QR System Pro - Dokumentacja Techniczna

Wersja: 1.0.0
Autor: MiniMax Agent
Ostatnia aktualizacja: 2025-07-01

Spis treści

---

1. Wprowadzenie i przegląd

Opis wtyczki i jej zastosowania

QR System Pro to zaawansowana wtyczka dla WordPress, która umożliwia tworzenie, zarządzanie i śledzenie kodów QR. Została zaprojektowana z myślą o firmach, marketerach i deweloperach, którzy potrzebują niezawodnego i skalowalnego rozwiązania do integracji kodów QR ze swoimi strategiami marketingowymi, systemami logistycznymi czy aplikacjami.

Wtyczka oferuje kompleksowy zestaw narzędzi, od generowania pojedynczych i masowych kodów, przez zaawansowaną analitykę skanowań w czasie rzeczywistym, po rozbudowane opcje bezpieczeństwa i archiwizacji danych.

Główne funkcjonalności

- **Zarządzanie kodami QR**: Pełen system CRUD (Create, Read, Update, Delete) dla kodów QR, z możliwością ich grupowania, przypisywania do użytkowników i ustawiania daty ważności.
- **Historia skanowań**: Szczegółowy rejestr każdego skanowania, zawierający informacje o czasie, lokalizacji (geolokalizacja) i urządzeniu skanującym.

- **Interaktywne statystyki**: Dynamiczne wykresy i tabele prezentujące dane o skanowaniach, popularności kodów i aktywności użytkowników.
- **Archiwizacja danych**: Automatyczne archiwizowanie i kompresja starych danych (skany, logi) w celu utrzymania optymalnej wydajności bazy danych.
- **Zaawansowane bezpieczeństwo**: Wbudowane mechanizmy chroniące przed atakami (np. rate limiting), system weryfikacji uprawnień oraz szczegółowe logowanie zdarzeń.
- **REST API i AJAX**: Pełne wsparcie dla zdalnej interakcji z wtyczką, umożliwiające integrację z zewnętrznymi systemami.
- **Import/Eksport danych**: Możliwość masowego importu kodów z plików CSV oraz eksportu danych o skanach do formatów CSV, JSON i XML.

Wymagania systemowe

- **WordPress:** 6.0 lub nowszy
- **PHP:** 8.0 lub nowszy
- **Baza danych:** MySQL 5.7+ lub MariaDB 10.3+
- **Serwer:** Zalecany serwer z obsługą Redis dla optymalnego działania cache.
- **Rozszerzenia PHP:** `mbstring`, `json`, `gd`

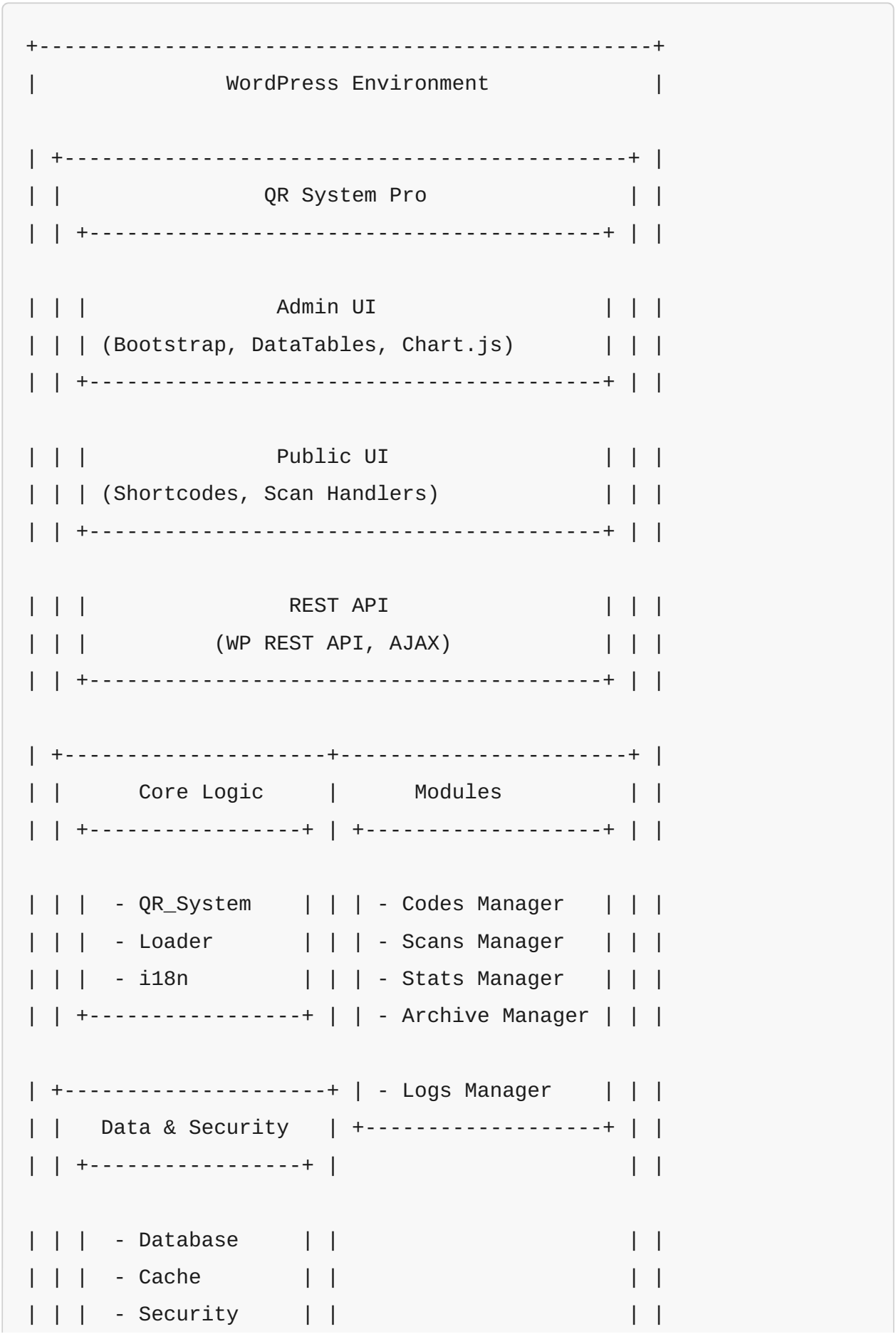
Kompatybilność

Wtyczka jest kompatybilna z większością nowoczesnych motywów i wtyczek WordPress. Została przetestowana z najpopularniejszymi wtyczkami do cache, bezpieczeństwa i e-commerce.

---

2. Architektura systemu

Diagram architektury


WordPress Environment
QR System Pro
Admin UI
(Bootstrap, DataTables, Chart.js)
Public UI
(Shortcodes, Scan Handlers)
REST API
(WP REST API, AJAX)
Core Logic
- QR_System
- Loader
- i18n
Data & Security
- Database
- Cache
- Security
Modules
- Codes Manager
- Scans Manager
- Stats Manager
- Archive Manager
- Logs Manager

```
| | +-----------------+ |                        | |

| +-----------------------------------------------+ |
| |         WordPress Core & Database             | |
| +-----------------------------------------------+ |
+-------------------------------------------------+
```

Wzorce projektowe

Wtyczka wykorzystuje nowoczesne wzorce projektowe w celu zapewnienia skalowalności, elastyczności i łatwości utrzymania kodu:

- **Singleton**: Główna klasa wtyczki `QR_System` oraz klasy zarządzające (np. `Database`, `Cache`) są implementowane jako Singletony, aby zapewnić jedną, globalną instancję tych obiektów.
- **Factory**: Wzorce fabryki są używane do tworzenia obiektów modułów, co ułatwia dodawanie nowych modułów bez modyfikacji rdzenia systemu.
- **Observer**: System hooków WordPress (`add_action`, `add_filter`) działa w oparciu o wzorzec obserwatora, co pozwala na rozszerzanie funkcjonalności wtyczki w sposób modułowy.
- **Repository**: Klasy menedżerów modułów (np. `QR_System_Codes_Manager`) pełnią rolę repozytoriów, abstrahując logikę dostępu do danych od reszty aplikacji.

Struktura katalogów

```
/qr-system/
|-- qr-system.php             # Główny plik wtyczki

|-- includes/                 # Klasy rdzenia
|   |-- class-qr-system.php   # Główna klasa
|   |-- class-loader.php      # Zarządzanie hookami

|   |-- class-database.php    # Obsługa bazy danych
|   |-- class-cache.php       # System cache
|   |-- class-security.php    # Funkcje bezpieczeństwa

|   |-- class-api.php         # REST API i AJAX
|   |-- class-i18n.php        # Internacjonalizacja
|   |-- class-activator.php   # Logika aktywacji
|   `-- class-deactivator.php # Logika deaktywacji

|-- admin/                    # Panel administracyjny
|   |-- class-admin.php       # Główna klasa admina
|   |-- css/                  # Style CSS
|   |-- js/                   # Skrypty JavaScript
|   `-- partials/             # Szablony widoków

|-- public/                   # Część publiczna
|   |-- class-public.php      # Główna klasa publiczna
|   |-- css/
|   |-- js/
|   `-- partials/

|-- modules/                  # Moduły funkcjonalne
|   |-- codes/                # Zarządzanie kodami
|   |   `-- class-codes-manager.php

|   |-- scans/                # Zarządzanie skanami
|   |   `-- class-scans-manager.php
|   |-- stats/                # Statystyki
```

```
|   |   `-- class-stats-manager.php
|   |-- archive/              # Archiwizacja
|   |   `-- class-archive-manager.php
|   `-- logs/                 # Logi
|       `-- class-logs-manager.php
|-- languages/                # Pliki tłumaczeń (.pot, .po, .mo)
`-- vendor/                   # Zależności Composer
```

Przepływ danych

1. **Żądanie (Request)**: Użytkownik wchodzi w interakcję z panelem admina, stroną publiczną lub wysyła żądanie do API.
2. **Routing**: WordPress kieruje żądanie do odpowiedniego handlera (w `admin/class-admin.php`, `public/class-public.php` lub `includes/class-api.php`).
3. **Przetwarzanie**: Handler wywołuje odpowiednie metody w menedżerach modułów (`modules/*`).
4. **Dostęp do danych**: Menedżer modułu komunikuje się z warstwą danych (`Database`, `Cache`) w celu pobrania lub zapisu informacji.
5. **Logika biznesowa**: Menedżer przetwarza dane zgodnie z logiką biznesową.
6. **Odpowiedź (Response)**: Wynik jest formatowany i zwracany do użytkownika w postaci widoku HTML, odpowiedzi JSON (dla API) lub przekierowania.

---

3. Instalacja i konfiguracja

Proces instalacji krok po kroku

1. Pobierz plik `.zip` wtyczki.
2. Zaloguj się do panelu administracyjnego WordPress.

3. Przejdź do `Wtyczki > Dodaj nową`.
4. Kliknij przycisk `Wyślij wtyczkę na serwer`.
5. Wybierz pobrany plik `.zip` i kliknij `Zainstaluj teraz`.
6. Po zakończeniu instalacji kliknij `Włącz wtyczkę`.

Po aktywacji wtyczka automatycznie utworzy niezbędne tabele w bazie danych oraz domyślne ustawienia.

Konfiguracja początkowa

Po instalacji przejdź do menu `QR System Pro > Ustawienia`. Główne opcje konfiguracyjne to:

- **Ustawienia ogólne**: Klucz API Google Maps (dla geolokalizacji), domyślne ustawienia dla nowych kodów.
- **Ustawienia cache**: Włącz/wyłącz Redis, skonfiguruj czas życia cache dla różnych obiektów.
- **Ustawienia bezpieczeństwa**: Skonfiguruj limity zapytań (rate limiting), włącz/wyłącz logowanie zdarzeń.
- **Ustawienia archiwizacji**: Ustaw harmonogram automatycznej archiwizacji (np. co tydzień) i okres przechowywania archiwów.

Migracja z starych systemów

Wtyczka oferuje narzędzie do importu kodów z plików CSV. Aby przeprowadzić migrację:

1. Przygotuj plik CSV z kodami w formacie: `code,type,group,valid_until`.
2. Przejdź do `QR System Pro > Kody QR > Importuj`.
3. Wybierz plik CSV i zmapuj kolumny do odpowiednich pól wtyczki.
4. Kliknij `Importuj`, aby rozpocząć proces.

Rozwiązywanie problemów instalacyjnych

- **Błąd "Wymagana wersja PHP"**: Upewnij się, że Twój serwer używa PHP w wersji 8.0 lub nowszej.
- **Błąd tworzenia tabel w bazie danych**: Sprawdź, czy użytkownik bazy danych WordPress ma uprawnienia `CREATE` i `ALTER`.
- **Konflikty z innymi wtyczkami**: Jeśli po aktywacji wystąpią problemy, spróbuj wyłączyć inne wtyczki, aby zidentyfikować potencjalny konflikt.

---

4. Dokumentacja API

REST API endpoints

Wtyczka rejestruje następujące endpointy w ramach przestrzeni nazw `qr-system/v1`:

Metoda	Endpoint	Opis	Uprawnienia
GET	`/codes`	Pobiera listę kodów QR	`manage_options`
POST	`/codes`	Tworzy nowy kod QR	`manage_options`
GET	`/codes/{id}`	Pobiera szczegóły kodu QR	`manage_options`
PUT	`/codes/{id}`	Aktualizuje kod QR	`manage_options`
DELETE	`/codes/{id}`	Usuwa kod QR	`manage_options`
GET	`/scans`	Pobiera historię skanowań	`manage_options`
GET	`/stats`	Pobiera statystyki	`manage_options`

AJAX handlers

Dostępne są również akcje AJAX dla interakcji z front-endu:

Akcja	Opis	Dostępność
`qr_scan_code`	Rejestruje skanowanie kodu (skaner)	Public
`qr_manual_code`	Rejestruje ręczne wprowadzenie kodu	Public

Parametry i odpowiedzi

Przykład: Tworzenie nowego kodu QR (POST /codes)

Parametry (JSON body):

```
{
  "code": "PROMO2025",
  "type": "discount",
  "group_id": 1,
  "valid_until": "2025-12-31 23:59:59",
  "meta": {
    "discount_value": "15%"
  }
}
```

Odpowiedź (201 Created):

```
{
  "id": 123,
  "code": "PROMO2025",
  "type": "discount",
  "status": "active",
  "created_at": "2025-07-01 10:00:00"
}
```

Przykłady użycia

Pobieranie listy kodów za pomocą cURL:

```
curl -X GET "https://twojadomena.pl/wp-json/qr-system/v1/codes" \
-H "Authorization: Bearer <TWÓJ_TOKEN_API>"
```

Rejestrowanie skanu za pomocą jQuery (AJAX):

```
jQuery.post(
    ajaxurl,
    {
        action: 'qr_scan_code',
        nonce: 'twoj_nonce',
        code: 'SCANNED_CODE'
    },
    function(response) {
        if(response.success) {
            alert('Kod został pomyślnie zweryfikowany!');
        } else {
            alert('Błąd: ' + response.data.message);
        }
    }
);
```

---

5. Baza danych

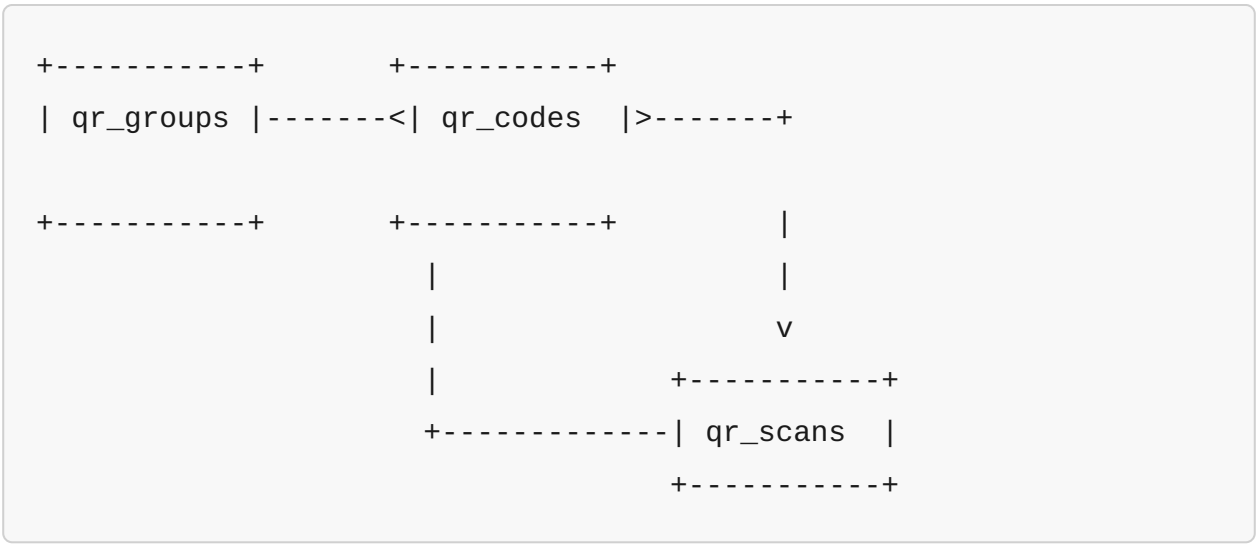
Schemat tabel

Wtyczka tworzy następujące tabele w bazie danych WordPress (z prefiksem `wp_`):

- `wp_qr_codes`: Przechowuje kody QR i ich metadane.
  - `id` (BIGINT, PK, AI)
  - `code` (VARCHAR)
  - `type` (VARCHAR)
  - `group_id` (BIGINT, FK)
  - `user_id` (BIGINT)
  - `status` (VARCHAR)
  - `valid_until` (DATETIME)
  - `created_at` (TIMESTAMP)
- `wp_qr_scans`: Historia skanowań.
  - `id` (BIGINT, PK, AI)
  - `code_id` (BIGINT, FK)
  - `scan_time` (TIMESTAMP)
  - `ip_address` (VARCHAR)
  - `user_agent` (TEXT)
  - `latitude` (DECIMAL)
  - `longitude` (DECIMAL)
- `wp_qr_groups`: Grupy kodów.
  - `id` (BIGINT, PK, AI)
  - `name` (VARCHAR)

- `wp_qr_logs`: Logi systemowe i bezpieczeństwa.
  - `id` (BIGINT, PK, AI)
  - `level` (VARCHAR)
  - `message` (TEXT)
  - `context` (JSON)
  - `created_at` (TIMESTAMP)
- `wp_qr_stats`: Cache dla statystyk.
  - `stat_key` (VARCHAR, PK)
  - `stat_value` (LONGTEXT)
  - `expires_at` (DATETIME)

Relacje między tabelami



Indeksy i optymalizacja

- Indeksy są założone na kolumnach `code` (`qr_codes`), `code_id` i `scan_time` (`qr_scans`) oraz kluczach obcych w celu przyspieszenia zapytań.
- Tabela `qr_stats` działa jako zmaterializowany widok, przechowując pre-agregowane dane statystyczne, co znacznie redukuje obciążenie bazy danych przy generowaniu raportów.

- Automatyczna archiwizacja i czyszczenie starych logów i skanów zapobiega nadmiernemu rozrostowi tabel.

Migracje

Wtyczka posiada wbudowany mechanizm migracji. Przy każdej aktualizacji, sprawdza ona wersję bazy danych i automatycznie wykonuje niezbędne zmiany w schemacie (np. dodanie nowych tabel, kolumn).

---

6. Bezpieczeństwo

Mechanizmy bezpieczeństwa

QR System Pro implementuje wielowarstwowe podejście do bezpieczeństwa, zgodne z rekomendacjami OWASP:

- **Input Sanitization**: Wszystkie dane wejściowe (z formularzy, API) są rygorystycznie walidowane i oczyszczane za pomocą funkcji WordPress (np. `sanitize_text_field`, `wp_kses_post`).
- **Output Escaping**: Wszystkie dane wyjściowe wyświetlane w HTML są eskejpowane (`esc_html`, `esc_attr`), aby zapobiec atakom XSS.
- **Nonces**: Każde żądanie AJAX i wysłanie formularza jest zabezpieczone za pomocą jednorazowych tokenów (nonces), aby chronić przed atakami CSRF.
- **Przygotowane zapytania SQL**: Wszystkie zapytania do bazy danych używają `wpdb::prepare()`, aby zapobiec atakom SQL Injection.

Rate limiting

Wtyczka posiada wbudowany mechanizm rate limiting, który ogranicza liczbę prób skanowania lub zapytań API z jednego adresu IP w określonym czasie. Konfiguracja (np. 100 zapytań na minutę) znajduje się w panelu ustawień.

Weryfikacja uprawnień

Każda akcja w panelu administracyjnym oraz każdy endpoint API wymaga odpowiednich uprawnień (capabilities) WordPress. Domyślnie, większość operacji wymaga uprawnienia `manage_options`, ale można to dostosować za pomocą filtrów.

Logowanie zdarzeń

System logowania (`QR_System_Logs_Manager`) rejestruje kluczowe zdarzenia, takie jak:

- Nieudane próby logowania
- Zablokowane żądania (rate limiting)
- Błędy krytyczne
- Operacje na kodach QR (tworzenie, usuwanie)

Logi są przechowywane przez 7 dni i można je przeglądać w panelu `QR System Pro > Logi`.

---

7. Cache i wydajność

Strategie cache

Wtyczka wykorzystuje wielowarstwowy system cache, aby zminimalizować liczbę zapytań do bazy danych i zapewnić maksymalną wydajność:

1. **Redis (opcjonalnie)**: Jeśli Redis jest dostępny na serwerze, wtyczka użyje go jako backendu dla WordPress Object Cache, co zapewnia najszybsze działanie.
2. **WordPress Transients API**: Używane do cachowania wyników złożonych zapytań i danych statystycznych. Transjenty mają zdefiniowany czas wygaśnięcia.
3. **WordPress Object Cache**: Wewnętrzny mechanizm cache WordPress, który przechowuje dane na czas jednego żądania.

Optymalizacja zapytań

- Zapytania do bazy danych zostały zoptymalizowane pod kątem wydajności.
- Dane statystyczne są agregowane w tle i przechowywane w tabeli `qr_stats`, aby uniknąć kosztownych operacji `GROUP BY` i `JOIN` w czasie rzeczywistym.

Monitoring wydajności

Wtyczka integruje się z popularnymi narzędziami do monitoringu wydajności (np. Query Monitor), wyświetlając liczbę i czas wykonania swoich zapytań SQL.

---

8. Przewodnik administratora

Zarządzanie kodami QR

Panel `QR System Pro > Kody QR` pozwala na:

- **Dodawanie nowych kodów**: Ręcznie lub poprzez import z pliku CSV.
- **Edycję istniejących kodów**: Zmianę typu, grupy, daty ważności.
- **Grupowanie kodów**: Przypisywanie kodów do grup w celu łatwiejszej organizacji.
- **Masowe operacje**: Usuwanie lub zmiana statusu wielu kodów jednocześnie.

Panel statystyk

Panel `QR System Pro > Statystyki` oferuje:

- **Interaktywne wykresy (Chart.js)**: Prezentujące liczbę skanowań w czasie, popularność kodów, podział na typy urządzeń.
- **Mapę skanowań**: Wizualizację geolokalizacji skanowań (wymaga klucza API Google Maps).
- **Filtrowanie i porównywanie okresów**: Analizę danych w wybranych zakresach czasowych.

Eksport i archiwizacja

- **Eksport**: Dane o skanowaniach można eksportować do formatów CSV, JSON i XML z panelu `Historia Skanów`.
- **Archiwizacja**: W panelu `Archiwum` można przeglądać i pobierać archiwa (pliki .zip) starych danych, które zostały automatycznie spakowane przez system.

Ustawienia systemu

Panel `QR System Pro > Ustawienia` jest podzielony na zakładki, umożliwiając intuicyjną konfigurację wszystkich aspektów wtyczki.

---

9. Przewodnik dewelopera

Rozszerzanie funkcjonalności

Wtyczka została zaprojektowana z myślą o rozszerzalności. Deweloperzy mogą dodawać nowe funkcje za pomocą standardowych mechanizmów WordPress (akcje i filtry).

Custom hooks i filtry

QR System Pro udostępnia szereg własnych hooków:

Akcje (Actions):

- `qr_system_after_scan( $scan_data )`: Wywoływana po pomyślnym zarejestrowaniu skanu.
- `qr_system_code_created( $code_id )`: Wywoływana po utworzeniu nowego kodu QR.
- `qr_system_daily_cleanup_done()`: Wywoływana po zakończeniu codziennego czyszczenia logów.

Filtry (Filters):

- `qr_system_api_permission_callback( <span class="math-inline" style="display: inline;"><math xmlns="http://www.w3.org/1998/Math/MathML" display="inline"><mrow><mi>p</mi><mi>e</mi><mi>r</mi><mi>m</mi><mi>i</mi><mi>s</mi><mi>s</mi><mi>i</mi><mi>o</mi><mi>n</mi><mo>,</mo></mrow></math></span>endpoint )`: Pozwala na modyfikację wymaganych uprawnień dla endpointów API.
- `qr_system_code_meta_fields( $fields )`: Umożliwia dodanie własnych pól meta do kodów QR.
- `qr_system_log_levels( $levels )`: Pozwala na dodanie niestandardowych poziomów logowania.

Tworzenie własnych modułów

Aby stworzyć własny moduł:

1. Utwórz nowy katalog w `/modules/`, np. `/modules/custom/`.
2. Stwórz w nim klasę menedżera, np. `class-custom-manager.php`.
3. Zarejestruj swój moduł i jego hooki w głównej klasie wtyczki (`includes/class-qr-system.php`) w metodzie `init_modules()`.
4. Wykorzystaj istniejące klasy `Database`, `Cache`, `Security` do interakcji z rdzeniem systemu.

Best practices

- **Używaj dostępnych hooków**: Zawsze staraj się używać akcji i filtrów zamiast modyfikować pliki rdzenia wtyczki.
- **Przestrzegaj struktury kodu**: Zachowaj spójność nazewnictwa i struktury katalogów.
- **Wykorzystuj API wtyczki**: Korzystaj z publicznych metod klas menedżerów (np. `QR_System_Codes_Manager::get_code()`) zamiast pisać własne zapytania SQL.

---

10. Rozwiązywanie problemów

Częste problemy i rozwiązania

- **Problem**: Wykresy statystyk się nie ładują.
  - **Rozwiązanie**: Sprawdź konsolę błędów JavaScript w przeglądarce. Upewnij się, że żadna inna wtyczka nie blokuje skryptów Chart.js.
- **Problem**: Import CSV nie działa.
  - **Rozwiązanie**: Sprawdź, czy plik CSV ma poprawne kodowanie (UTF-8) i czy separator kolumn jest zgodny z ustawieniami.
- **Problem**: Geolokalizacja nie jest zapisywana.
  - **Rozwiązanie**: Upewnij się, że w ustawieniach wtyczki został podany prawidłowy klucz API Google Maps i że jest on aktywny.

Diagnostyka

W panelu `QR System Pro > Ustawienia > Diagnostyka` znajduje się strona diagnostyczna, która sprawdza:

- Wersję PHP i WordPress.
- Dostępność wymaganych rozszerzeń PHP.
- Uprawnienia do zapisu w katalogach `uploads`.
- Status połączenia z Redis (jeśli włączony).

Logi i debugging

- **Logi systemowe**: Przejrzyj logi w `QR System Pro > Logi`, aby zidentyfikować potencjalne błędy operacyjne lub związane z bezpieczeństwem.
- **WordPress Debug Mode**: Włącz `WP_DEBUG`, `WP_DEBUG_LOG` i `WP_DEBUG_DISPLAY` w pliku `wp-config.php`, aby wyświetlić szczegółowe błędy PHP.

Kontakt z supportem

Jeśli powyższe kroki nie rozwiążą problemu, skontaktuj się z pomocą techniczną, podając następujące informacje:

- Wersja wtyczki QR System Pro.
- Wersja WordPress i PHP.
- Opis problemu i kroki, które prowadzą do jego wystąpienia.
- Zrzut ekranu strony diagnostycznej.
- Odpowiednie fragmenty z logów błędów.