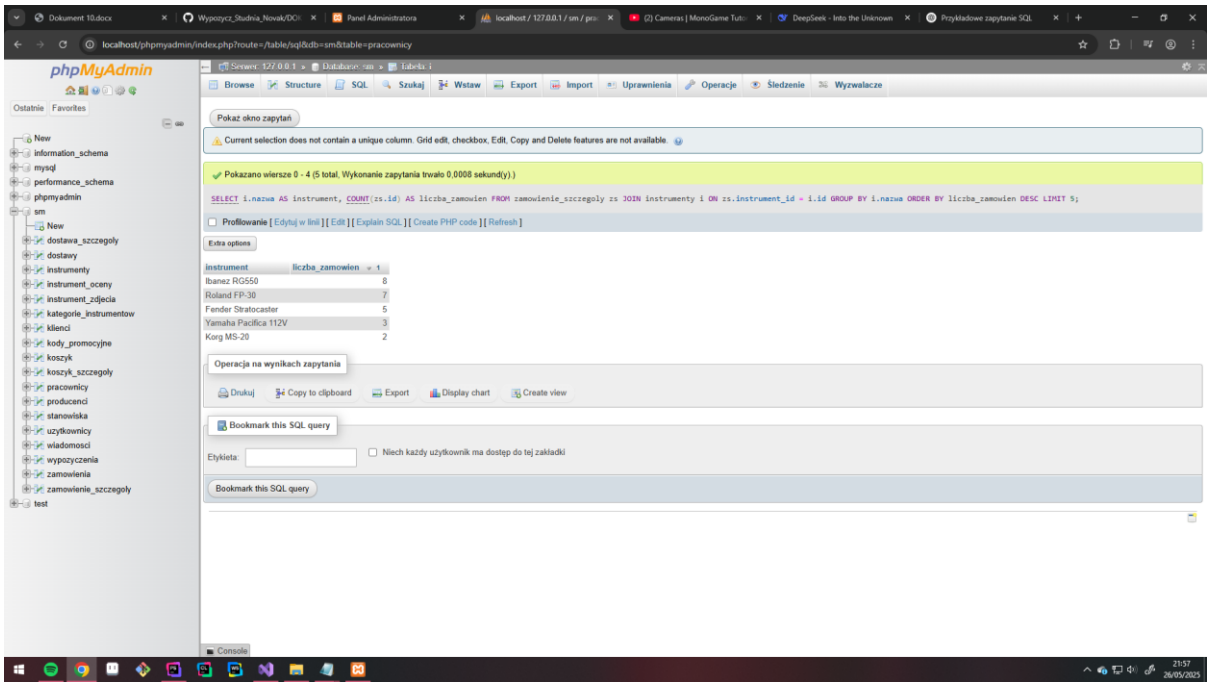
Relacje związane z ocenami i zdjęciami

19. **instrumenty ↔ instrument_oceny**
    a. `instrument_oceny.instrument_id` → `instrumenty.id`
    b. Każda ocena dotyczy jednego instrumentu
20. **uzytkownicy ↔ instrument_oceny**
    a. `instrument_oceny.user_id` → `uzytkownicy.id`
    b. Każda ocena jest wystawiona przez jednego użytkownika
21. **instrumenty ↔ instrument_zdjecia**
    a. `instrument_zdjecia.instrument_id` → `instrumenty.id`
    b. Każde zdjęcie jest przypisane do jednego instrumentu

7.3. Diagram ERD



7.5. Przykładowe zapytanie



To zapytanie:

1. **Łączy** tabelę `zamowienie_szczegoly` z tabelą `instrumenty` na podstawie `instrument_id`.
2. **Zlicza**, ile razy każdy instrument pojawił się w zamówieniach (`COUNT(zs.id)`).
3. **Grupuje** wyniki po nazwie instrumentu.
4. **Sortuje** wynik malejąco po liczbie zamówień.
5. **Zwraca** maksymalnie 5 najczęściej zamawianych instrumentów.

8. Bezpieczeństwo

8.1. Walidacja danych

- Sanityzacja danych wejściowych

```php
function handleEmployeeLogin(mysqli $connection, array &$errors, array &$values): bool {
    if (empty($_POST['employeeId'])) return false;

    $employee_id = $_POST['employeeId'];
    $employee_password = $_POST['employeePassword'];
    $values['employee_id'] = htmlspecialchars($employee_id);

    $query = "
        SELECT pracownicy.*, uzytkownicy.haslo
        FROM pracownicy
        JOIN uzytkownicy ON pracownicy.uzytkownik_id = uzytkownicy.id
        WHERE pracownicy.identyfikator = ?
    ";
    $stmt = mysqli_prepare($connection, $query);
    mysqli_stmt_bind_param($stmt, types: 's', &...vars: $employee_id);
    mysqli_stmt_execute($stmt);
    $result = mysqli_stmt_get_result($stmt);

    if ($employee = mysqli_fetch_assoc($result)) {
        if ($employee_password === $employee['haslo']) {
            $_SESSION['user_id'] = $employee['uzytkownik_id'];
            $_SESSION['employee_id'] = $employee['identyfikator'];;
            loadUserCart($connection, $employee['uzytkownik_id']);
            header( header: "Location: home.php");
            exit();
        } else {
            $errors['employee_password'] = "Nieprawidłowe hasło.";
        }
    } else {
        $errors['employee'] = "Nie znaleziono pracownika z tym ID.";
    }

    mysqli_stmt_close($stmt);
    return true;
}
```

- Walidacja formularzy

```
function validateForm() {
  const formAction = document.getElementById('formAction').value;

  if (formAction === 'add' || formAction === 'edit') {
    const kod = document.getElementById('kod_produktu').value;
    const nazwa = document.getElementById('nazwa').value;
    const opis = document.getElementById('opis').value;
    const cena = document.getElementById('cena_sprzedazy').value;
    const stan = document.getElementById('stan_magazynowy').value;
    const producent = document.getElementById('selectedProducent').value;
    const kategoria = document.getElementById('selectedCategory').value;

    if (!kod || !nazwa || !opis || !cena || !stan || !producent || !kategoria) {
      alert('Wypełnij wszystkie pola formularza');
      return false;
    }

    if (parseFloat(cena) <= 0) {
      alert('Cena musi być większa od 0');
      return false;
    }

    if (parseInt(stan) < 0) {
      alert('Stan magazynowy nie może być ujemny');
      return false;
    }
  } else if (formAction === 'edit_stock') {
    const stan = document.getElementById('stan_magazynowy').value;

    if (stan === '' || parseInt(stan) < 0) {
      alert('Stan magazynowy nie może być pusty ani ujemny');
      return false;
    }
  }

  return true;
}
```

- Zabezpieczenie przed SQL Injection (prepared statements)

```php
if ($search_query) {
    $query .= " AND (i.nazwa LIKE ? OR i.opis LIKE ? OR i.kod_produktu LIKE ?)";
    $search_param = "%$search_query%";
    $params[] = $search_param;
    $params[] = $search_param;
    $params[] = $search_param;
    $types .= "sss";
}

$stmt = $connection->prepare($query);

if (!empty($params)) {
    $stmt->bind_param($types, ...$params);
}

$stmt->execute();
$result = $stmt->get_result();
$row = $result->fetch_assoc();
```

- XSS Protection

8.2. Obsługa sesji

- Bezpieczne zarządzanie sesjami
- Timeout sesji

Zrealizowane za pomocą pliku session_config.php

```php
<?php
  if (session_status() !== PHP_SESSION_ACTIVE) {
    session_start();
  }

  $timeout = 15 * 60;

  if (isset($_SESSION['last_activity']) && (time() - $_SESSION['last_activity']) > $timeout) {
    session_unset();
    session_destroy();
    header( header: "Location: profile.php");
    exit;
  }

  $_SESSION['last_activity'] = time();
?>
```

8.3. Autoryzacja

- Role użytkowników
- Kontrola dostępu

9. Możliwości rozbudowy

9.1. Funkcjonalności

- System newslettera
- Program lojalnościowy
- Blog muzyczny
- Forum użytkowników
- Wypożyczanie instrumentów (teoretycznie jest ono połowicznie zaimplementowane jednak nie trafiło do wersji v1.0.0)

9.2. Techniczne

- API REST
- Aplikacja mobilna
- Integracja z systemami płatności
- System powiadomień push

9.3. Biznesowe

- System dropshippingu
- Program partnerski
- System rabatów grupowych
- Sprzedaż B2B

10. Załączniki

10.1. Lista błędów/uwag i pomysłów na ulepszenia

- Najprawdopodobniej nie wszystkie formularze po wywołaniu używają funkcji header() oraz exit() by zapobiec ponownemu wysłaniu formularza z takimi samymi danymi, co może prowadzić do błędu, gdy np. spróbujemy usunąć drugi raz ten sam rekord z bazy, co oczywiście wyrzuci błąd ze względu na brak tego rekordu w bazie.

- Najprawdopodobniej nie wszystkie zapytania zapisane w języku PHP zostały stworzone z użyciem prepared statementów (głównie te, które utworzone zostały na samym początku projektu i od tamtej pory nie były edytowane), co może, ale nie musi prowadzić do możliwości przeprowadzenia SQL Injection.

- Nie wszystkie akcje wykonywane są za pomocą AJAX, co sprawia, że strona jest odświeżana, gdy chcemy zastosować zmiany np. w filtrowaniu katalogu produktów, co może prowadzić do frustracji użytkownika ciągłym przeładowywaniem strony i koniecznością klikania przycisku do zastosowania filtrów.