Blok Services - Dokumentacja Projektu

O Projekcie

Blok Services to aplikacja webowa napisana w PHP, która oferuje użytkownikom różne narzędzia i funkcje społecznościowe. Aplikacja ma modularną strukturę, gdzie każda funkcja jest podzielona na oddzielne katalogi.

Struktura Głównych Katalogów

Katalog	Przeznaczenie	Opis
api/	Komunikacja z serwerem	Wszystkie pliki PHP obsługujące żądania z przeglądarki
pages/	Strony aplikacji	Każda strona ma własny katalog z plikami PHP, JS i CSS
page-container/	Podstawa stron	Pliki które są wspólne dla wszystkich stron
controls/	Komponenty interfejsu	Elementy które mogą być używane na różnych stronach
images/	Grafiki	Obrazy i ikony używane w aplikacji
documents/	Dokumenty prawne	Regulamin i polityka prywatności
db_import/	Baza danych	Skrypty do tworzenia struktury bazy danych

---

Architektura Bazy Danych

Przegląd Systemu Bazy Danych

Blok Services wykorzystuje bazę danych MySQL (MariaDB 10.4.32) z charakterem UTF-8 dla pełnej obsługi znaków międzynarodowych. Baza składa się z 13 powiązanych tabel, które można podzielić na kilka głównych grup funkcjonalnych.

Grupy Funkcjonalne Tabel

1. Zarządzanie Użytkownikami

Tabela `users` - Centralny punkt systemu przechowujący wszystkie informacje o użytkownikach:

- **Identyfikacja**: `id` (klucz główny), `login` (unikalny), `email` (unikalny), `phone_number` (unikalny)

- **Bezpieczeństwo**: `password` (zahashowane hasło), `is_admin` (uprawnienia administratora)
- **Dane osobowe**: `name`, `last_name`, `gender`, `birth_date`, `description`
- **Personalizacja**: `avatar` (ścieżka do zdjęcia profilowego)
- **Aktywność**: `register_date`, `last_login_date`, `last_logout_date`

Tabela `user_settings` - Personalne preferencje każdego użytkownika:

- **Wygląd**: `current_theme` (Light/Dark)
- **Prywatność**: `profile_visibility` (public/friends)
- **Audyt**: `last_modification_date`

Połączenie: `user_settings.user_id` → `users.id` (relacja 1:1)

2. System Społecznościowy

Tabela `friend_requests` - Zarządzanie relacjami między użytkownikami:

- **Uczestnicy**: `sender_user_id`, `reciver_user_id` (oba łączą się z `users.id`)
- **Status**: `status` (pending/accepted/rejected)
- **Czas**: `creation_date`

Tabela `messages` - Prywatna komunikacja między użytkownikami:

- **Uczestnicy**: `sender_user_id`, `reciver_user_id`
- **Treść**: `content` (tekst wiadomości)
- **Zarządzanie**: `creation_date`, `read_date` (sprawdzenie czy przeczytana)

Tabela `posts` - Publiczne wpisy użytkowników:

- **Autor**: `owner_user_id`
- **Zawartość**: `title`, `content`
- **Kontrola publikacji**: `publication_date`, `access_level` (private/public/friends)
- **Interakcje**: `is_commentable` (czy można komentować)
- **Audyt**: `creation_date`, `last_modification_date`

Tabela `posts_comments` - Komentarze do postów:

- **Powiązania**: `post_id` → `posts.id`, `comment_author_id` → `users.id`
- **Zawartość**: `content`
- **Audyt**: `creation_date`, `last_modification_date`

Tabela `posts_likes` - Polubienia postów:

- **Powiązania**: `post_id` → `posts.id`, `user_id` → `users.id`
- **Czas**: `reaction_data` (kiedy polubiono)

Tabela `posts_comments_likes` - Polubienia komentarzy:

- **Powiązania**: `post_comment_id` → `posts_comments.id`, `user_id` → `users.id`
- **Czas**: `reaction_data`

3. System Gier

Tabela `games` - Katalog dostępnych gier w systemie:

- **Identyfikacja**: `id`, `title`
- **Dostęp**: `game_path` (URL do gry), `image_path` (miniatura)
- **Opis**: `description`, `type` (21 różnych kategorii od Action po Educational)
- **Oceny**: `average_rating` (średnia ocena użytkowników)
- **Audyt**: `creation_date`

Aktualnie w systemie znajduje się 16 gier, od klasycznych jak Agar.io po strategiczne jak Chess.com

Tabela `games_library` - Osobiste biblioteki gier użytkowników:

- **Powiązania**: `game_id` → `games.id`, `owner_user_id` → `users.id`
- **Personalizacja**: `alias_name` (własna nazwa gry), `is_favourite` (ulubiona)
- **Aktywność**: `library_game_added_date`, `last_play_date`
- **Ocena**: `user_rating` (osobista ocena użytkownika)

4. System Multimediów

Tabela `resources` - Fizyczne pliki w systemie:

- **Właściciel**: `owner_user_id` → `users.id`
- **Lokalizacja**: `path` (ścieżka do pliku)
- **Audyt**: `creation_date`

Tabela `media` - Metadane plików multimedialnych:

- **Powiązania**: `resource_id` → `resources.id` (unikalny), `owner_user_id` → `users.id`
- **Opis**: `title`, `description`, `type` (Picture/Video/Music/E-book/Document/Other)
- **Dostęp**: `access_level` (private/public/friends)
- **Audyt**: `creation_date`, `last_modification_date`

5. System Komunikacji

Tabela `newsletters_articles` - Artykuły i ogłoszenia systemowe:

- **Autor**: `author_id` → `users.id`
- **Typ**: `type` (news/warning/offer/promotion)
- **Zawartość**: `title` (ograniczone do 50 znaków), `content`
- **Czas**: `creation_date`

Kluczowe Relacje i Ograniczenia

Ograniczenia Integralności (Foreign Keys)

System wykorzystuje ścisłe ograniczenia referencyjne z różnymi strategiami:

CASCADE (usuwanie kaskadowe):

- Usunięcie użytkownika automatycznie usuwa wszystkie jego dane (posty, komentarze, pliki)
- Usunięcie posta usuwa wszystkie jego komentarze i polubienia
- To zapewnia czystość bazy danych i zapobiega "osieroconymi" rekordami

NO ACTION (bez akcji):

- Wiadomości zachowują się nawet po usunięciu użytkownika (dla celów audytu)

- Pozwala na zachowanie historii komunikacji

Indeksy i Wydajność

System zawiera przemyślane indeksy:

- **Klucze główne**: Automatyczne indeksy dla szybkiego dostępu
- **Klucze obce**: Indeksy dla szybkich joinów między tabelami
- **Unikalne ograniczenia**: Na `login`, `email`, `phone_number` w tabeli users

Bezpieczeństwo Bazy Danych

Enkodowanie i Sortowanie

- **Charset**: utf8mb4 - pełna obsługa UTF-8 włącznie z emoji
- **Collation**: utf8mb4_general_ci - sortowanie case-insensitive

Typy Danych i Walidacja

- **ENUM**: Ograniczone wartości dla statusów, typów, poziomów dostępu
- **Unsigned INT**: Tylko pozytywne ID dla kluczy głównych
- **TEXT vs VARCHAR**: Optymalny wybór dla różnych długości tekstów
- **DATETIME**: Precyzyjne znaczniki czasu z domyślnymi wartościami

Przykłady Użycia Bazy Danych

Pobieranie Postów Użytkownika z Komentarzami:

```sql
SELECT
    p.title, p.content, p.creation_date,
    u.name, u.last_name,
    COUNT(pl.id) as likes_count,
    COUNT(pc.id) as comments_count
FROM posts p
JOIN users u ON p.owner_user_id = u.id
LEFT JOIN posts_likes pl ON p.id = pl.post_id
LEFT JOIN posts_comments pc ON p.id = pc.post_id
WHERE p.access_level = 'public'
GROUP BY p.id
ORDER BY p.creation_date DESC;
```

Znajdowanie Popularnych Gier:

```sql
SELECT
    g.title, g.type, g.average_rating,
    COUNT(gl.id) as users_count,
    AVG(gl.user_rating) as user_avg_rating
FROM games g
LEFT JOIN games_library gl ON g.id = gl.game_id
GROUP BY g.id
HAVING users_count > 0
ORDER BY g.average_rating DESC, users_count DESC;
```

System API - Jak Aplikacja Komunikuje się z Serwerem

API to zbiór plików PHP, które obsługują różne akcje użytkowników. Każdy typ funkcji ma swój własny katalog:

Chat (Czat)

Plik	Co robi
send_message.php	Wysyła nową wiadomość
get_messages.php	Pobiera wszystkie wiadomości
delete_message.php	Usuwa wybraną wiadomość
update_message.php	Edytuje istniejącą wiadomość
get_users.php	Pobiera listę użytkowników
clear_chat.php	Czyści całą historię czatu

Social (Media Społecznościowe)

Plik	Co robi
add_post.php	Dodaje nowy post
edit_post.php	Edytuje istniejący post
delete_post.php	Usuwa post
add_comment.php	Dodaje komentarz do posta
edit_comment.php	Edytuje komentarz
delete_comment.php	Usuwa komentarz
like_post.php	Polubienie posta
like_comment.php	Polubienie komentarza
posts_category_filter.php	Filtruje posty według kategorii

Games (Gry)

Plik	Co robi
like_game.php	Pozwala polubić grę

Strony Aplikacji - Co Może Robić Użytkownik

Każda strona ma identyczną strukturę plików, co ułatwia dodawanie nowych funkcji:

Struktura Każdej Strony

Plik/Katalog	Przeznaczenie
index.php	Główny plik strony - to go otwiera przeglądarka
content.php	Treść strony (HTML)
scripts/	Pliki JavaScript dla interakcji
styles/	Pliki CSS dla wyglądu

Lista Wszystkich Stron

Strona	Co oferuje	Dodatkowe pliki procedur
welcome-page	Strona powitalna dla nowych użytkowników	Pierwsza strona którą widzi gość
main-page	Strona główna po zalogowaniu	Centrum aplikacji dla zalogowanych
login-page	Logowanie do systemu	login-procedure.php, logout-procedure.php
register-page	Rejestracja nowego konta	register-procedure.php
reset-password-page	Resetowanie hasła	Odzyskiwanie dostępu do konta
profile-page	Profil użytkownika	delete-account-procedure.php
modify-account-page	Edycja danych konta	modify-account-procedure.php
settings-page	Ustawienia aplikacji	Personalizacja interfejsu
chat-page	System czatu	Komunikacja między użytkownikami
social-page	Media społecznościowe	Posty, komentarze, polubienia
post-page	Wyświetlanie pojedynczego posta	Szczegółowy widok posta
search-page	Wyszukiwarka	Znajdowanie treści w aplikacji
games-page	Sekcja gier	Lista gier do grania
gallery-page	Galeria zdjęć	Przeglądanie obrazów
media-page	Multimedia	Zarządzanie plikami audio/video
drive-page	Dysk internetowy	Przechowywanie plików
newsletters-page	Biuletyny informacyjne	Zarządzanie subskrypcjami
applications-page	Lista aplikacji	Katalog dostępnych narzędzi
calculator-page	Kalkulator online	Narzędzie do obliczeń
clock-page	Zegar i czas	Informacje o czasie
contact-page	Kontakt	Formularz kontaktowy
_template-page	Szablon dla nowych stron	Wzorzec do kopiowania

---

Pliki Konfiguracyjne - Ustawienia Aplikacji

Plik	Przeznaczenie	Dlaczego ważny
config.json	Główne ustawienia	Wszystkie parametry aplikacji w jednym miejscu
_template.html	Szablon HTML	Podstawowa struktura każdej strony

---

System Kontenerów Stron - Jak Budowane są Strony

Każda strona w aplikacji jest budowana według tego samego wzorca:

Plik	Kiedy się wykonuje	Co robi
page-container-pre.php	Przed treścią strony	Ładuje wspólne elementy (menu, nagłówek)
page-container-post.php	Po treści strony	Zamyka stronę (stopka, skrypty)
_page-container.php	Główny kontroler	Łączy wszystko w całość
db.php	Na początku	Łączy się z bazą danych
json-config-load.php	Na początku	Ładuje ustawienia z config.json

---

Komponenty Wielokrotnego Użytku

Komponent	Gdzie się używa	Pliki
Popup Window	Na różnych stronach	popup-window.js, popup-window.css, popup-examples.js
Nawigacja	Na każdej stronie	navigation/nav.php

Jak Uruchomić Projekt

1. **Przygotowanie bazy danych** - Zaimportuj plik `db_import/blok_services.sql` do swojej bazy MySQL
2. **Konfiguracja** - Dostosuj ustawienia w pliku `config.json`
3. **Serwer web** - Umieść pliki na serwerze z obsługą PHP
4. **Pierwsze uruchomienie** - Wejdź na stronę przez przeglądarkę

Dlaczego Taka Struktura?

Ta organizacja plików i bazy danych ma kilka zalet:

Łatwość w utrzymaniu - Każda funkcja ma swoje miejsce, więc łatwo znaleźć co trzeba zmienić. Baza danych jest znormalizowana, co eliminuje redundancję danych.

Możliwość rozbudowy - Dodanie nowej strony to skopiowanie struktury istniejącej i dostosowanie zawartości. Nowe tabele można łatwo dodać dzięki przemyślanej architekturze.

Praca zespołowa - Różne osoby mogą pracować nad różnymi stronami bez konfliktów. Relacje w bazie danych są jasno zdefiniowane.

Bezpieczeństwo - API jest oddzielone od stron, co pozwala lepiej kontrolować dostęp do danych. Ograniczenia integralności w bazie zapobiegają nieprawidłowym danym.

Wydajność - Przemyślane indeksy i relacje zapewniają szybkie zapytania nawet przy dużej ilości danych.

Następne Kroki

Jeśli chcesz rozwijać ten projekt, zacznij od:

1. **Przeanalizowania pliku config.json** aby zrozumieć ustawienia
2. **Sprawdzenia jak działa _page-container.php** aby zrozumieć jak budowane są strony
3. **Przyjrzenia się istniejącym stronom** jako wzorcom do nowych funkcji
4. **Zrozumienia relacji w bazie danych** - jak tabele są ze sobą połączone
5. **Przetestowania zapytań SQL** na przykładowych danych aby zobaczyć jak działa system

Przydatne Narzędzia do Rozwoju

- **phpMyAdmin** - do zarządzania bazą danych
- **MySQL Workbench** - do wizualizacji relacji między tabelami
- **Postman** - do testowania API endpoints
- **Browser Developer Tools** - do debugowania JavaScript i CSS