

POLITECHNIKA ŚWIĘTOKRZYSKA
Wydział Elektrotechniki, Automatyki i Informatyki

Maciej Bandura
Numer albumu: 91234

Marcin Ślusarczyk
Numer albumu: 91348

Programowa Sieć Dostarczania Zawartości

**Praca dyplomowa
na studiach I-go stopnia
na kierunku Informatyka**

Promotor pracy dyplomowej:
dr inż. Tomasz Kaczmarek
Katedra Systemów Informatycznych

POLITECHNIKA ŚWIĘTOKRZYSKA
WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI I INFORMATYKI

Studia stacjonarne

Kierunek: Informatyka

Zatwierdzam
dr. inż. Barbara Łukawska
Wydzięlu Elektrotechniki, Automatyki i Informatyki

.....
dr. inż. Barbara Łukawska

Rok akademicki: 2023/24

BL

Barbara Łukawska

**ZADANIE NA PRACĘ DYPLOMOWĄ
STUDIÓW PIERWSZEGO STOPNIA**

Wydano studentowi:

Maciej Bandura

nr albumu: 91234

I. Temat pracy:

**Programowa Sieć Dostarczania Zawartości
Software Content Delivery Network**

II. Cel pracy:

Celem pracy jest projekt i implementacja serwera źródłowego do świadczenia usługi CDN na dostępnej komercyjnie infrastrukturze serwerowej hostingującej strony internetowe ogólnego przeznaczenia. Przykładowe funkcjonalności: przekierowywanie żądań o treści na najbliższe skrajne serwery, zbieranie informacji o infrastrukturze, gromadzenie ustawień użytkowników oraz ich przesyłanie na skrajne serwery, a także zapewnienie interfejsów do komunikacji z serwerami skrajnymi. Po integracji z projektem stworzonym przez studenta Marcina Ślusarczyka powstanie kompletne rozwiązanie problemu budżetowego świadczenia usługi CDN w sposób programowy.

III. Plan pracy (zakres pracy):

1. Zapoznanie się z dziedziną, w szczególności z zagadnieniami dotyczącymi usługi CDN.
2. Opracowanie projektu Programowej Sieci Dostarczania Zawartości w zakresie serwera źródłowego.
3. Dobór języka programowania, systemu bazodanowego oraz innych narzędzi programistycznych.
4. Implementacja oraz wdrożenie serwera źródłowego zgodnie z celem pracy.
5. Testy opracowanego rozwiązania.
6. Podsumowanie pracy - wnioski.

IV. Uwagi dotyczące pracy: praca we współpracy ze studentem Marcinem Ślusarczykiem (nr albumu: 91348).

V. Termin oddania pracy: zgodnie z Regulaminem Studiów.

VI. Konsultant: Praca nie wymaga konsultanta.

Opiekun merytoryczny

dr hab. inż. Stanisław Deniziak, prof. PŚk

DZIEKAN
Wydzięlu Elektrotechniki, Automatyki i Informatyki

.....
dr inż. Roman Stanisław Deniziak, prof. PŚk
(podpis)

Promotor pracy dyplomowej

dr inż. Tomasz Kaczmarek

T. Kaczmarek
(podpis)

Temat pracy dyplomowej celem jej wykonania otrzymałem(am):

Kielce, dnia 23.10.2023 r.

Maciej Bandura
czytelny podpis studenta

POLITECHNIKA ŚWIĘTOKRZYSKA
WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI I INFORMATYKI

Studia stacjonarne

Kierunek: Informatyka

Zatwierdzam:
DZIEKAN
ds. Studenckich i Dydaktyki
Wydziału Elektrotechniki, Automatyki i Informatyki

.....
n/a
Przedsięwzor ds. studenckich i dydaktyki

Rok akademicki: 2023/24

ZADANIE NA PRACĘ DYPLOMOWĄ
STUDIÓW PIERWSZEGO STOPNIA

Wydano studentowi:

Marcin Ślusarczyk

nr albumu: 91348

I. Temat pracy:

Programowa Sieć Dostarczania Zawartości
Software Content Delivery Network

II. Cel pracy:

Celem pracy jest projekt i implementacja serwerów skrajnych (ang. Edge servers) do świadczenia usługi CDN na dostępnej komercyjnie infrastrukturze serwerowej hostingującej strony internetowe ogólnego przeznaczenia. Przykładowe funkcjonalności: tunelowanie zapytań o zawartość do serwera klienta, lokalne zapamiętywanie zawartości, zarządzanie zapisanymi informacjami, zamiana ścieżek dostępu w dostarczanych dokumentach na adres skrajnego serwera, zapewnienie interfejsu komunikacyjnego ze źródłowym serwerem. Po integracji z projektem stworzonym przez studenta Macieja Bandurę powstanie kompletne rozwiązanie problemu budżetowego świadczenia usługi CDN w sposób programowy.

III. Plan pracy (zakres pracy):

1. Zapoznanie się z dziedziną, w szczególności z zagadnieniami dotyczącymi usługi CDN.
2. Opracowanie projektu Programowej Sieci Dostarczania Zawartości w zakresie serwerów skrajnych.
3. Dobór języka programowania, systemu bazodanowego oraz innych narzędzi programistycznych.
4. Implementacja oraz wdrożenie serwerów skrajnych zgodnie z celem pracy.
5. Testy opracowanego rozwiązania.
6. Podsumowanie pracy - wnioski.

IV. Uwagi dotyczące pracy: praca we współpracy ze studentem Maciejem Bandurą (nr albumu: 91234).

V. Termin oddania pracy: zgodnie z Regulaminem Studiów.

VI. Konsultant: Praca nie wymaga konsultanta.

Opiekun merytoryczny

dr hab. Inż. Stanisław Deniziak, prof. PŚk

DZIEKAN
Wydziału Elektrotechniki, Automatyki i Informatyki

.....
dr hab. inż. Stanisław Deniziak, prof. PŚk
(podpis)

Promotor pracy dyplomowej

dr inż. Tomasz Kaczmarek

T. Kaczmarek
(podpis)

Temat pracy dyplomowej celem jej wykonania otrzymałem(am):

Kielce, dnia ...23.10.2023...r.

czytelny podpis studenta

Marcin Ślusarczyk

Kielce, dnia 15.01.2024

Maciej Bandura, 91234

Imię i nazwisko studenta, nr albumu

26-021, Daleszyce, ul. Kilińskiego 59

Adres zamieszkania

Studia stacjonarne, pierwszego stopnia

Studia pierwszego/drugiego* stopnia, forma studiów stacjonarne/niestacjonarne*

Informatyka, Systemy Informacyjne

Kierunek, zakres

dr.inż. Tomasz Kaczmarek

Promotor pracy dyplomowej

OŚWIADCZENIE

Przedkładając w roku akademickim 20.23./24. promotorowi pracy dyplomowej studiów pierwszego/drugiego* stopnia, powołanemu przez Dziekana Wydziału Elektrotechniki, Automatyki i Informatyki

Politechniki Świętokrzyskiej, pracę dyplomową pod tytułem: Programowa Sieć Dostarczania Zawartości

oświadczam, że:

- 1) przedstawiona praca dyplomowa została opracowana przeze mnie samodzielnie, stosownie do wskazówek merytorycznych opiekuna pracy,
- 2) przy wykonywaniu pracy dyplomowej wykorzystano materiały źródłowe, w granicach dozwolonego użytku wymieniając autora, tytuł pozycji i miejsce jej publikacji,
- 3) praca dyplomowa nie zawiera żadnych danych, informacji i materiałów, których publikacja nie jest prawnie dozwolona,
- 4) przedstawiona praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu zawodowego/stopnia naukowego w wyższej uczelni,
- 5) niniejsza wersja pracy jest identyczna z treścią elektroniczną w systemie Archiwum Prac Dyplomowych.

Przyjmuję do wiadomości, że w przypadku ujawnienia w mojej pracy dyplomowej, stanowiącej podstawę nadania tytułu zawodowego, przypisania sobie przeze mnie autorstwa istotnego fragmentu lub innych elementów cudzego utworu lub ustalenia naukowego, rektor, w drodze decyzji administracyjnej, stwierdzi nieważność dyplomu.

Zostałem uprzedzony:

- 1) o odpowiedzialności karnej wynikającej z art. 115 ust. 1 i 2 ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (t. j. Dz. U. z 2022 r. poz. 2509 ze zm.): „Kto przywłaszcza sobie autorstwo albo wprowadza w błąd co do autorstwa całości lub części cudzego utworu albo artystycznego wykonania, podlega grzywnie, karze ograniczenia wolności albo pozbawienia wolności do lat 3. Tej samej karze podlega, kto rozpowszechnia bez podania nazwiska lub pseudonimu twórcy cudzy utwór w wersji oryginalnej albo w postaci opracowania, artystyczne wykonanie albo publicznie zniekształca taki utwór, artystyczne wykonanie, fonogram, videogram lub nadanie.”,
- 2) o odpowiedzialności dyscyplinarnej na podstawie art. 307 ust. 1 ustawy z dnia 20 lipca 2018 r. Prawo o szkolnictwie wyższym i nauce (t. j. Dz. U. z 2022 r. poz. 574, ze zm.): „Student podlega odpowiedzialności dyscyplinarnej za naruszenie przepisów obowiązujących w uczelni oraz za czyn uchybiający godności studenta.”

Maciej Bandura
Czytelny podpis studenta

*niepotrzebne skreślić

Kielce, dnia 15.01.2024

.....
Marcin Ślusarczyk

Imię i nazwisko studenta, nr albumu
25-366 Kielce, ul. Prosta 290/2A/2

Adres zamieszkania
Studia stacjonarne, pierwszego stopnia

Studia pierwszego/drużego* stopnia, forma studiów stacjonarne/niestacjonarne*
Informatyka, Systemy Informacyjne

Kierunek, zakres
dr inż. Tomasz Kaczmarek

Promotor pracy dyplomowej

OSWIADCZENIE

Przedkładając w roku akademickim 20.23/24. promotorowi pracy dyplomowej studiów pierwszego/drużego* stopnia, powołanemu przez Dziekana Wydziału **Elektrotechniki, Automatyki i Informatyki** Politechniki Świętokrzyskiej, pracę dyplomową pod tytułem: **Programowa Sieć Dostarczania Zawartości**.

.....
oświadczam, że:

- 1) przedstawiona praca dyplomowa została opracowana przeze mnie samodzielnie, stosownie do wskazówek merytorycznych opiekuna pracy,
- 2) przy wykonywaniu pracy dyplomowej wykorzystano materiały źródłowe, w granicach dozwolonego użytku wymieniając autora, tytuł pozycji i miejsce jej publikacji,
- 3) praca dyplomowa nie zawiera żadnych danych, informacji i materiałów, których publikacja nie jest prawnie dozwolona,
- 4) przedstawiona praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu zawodowego/stopnia naukowego w wyższej uczelni,
- 5) niniejsza wersja pracy jest identyczna z treścią elektroniczną w systemie Archiwum Prac Dyplomowych.

Przyjmuję do wiadomości, że w przypadku ujawnienia w mojej pracy dyplomowej, stanowiącej podstawę nadania tytułu zawodowego, przypisania sobie przeze mnie autorstwa istotnego fragmentu lub innych elementów cudzego utworu lub ustalenia naukowego, rektor, w drodze decyzji administracyjnej, stwierdzi nieważność dyplomu.

Zostałem uprzedzony:

- 1) o odpowiedzialności karnej wynikającej z art. 115 ust. 1 i 2 ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (t. j. Dz. U. z 2022 r. poz. 2509 ze zm.): „Kto przywłaszcza sobie autorstwo albo wprowadza w błąd co do autorstwa całości lub części cudzego utworu albo artystycznego wykonania, podlega grzywnie, karze ograniczenia wolności albo pozbawienia wolności do lat 3. Tej samej karze podlega, kto rozpowszechnia bez podania nazwiska lub pseudonimu twórcy cudzy utwór w wersji oryginalnej albo w postaci opracowania, artystyczne wykonanie albo publicznie zniekształca taki utwór, artystyczne wykonanie, fonogram, videogram lub nadanie.”,
- 2) o odpowiedzialności dyscyplinarnej na podstawie art. 307 ust. 1 ustawy z dnia 20 lipca 2018 r. Prawo o szkolnictwie wyższym i nauce (t. j. Dz. U. z 2022 r. poz. 574, ze zm.): „Student podlega odpowiedzialności dyscyplinarnej za naruszenie przepisów obowiązujących w uczelni oraz za czyn uchybiający godność studenta.”

Marcin Ślusarczyk
Czytelny podpis studenta

*niepotrzebne skreślić

Kielce, dnia 15.01.2024.....

Maciej Bandura
Imię i nazwisko studenta, nr albumu
26-021 Daleszyce, ul. Kilińskiego 59
Adres zamieszkania
Studia stacjonarne, pierwszego stopnia
Studia pierwszego/drużego stopnia, forma studiów stacjonarne/niestacjonarne
Informatyka, Systemy Informacyjne
Kierunek, zakres
dr inż. Tomasz Kaczmarek
Promotor pracy dyplomowej

OŚWIADCZENIE AUTORA PRACY

Zgodnie z ustawą z dnia 4 lutego 1994r. o prawie autorskim i prawach pokrewnych (t.j. Dz. U. 2022 poz. 2509), wyrażam zgodę na udostępnianie mojej pracy dyplomowej dla celów naukowych i dydaktycznych.

Maciej Bandura
Czytelny podpis studenta

Kielce, dnia 15.01.2024

Marcin Ślusarczyk, 91348

Imię i nazwisko studenta, nr albumu

25-366 Kielce, ul. Prosta 290/2A/2

Adres zamieszkania

Studia stacjonarne, pierwszego stopnia

Studia pierwszego/drużego stopnia, forma studiów stacjonarne/niestacjonarne

Informatyka, Systemy Informacyjne

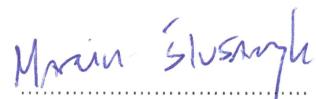
Kierunek, zakres

dr inż. Tomasz Kaczmarek

Promotor pracy dyplomowej

OŚWIADCZENIE AUTORA PRACY

Zgodnie z ustawą z dnia 4 lutego 1994r. o prawie autorskim i prawach pokrewnych (t.j. Dz. U. 2022 poz. 2509), wyrażam zgodę na udostępnianie mojej pracy dyplomowej dla celów naukowych i dydaktycznych.



Czytelny podpis studenta

Programowa Sieć Dostarczania Zawartości

Streszczenie

Celem niniejszej pracy inżynierskiej jest zaprojektowanie i wdrożenie *Programowej sieci dostępu do zawartości* (sCDN), wykorzystującej oferowaną komercyjnie infrastrukturę serwerową. Praca skupia się na opracowaniu dwóch głównych komponentów: oprogramowania aplikacyjnego serwera skrajnego, działającego w technologii PHP, które ma na celu efektywne świadczenie treści z serwerów źródłowych, oraz serwera centralnego, będącego serwisem internetowym do konfiguracji i zarządzania siecią sCDN z perspektywy administratora i klienta. Dodatkowo, zostanie stworzony sandbox do dystrybucji zapytań użytkowników na najbliższe serwery skrajne, zapewniając efektywną i szybką obsługę treści. Wszystkie te elementy będą ze sobą współpracować, tworząc spójną i funkcjonalną sieć sCDN.

Słowa kluczowe: Sieć dostępu do zawartości, Infrastruktura serwerowa, Świadczenie treści, System zarządzania, Tunelowanie zapytań, Obliczenia geolokalizacyjne

Software Content Delivery Network

Summary

The aim of this engineering thesis is to design and implement a software content delivery network (sCDN), utilizing commercially available server infrastructure. The work focuses on the development of two main components: the application software of the edge server, operating on PHP technology, which aims to efficiently deliver content from source servers, and the central server, which serves as an internet service for configuring and managing the sCDN network from both the administrator's and the client's perspective. Additionally, a sandbox will be created for distributing user queries to the nearest edge servers, ensuring efficient and rapid content delivery. All these elements will work together to form a cohesive and functional sCDN network.

Keywords: Content delivery network, Server infrastructure, Content delivery, Management system, Request tunneling, Geolocation Computing

Spis treści

1. Wstęp	19
2. Usługi CDN	22
3. Programowe podejście do świadczenia usług CDN	31
4. Projekt serwera centralnego (<i>Maciej Bandura</i>)	36
4.1. Pojęcie serwera centralnego	36
4.2. Proces dostępu do zawartości	37
4.3. Dobór serwera skrajnego	42
5. Projekt serwera skrajnego (<i>Marcin Ślusarczyk</i>)	44
5.1. Pojęcie serwera skrajnego	44
5.2. Ścieżki specjalne	46
5.3. Proces komunikacji	47
5.4. Mechanizm świadczenia treści	49
6. Implementacja serwera centralnego (<i>Maciej Bandura</i>)	52
6.1. Wykorzystywane technologie	52
6.2. Struktura aplikacji	55
6.3. Schemat bazy danych	57
6.4. Identyfikacja wizualna	61
6.5. Konfiguracja serwera	62
6.5.1. Application.properties	62
6.5.2. pom.xml	63
6.6. Funkcjonalności	66
6.6.1. Wymagania funkcjonalne	66
6.6.2. Wymagania niefunkcjonalne	67
6.6.3. Diagramy przypadków użycia	68
6.7. Prezentacja funkcjonalności	70
6.7.1. Logowanie i rejestracja	70
6.7.2. Rejestracja domeny	76
6.7.3. Konfiguracja domeny	78
6.7.4. Zarządzanie serwerami skrajnymi	82
6.7.5. Raportowanie błędów	85
6.7.6. Zarządzanie kontem użytkownika	89

6.7.7. Dostarczanie treści - sandbox	91
6.8. Komunikacja z serwerami skrajnymi	94
6.9. Mechanizm wysyłania maili	96
6.10. Obsługa systemu płatności	101
7. Implementacja serwera skrajnego (<i>Marcin Ślusarczyk</i>)	107
7.1. Wykorzystywane technologie	107
7.2. Struktura aplikacji	109
7.3. Schemat bazy danych	113
7.4. Funkcjonalności	115
7.4.1. Wymagania funkcjonalne	115
7.4.2. Wymagania niefunkcjonalne	116
7.5. Plik konfiguracyjny i .htaccess	117
7.6. Połączenie serwera z bazą danych	120
7.7. Challenge page	124
7.8. Moduły serwera	128
7.8.1. Inicjalizujący świadczenie treści (qc_CDN)	128
7.8.2. Zarządzający lokalną zawartością (qc_Cache)	131
7.8.3. Przekazywanie żądań (qc_Tun)	134
7.8.4. Modyfikacji treści (qc_Conv)	140
7.8.5. Bezpieczeństwa (qc_Security)	147
7.8.6. Obsługi błędów (qc_Error)	149
7.9. Interfejs programistyczny	151
7.9.1. Zarządzanie serwerem skrajnym	152
7.9.2. Zarządzanie domenami	152
7.9.3. Zarządzanie lokalną zawartością	153
7.10. Źródło dostarczanych treści	153
8. Podsumowanie - ujęcie całościowe	156
8.1. Efekt końcowy pracy	156
8.2. Testowanie wdrożonego rozwiązania	158
8.3. Możliwości rozwoju serwisu	163
Literatura	165
Spis rysunków	167
Spis kodów źródłowych	169

1. Wstęp

W ostatnich latach dziedzina sieci dostarczania treści (Content Delivery Network, CDN) staje się jednym z kluczowych elementów współczesnej infrastruktury internetowej. Wartość i znaczenie sieci CDN w dzisiejszym świecie cyfrowym jest trudne do przecenienia, a jej rozwój jest zarówno dynamiczny, jak i nieustannie ewoluujący. W obliczu tego wzrostu, projektowanie i implementacja efektywnych sieci CDN staje się zadaniem wyjątkowo złożonym i wymagającym.

W ramach niniejszej pracy inżynierskiej, podjęliśmy się zadania dogłębnego badania i opracowania nowatorskiego podejścia do budowy sieci CDN - programowej sieci CDN. Wymagało to nie tylko rozbudowanej wiedzy teoretycznej na temat działania takich sieci, ale również umiejętności praktycznej implementacji. Z jednej strony, serwer centralny ma być w pełni funkcjonalnym serwisem internetowym, który będzie pełnił kluczową rolę w zarządzaniu i dystrybucją treści w ramach sieci CDN. Z drugiej zaś strony, serwer skrajny, jako złożona aplikacja, będzie odpowiadać za efektywne dostarczanie treści do użytkowników końcowych. Oba te elementy, mimo iż różniące się funkcjonalnością i charakterem, muszą ze sobą współpracować w sposób spójny i efektywny.

Głównym celem niniejszej pracy inżynierskiej jest zaprojektowanie i implementacja oprogramowania do świadczenia usług Content Delivery Network (CDN) na istniejącej komercyjnej infrastrukturze serwerowej, która zazwyczaj jest wykorzystywana do hostowania stron internetowych o różnorodnym przeznaczeniu.

Idea opracowania niniejszej pracy inżynierskiej, a w szczególności projektowania alternatywnego rozwiązania sieci CDN, została zainspirowana przez kilka przypadków awarii, które miały miejsce u jednego z największych dostawców usług CDN - Cloudflare. Awaria z 2 lipca 2019 roku była szczególnie znacząca, ponieważ spowodowała utratę ruchu na poziomie aż 80% całego serwisu. Taki skokowy spadek dostępności treści nie tylko wywołał szerokie konsekwencje dla użytkowników i firm polegających na usługach Cloudflare, ale również ujawnił wrażliwość internetu na problemy związane z pojedynczymi, dużymi punktami dystrybucji treści. Inny incydent (*Cloudbleed*), który miał miejsce 18 lutego 2017 roku, dotyczył wycieków pamięci w integralnej części systemu firmy Cloudflare. W trakcie zaistnienia pewnych warunków, losowym użytkownikom były przekazywane dane wrażliwe innych użytkowników, w tym pliki cookies, tokeny sesji, i adresy IP. Ta sytuacja podniosła kwestie bezpieczeństwa da-

nych i prywatności, stanowiąc poważne zagrożenie dla ochrony informacji osobowych i firmowych. Zgodnie z informacjami dostarczonymi przez firmę Cloudflare, do takich wycieków doszło ponad 18 mln razy.

To ukazało jak internet jest zależny od właścicieli sieci CDN. Niestety, ale posiadanie tradycyjnej sieci wiąże się ze znaczącymi kosztami. Chęć minimalizacji tych kosztów skłoniła do implementacji sieci CDN na istniejącej infrastrukturze serweroowej.

Zawartość pracy:

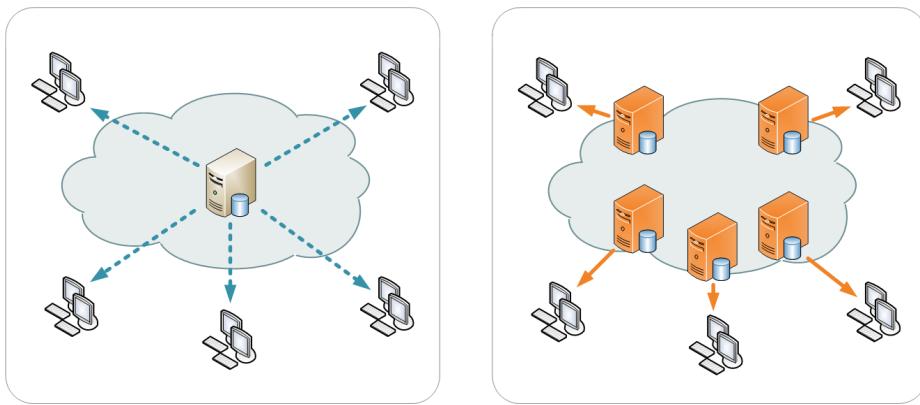
- Rozdział 2: **Usługi CDN**: w tym rozdziale zostanie przedstawiony szczegółowy opis zasad działania i struktury tradycyjnych sieci CDN. Omówione będą kluczowe aspekty, takie jak sposób dystrybucji treści, zarządzanie ruchem sieciowym, a także techniki optymalizacji wydajności. Ponadto, rozdział ten zawiera analizę dodatkowych funkcjonalności związanych z sieciami CDN, w tym zarządzania bezpieczeństwem, skalowalnością i elastycznością sieci.
- Rozdział 3: **Programowe podejście do świadczenia usług CDN**: ten rozdział skupia się na programowym podejściu do świadczenia usług CDN. Omówione zostaną konsekwencje tego podejścia, w tym możliwości i wyzwania, które niesie ze sobą programowe podejście do budowy sieci CDN. Przedstawione również będą różnice między klasycznymi a programowymi sieciami CDN, szczególnie pod kątem elastyczności, skalowalności i kosztów.
- Rozdziały 4-5: **Projekt serwera centralnego, Projekt serwera skrajnego**: w tych rozdziałach zostaną wyjaśnione podstawy teoretyczne stojące za działaniem serwera centralnego oraz serwerów skrajnych. Omówione również będą ich role, funkcjonalności, a także sposoby współpracy i interakcji w ramach sieci CDN.
- Rozdziały 6-7: **Implementacja serwera centralnego i serwerów skrajnych**: te rozdziały przedstawiają szczegółowy opis implementacji serwera centralnego i serwerów skrajnych. Zaprezentowane zostaną zrzuty ekranu interfejsu użytkownika, ilustrujące funkcjonalności i design systemu. Ponadto, w rozdziale zawarte będą fragmenty kodu źródłowego, które ukażą najważniejsze mechanizmy i algorytmy wykorzystane w projekcie. Te rozdziały stanowią praktyczne uzupełnienie teoretycznych podstawa przedstawionych w poprzednich częściach pracy.

2. Usługi CDN

Sieć dostarczania treści - Content Delivery Network (CDN) stanowi kluczowy element współczesnej architektury internetowej, rewolucjonizując sposób, w jaki treści są dostarczane online. Sieci CDN, będące rozbudowanymi systemami, mają na celu optymalizację transferu danych między serwerami źródłowymi a użytkownikami końcowymi. CDN to nic innego jak grupa serwerów, które są strategicznie rozmieszczone na całym świecie, z treścią replikowaną i przechowywaną dla szybkiego dostępu. Głównym celem CDN jest zapewnienie szybkiego dostarczania treści do użytkowników końcowych, minimalizując opóźnienia.

Fundamentalną zasadą funkcjonowania sieci CDN jest dystrybucja kopii treści na serwerach skrajnych, umieszczonych w różnych lokalizacjach geograficznych. W odpowiedzi na żądania użytkowników, system CDN automatycznie kieruje je do najbliższego serwera skrajnego, minimalizując czas odpowiedzi. Treść jest dostarczana szybciej, ponieważ jest mniej "skoków" między serwerami, co oznacza mniej opóźnień i szybsze czasy ładowania.

Mechanizm dostarczania treści bez wykorzystania CDN skupia się na centralnym serwerze źródłowym, z którego treść jest kierowana bezpośrednio do użytkownika. W takim podejściu każde żądanie o treść musi przejść przez całą trasę od centralnego serwera do urządzenia końcowego użytkownika. Każdy taki "skok" dodaje pewne opóźnienie do czasu odpowiedzi, co oznacza, że im dalej serwer jest od klienta, tym dłuższy jest czas ładowania strony. Warto zauważyc, że redukcja "przejść" między serwerami w kontekście sieci CDN ma szczególne znaczenie dla treści statycznych, takich jak strony internetowe, multimedia czy aplikacje internetowe. Dzięki lokalnym kopiom treści, nawet użytkownicy odlegli od centralnego źródła doświadczają szybkiego dostępu, co przekłada się na efektywność i wydajność dostarczanych usług online. Ostatecznym efektem jest poprawa globalnego doświadczenia użytkownika poprzez skrócenie czasów ładowania i eliminację opóźnień wynikających z długich tras dostarczania treści.



Rysunek 2.1. Konwencjonalny model dostępu do zawartości a sieć CDN.

Usługi CDN stanowią nie tylko środek do szybkiego dostarczania treści, ale także kompleksowe platformy oferujące zaawansowane funkcje wspierające wydajność, bezpieczeństwo i optymalizację witryn internetowych. W dzisiejszym środowisku online, ataki typu DDoS stanowią zagrożenie dla dostępności witryn. Wiodące firmy CDN implementują zaawansowane mechanizmy obronne, wykorzystując swoją rozproszoną infrastrukturę, aby zminimalizować wpływ takich ataków. Kolejnym kluczowym elementem bezpiecznego przesyłania danych jest świadczenie usług certyfikatów SSL/TLS. Firmy z branży CDN nie tylko oferują szyfrowanie komunikacji, ale także łatwe zarządzanie certyfikatami, co przekłada się na zwiększone bezpieczeństwo witryny.

Kolejną funkcjonalnością, o której należy wspomnieć, jest automatyczna optymalizacja obrazów i innych zasobów statycznych, co pozytywnie wpływa na szybkość ładowania witryny. Ta funkcja obejmuje proces dostosowywania i zoptymalizowania rozmiarów plików graficznych oraz innych statycznych elementów, takich jak arkusze stylów czy skrypty JavaScript. Działanie tego mechanizmu przekłada się na znaczne przyspieszenie czasu ładowania strony internetowej. Redukcja rozmiarów plików pozwala na efektywniejsze przesyłanie danych z serwera CDN do użytkownika końcowego. Dodatkowo, korzyści z automatycznej optymalizacji nie ograniczają się jedynie do aspektu praktycznego. Szybsze ładowanie witryny ma istotne znaczenie dla jej pozycji w wynikach wyszukiwarek. Algorytmy wyszukiwarek, takie jak te stosowane przez Google, biorą pod uwagę czas ładowania strony jako jeden z czynników rankinguowych. Strony o szybkim czasie ładowania mogą zyskać lepszą pozycję w wynikach wyszukiwania, co z kolei zwiększa ich widoczność i dostępność dla użytkowników. W rezultacie, automatyczna optymalizacja zasobów statycznych nie tylko przyczynia się do efektywności technicznej witryny, ale również kształtuje jej postrzeganie przez użytkowników i wyszukiwarki, co stanowi kluczowy element strategii online.

Zaawansowane CDN wprowadzają nowoczesne podejście do optymalizacji działania witryn internetowych poprzez wsparcie dla rozproszonego przetwarzania, znane jako "edge computing". To zaawansowane rozwiązanie polega na umożliwieniu wykonywania małych aplikacji i skryptów bezpośrednio na serwerach skrajnych, znajdujących się blisko użytkowników końcowych. Rozproszone przetwarzanie eliminuje konieczność przesyłania danych do centralnego serwera źródłowego w celu przetworzenia. To oznacza, że małe aplikacje, funkcje czy skrypty mogą być uruchamiane bliżej użytkownika, co przekłada się na zminimalizowanie opóźnień i szybsze dostarczanie treści. Przykładowe zastosowania tego podejścia obejmują dynamiczne renderowanie treści, personalizację zawartości na podstawie lokalizacji czy preferencji użytkownika, a także obsługę małych interaktywnych elementów na stronie.

W dzisiejszym środowisku online, skuteczne zarządzanie witryną wymaga nie tylko szybkiego dostarczania treści, ale również precyzyjnej analizy jej wydajności. Usługi monitorowania i analizy ruchu - integralne elementy zaawansowanych CDN, stanowią kluczowe narzędzie dla administratorów systemów, dostarczając cenne dane niezbędne do ciągłej optymalizacji działania witryny i dostosowania jej do zmieniających się potrzeb użytkowników. Usługi te śledzą czasy ładowania poszczególnych elementów witryny, identyfikując potencjalne obszary o największym opóźnieniu. To pozwala administratorom zlokalizować i zidentyfikować potencjalne problemy z wydajnością, takie jak wolno ładowane zasoby czy błędy w renderowaniu strony. Rozbudowana analiza ruchu pozwala na dostosowywanie infrastruktury CDN do aktualnych potrzeb. Administracja może skalować zasoby CDN w zależności od obciążenia, zapewniając optymalne dostarczanie treści nawet w sytuacjach wzmożonego ruchu. Dane monitorowania pomagają w zrozumieniu, które elementy witryny mogą być skutecznie cachowane, a które wymagają częstego odświeżania. Dzięki temu, administratorzy mogą dostosowywać ustawienia cache'owania, eliminując niepotrzebne zapytania do serwera źródłowego. Prześledzenie ruchu po stronie witryny umożliwia zrozumienie, które sekcje są najczęściej odwiedzane, a także jak użytkownicy poruszają się między różnymi stronami. To kluczowe dane dla projektantów UX/UI, umożliwiające optymalizację nawigacji i układu witryny. Nazywane jest to mapowaniem ruchu. Szczegółowa analiza użytkowników umożliwia administratorom gromadzenie danych, takich jak lokalizacja geograficzna, używane urządzenia czy przeglądarki. Dzięki temu mogą oni dostosowywać witrynę do różnych grup odbiorców, zapewniając optymalne doświadczenia użytkownika w zależności od ich potrzeb i kontekstu. Monitorowanie ruchu pozwala również na bieżące identyfikowanie i śledzenie błędów,

takich jak błędy ładowania zasobów czy błędy zapytań HTTP. Szybka reakcja na te problemy jest kluczowa dla utrzymania wysokiej dostępności witryny. Dane monitorowania mogą ujawnić obszary w kodzie źródłowym, które generują duże obciążenie lub są przyczyną błędów. Administratorzy mogą wprowadzać zmiany w kodzie, eliminując potencjalne bottlenecki i zwiększaając ogólną wydajność witryny.



Rysunek 2.2. Jak działa CDN.¹

Kolejnym ważnym aspektem, który również jest powiązany z usługami CDN, jest zagrożenie dotyczące bezpieczeństwa witryn internetowych. Stanowią one poważne wyzwanie dla administratorów systemów. Aby zabezpieczyć witrynę przed potencjalnymi atakami aplikacyjnymi, kluczowym elementem staje się implementacja zintegrowanych rozwiązań typu **Web Application Firewall** (WAF). WAF to specjalizowane narzędzie, które dostarcza dodatkową warstwę obrony, chroniącą witrynę przed różnorodnymi atakami i zabezpieczającą integralność oraz poufność danych. Korzyści, jakie niesie ze sobą korzystanie z WAF, to między innymi zintegrowane rozwiązania do identyfikacji i blokowania różnorodnych ataków aplikacyjnych, takich jak ataki *SQL Injection*, *Cross-Site Scripting* (XSS) czy *Cross-Site Request Forgery* (CSRF). Ataki wstrzykiwania kodu, takie jak SQL Injection, stanowią powszechnie zagrożenie dla witryn internetowych. Rozwiązania WAF skutecznie wykrywają i blokują próby wstrzykiwania złośliwego kodu, chroniąc wrażliwe dane i zabezpieczając

¹What is a CDN by Cloudflare: www.cloudflare.com/learning/cdn/what-is-a-cdn/

integralność bazy danych. Dzięki temu, nawet w przypadku nowoczesnych i zaawansowanych ataków, witryna pozostaje chroniona przed potencjalnymi zagrożeniami. WAF umożliwia filtrowanie treści przechodzących przez witrynę, eliminując potencjalnie niebezpieczne elementy. To obejmuje analizę zarówno danych wejściowych, jak i wyjściowych, zapewniając, że żadne złośliwe treści nie przedostaną się do witryny, ani nie zostaną dostarczone do użytkowników. Zintegrowane rozwiązania WAF oferują funkcje monitorowania ruchu w czasie rzeczywistym. Dzięki temu administratorzy mogą błyskawicznie reagować na wykryte zagrożenia, podejmując natychmiastowe kroki w celu zablokowania potencjalnie szkodliwych działań. Zastosowanie zintegrowanych rozwiązań WAF staje się nieodzowne w kontekście zapewnienia kompleksowej ochrony witryny przed atakami aplikacyjnymi. Te zaawansowane narzędzia nie tylko zabezpieczają przed znymi zagrożeniami, ale także dostarczają elastyczność i adaptacyjność, niezbędną do skutecznego przeciwdziałania nowym, ewoluującym formom cyberzagrożeń.

Funkcjonalnością, o której należy wspomnieć w kontekście sieci CDN, jest **Global Load Balancing** (GLB), stanowiąca integralną część sieci dostarczania zawartości. W skrócie, GLB działa poprzez automatyczne kierowanie użytkowników do najbliższego dostępnego serwera, co znacząco skraca czas odpowiedzi witryn internetowych. Mechanizm ten jest szczególnie istotny w kontekście globalnych sieci dostarczania zawartości, gdzie użytkownicy mogą pochodzić z różnych regionów geograficznych. GLB uwzględnia szereg czynników przy podejmowaniu decyzji dotyczących kierowania ruchu użytkowników. Oprócz wspomnianej już lokalizacji, brane są też pod uwagę:

- **Obciążenie serwerów:** GLB monitoruje bieżące obciążenie poszczególnych serwerów w czasie rzeczywistym. Informacje te obejmują takie parametry jak zużycie CPU, ilość dostępnej pamięci RAM czy przepustowość sieci.
- **Dostępność i stabilność serwerów:** GLB uwzględnia dostępność poszczególnych serwerów, eliminując z ruchu te, które są chwilowo niedostępne lub doświadczają awarii.
- **Priorytetyzacja usług:** W zależności od wymagań projektu, GLB może być skonfigurowane do priorytetyzacji określonych usług lub rodzajów treści.
- **Ruch i trendy użytkowników:** GLB analizuje bieżące trendy ruchu użytkowników, identyfikując wzorce i dostosowując się do zmian w zachowaniach użytkowników. Działa to na zasadzie adaptacji do zmieniających się warunków, co pozwala zoptymalizować ruch online.

Bez wątpienia, znaczenie usług CDN dla współczesnego internetu jest nieocenione. Bez ich obecności na popularnych stronach internetowych, skala ruchu byłaby na tyle ogromna, że mogłaby prowadzić do krytycznych awarii serwerów. CDN odgrywają kluczową rolę w równoważeniu i dystrybucji tego potężnego obciążenia, co ma istotne konsekwencje dla niezawodności, wydajności i dostępności całego systemu. Dzięki zastosowaniu CDN i rozproszeniu treści na wielu serwerach skrajnych, obciążenie na serwerach źródłowych zostaje znacznie zredukowane. To nie tylko zmniejsza presję na centralnych serwerach, ale również skutkuje znacznym wzrostem ogólnej wydajności systemu. Mechanizm dystrybucji treści przez sieci CDN umożliwia obsługę jednoczesnych żądań wielu użytkowników, minimalizując czas odpowiedzi i eliminując ryzyko przeciążenia pojedynczych punktów dostępu. Ponadto, sieci CDN efektywnie zarządzają transferem danych. Minimalizują zużycie przepustowości sieciowej poprzez lokalne dostarczanie treści. Ważnym aspektem jest również redundancja treści. Sieci CDN nie tylko przechowują kopie treści na wielu serwerach, ale także automatycznie przekierowują ruch w przypadku awarii jednego z serwerów. Dzięki temu, nawet w sytuacjach krytycznych, gdy jeden serwer ulegnie awarii, użytkownicy nadal mogą uzyskiwać dostęp do treści, co znacząco podnosi ogólną dostępność systemu.

Niemniej jednak, mimo licznych korzyści, implementacja sieci CDN niesie ze sobą pewne wyzwania, które wymagają uwagi i ostrożności. Jednym z kluczowych aspektów, który może stanowić barierę, są koszty związane z wprowadzeniem i utrzymaniem takiej infrastruktury. Szczególnie dla mniejszych przedsiębiorstw, nakłady finansowe związane z CDN mogą być uznane za znaczące obciążenie budżetowe. Dodatkowo, skomplikowane procesy konfiguracji i zarządzania siecią CDN mogą stanowić wyzwanie, zwłaszcza dla tych firm, które nie dysponują specjalistyczną wiedzą w tym obszarze. Wymaga to zatrudnienia ekspertów ds. CDN, co może generować dodatkowe koszty i utrudnić proces wdrożenia. Kwestie bezpieczeństwa stanowią kolejne wyzwanie. Pomimo zaawansowanych mechanizmów ochronnych, żadna infrastruktura nie jest całkowicie wolna od ryzyka ataków. Bezpieczeństwo sieci CDN wymaga stałego monitorowania, aktualizacji zabezpieczeń i reakcji na nowe zagrożenia, co może być trudne do utrzymania. Warto również zauważać, że niektóre CDN mogą nie posiadać serwerów w pewnych lokalizacjach geograficznych. To oznacza, że użytkownicy w tych konkretnych obszarach mogą nie doświadczać poprawy wydajności, co wpływa na spójność świadczonych usług dla wszystkich użytkowników na globalną skalę. Również, złożoność zarządzania kontentem w sieciach CDN wymaga zaawansowanych narzędzi i umiejętności, co stanowi dodatkowe wyzwanie dla zarządzających treścią.

Firmy CDN stale rozwijają swoje oferty, aby sprostać dynamicznym wymaganiom współczesnego środowiska internetowego. Oferują kompleksowe rozwiązania, które nie tylko przyspieszają dostarczanie treści, ale także chronią witryny i optymalizują ich działanie. To kluczowy aspekt skutecznej strategii online, umożliwiający firmom osiągnięcie sukcesu w dzisiejszym konkurencyjnym środowisku.

Największe firmy świadczące usługi CDN to potężne przedsiębiorstwa, które odgrywają kluczową rolę w optymalizacji dostarczania treści online. Nie jest łatwo jednoznacznie określić, która firma jest największym dostawcą usług CDN, ponieważ wiele czynników wpływa na tę kwestię, takich jak liczba serwerów, przepustowość, liczba klientów, zasięg geograficzny i wiele innych. Jednak według niektórych źródeł, jednym z największych dostawców usług CDN jest **Akamai Technologies**. Akamai jest jednym z pionierów w dziedzinie CDN i obsługuje wiele dużych firm internetowych. Została założona w 1998 roku i szybko stała się jednym z liderów branży CDN. Firma ma siedzibę główną w Cambridge, Massachusetts, USA. Sieć serwerów skrajnych Akamai jest rozproszona na całym świecie, obejmując liczne lokalizacje geograficzne. Dzięki temu globalnemu zasięgowi, Akamai jest w stanie dostarczać treści szybko i efektywnie do użytkowników na różnych kontynentach. Liczba serwerów Akamai jest dynamicznie dostosowywana w zależności od potrzeb i wymagań ruchu online. Według informacji dostępnych na stronie firmy, sieć Akamai obejmuje ponad 4100 lokalizacji w 131 krajach². Firma nie publikuje jednak szczegółowych informacji na temat rozkładu swoich serwerów. W trzecim kwartale 2023 roku, roczne przychody Akamai wyniosły 965 milionów dolarów, co stanowiło wzrost o 9% w porównaniu do poprzedniego roku.³ Firma zatrudnia ponad 9,800 pracowników na całym świecie. Podobnie jak Akamai, **Cloudflare** jest również czołowym graczem w dziedzinie usług sieciowych, oferującym swoje własne unikalne rozwiązania w zakresie bezpieczeństwa i wydajności. Założona w 2009 roku, firma ma siedzibę w San Francisco w Kalifornii, i szybko zdobyła pozycję jednego z czołowych dostawców usług CDN na rynku. Cloudflare oferuje szereg usług, obejmujących: dostarczanie treści, bezpieczeństwo internetowe, rozwiązania DNS oraz rozbudowaną usługę VPN⁴.

²Komunikat prasowy firmy Akamai: <https://www.akamai.com/newsroom/press-release/akamai-expands-world-s-most-distributed-cloud-network-with-new-c>

³Komunikat prasowy firmy Akamai: <https://www.akamai.com/newsroom/press-release/akamai-reports-third-quarter-2023-financial-results>

⁴Produkty firmy Cloudflare: <https://www.cloudflare.com/application-services/products/>

Warto podkreślić, że Cloudflare udostępnia również darmową wersję swojego CDN, co sprawia, że jest dostępna dla szerokiego spektrum użytkowników, w tym dla małych i średnich przedsiębiorstw. Firma ta jest ceniona za innowacyjne podejście do usług CDN, skupiając się zarówno na efektywnym dostarczaniu treści, jak i na zapewnianiu bezpieczeństwa online. Sieć Cloudflare obejmuje ponad 310 miast w ponad 120 krajach, działając w odległości około 50 milisekund od około 95% globalnej populacji podłączonej do Internetu⁵. Firma stale rozwija swoją sieć, dodając nowe lokalizacje, co zwiększa bezpieczeństwo, wydajność i niezawodność dużych części Internetu.

Cloudflare zyskało ogromną popularność jako jedno z najczęściej wybieranych CDN na świecie. Z usług Cloudflare korzysta 4,1 miliona klientów, w tym imponująca liczba około 120 tysięcy płatnych klientów, generujących roczne przychody przekraczające 100 milionów dolarów. Należy zauważyć, że według informacji ze strony internetowej Cloudflare, ponad 25 milionów stron internetowych korzysta z ich usług, i ta liczba stale rośnie. Cloudflare stanowi solidną podstawę dla prawie 20% z pierwszych 1000 najpopularniejszych witryn internetowych i obsługuje imponującą liczbę ponad 25 milionów żądań HTTP na sekundę⁶. Obecnie, Cloudflare obsługuje ponad 14 milionów aktywnych stron internetowych⁷, co świadczy o ich znacznej roli w dziedzinie dostarczania treści online.

Sieci dostępu do zawartości zrewolucjonizowały dostarczanie treści, umożliwiając szybki dostęp poprzez globalne serwery skrajne. Ich głównym celem jest skrócenie czasu odpowiedzi i zapewnienie efektywnego dostępu do treści poprzez lokalne kopie. Korzyści to nie tylko natychmiastowe dostarczanie treści, ale także optymalizacja transferu danych, poprawa bezpieczeństwa w sieci, certyfikaty SSL/TLS i automatyczna optymalizacja zasobów. Usługi CDN oferują również monitorowanie ruchu, globalne równoważenie obciążenia, zabezpieczanie aplikacji webowych (WAF) i mapowanie ruchu. Te funkcje nie tylko efektywnie dostarczają treści, ale także umożliwiają skuteczne zarządzanie ruchem, zwiększając wydajność i bezpieczeństwo aplikacji online.

Mimo pozytywnego wpływu na dostępność i wydajność systemów online, implementacja i utrzymanie infrastruktury CDN mogą być kosztowne, a skomplikowane procesy zarządzania wymagają znajomości specyfiki tej dziedziny.

⁵Blog firmy Cloudflare: <https://blog.cloudflare.com/cloudflare-connected-in-over-300-cities/>

⁶Statystyki cloudflare: <https://www.wpoven.com/blog/cloudflare-market-share/>

⁷Statystyki cloudflare: <https://firstsiteguide.com/cloudflare-stats/>

3. Programowe podejście do świadczenia usług CDN

Utrzymanie własnej infrastruktury serwerowej to zadanie pełne wyzwań, związane z różnorodnymi aspektami technicznymi, finansowymi i operacyjnymi. Przede wszystkim, proces zakupowy i wdrażanie własnych serwerów wymaga skomplikowanego planowania, budżetowania oraz negocjacji z dostawcami sprzętu. To czasochłonny proces, który może generować dodatkowe koszty związane z wyborem odpowiednich dostawców i dostosowywaniem infrastruktury do specyficznych potrzeb firmy i świadczonych usług. Koszty inwestycyjne stanowią kolejne wyzwanie, zwłaszcza dla małych i średnich firm, które muszą zmierzyć się z znaczącymi nakładami finansowymi na zakup, instalację i konfigurację sprzętu. Konieczność przewidywania ruchu jest kolejnym aspektem, który wymaga precyzyjnych prognoz. Błędne oszacowania mogą prowadzić do nadmiernego lub niewystarczającego rozmiaru infrastruktury, wpływając negatywnie na efektywne dostarczanie treści. Zatrudnienie specjalistycznego personelu technicznego to następny istotny element utrzymania własnej infrastruktury. Personel musi być kompetentny w obszarze zarządzania serwerami, konserwacji sprzętu, monitorowania wydajności i reagowania na ewentualne awarie. Ponoszenie kosztów związanych z wynagrodzeniem i szkoleniem takiego personelu stanowi dodatkowe wyzwanie.

Owa złożoność i ryzyko związane z utrzymaniem własnej infrastruktury serwerowej stawiają przed firmami poważne dylematy. Warto zatem przyjrzeć się z bliska programowemu podejściu do świadczenia usług CDN jako innowacyjnemu rozwiązaniu, które eliminuje te trudności, umożliwia bardziej elastyczne dostosowanie się do zmieniających się warunków i otwiera nowe perspektywy efektywnego dostarczania treści online. W ramach tego rozwiązania, zamiast korzystać z własnej infrastruktury serwerowej, sCDN (software CDN - programowy CDN) bazuje na istniejącej popularnej infrastrukturze serwerów hostingowych obsługujących język PHP. Główną różnicą jest sposób zarządzania ruchem, gdzie zamiast wielu serwerów DNS kierujących użytkowników do różnych serwerów skrajnych, wykorzystuje się jednolity, centralizowany serwer główny - **serwer centralny**. Warto podkreślić, że kluczowym elementem funkcjonowania programowej sieci CDN jest inteligentne wykorzystanie danych geolokalizacyjnych. Dzięki tym danym serwer centralny jest w stanie precyzyjnie kierować użytkownika do najbardziej optymalnego serwera skrajnego.

Infrastruktura serwerów obsługujących język PHP jest niezwykle rozległa i zróżnicowana. PHP jest jednym z najpopularniejszych języków programowania do tworzenia stron internetowych i aplikacji webowych, co przyczyniło się do ogromnej popularności serwerów hostingowych obsługujących ten język. Na całym świecie istnieje setki tysięcy serwerów hostingowych, które obsługują PHP. Ich liczba stale rośnie w odpowiedzi na rosnące zapotrzebowanie na strony internetowe i aplikacje webowe. To ogromna liczba, która potwierdza dominację PHP w dziedzinie web developmentu. Według danych z grudnia 2023 roku, PHP jest używany przez 76,6% wszystkich stron internetowych, których język programowania po stronie serwera jest nam znany⁸.



Rysunek 3.1. Użycie języków programowania po stronie serwera⁹

Po pierwsze, popularność PHP wynika z jego wszechstronności i łatwości użycia, co sprawia, że jest preferowanym wyborem dla wielu programistów i firm. Język ten jest doskonale zintegrowany z wieloma bazami danych i systemami operacyjnymi, co umożliwia elastyczne dostosowanie się do różnorodnych potrzeb projektów internetowych. Dodatkowo, serwery hostingowe obsługujące PHP są najbardziej rozpowszechnione na świecie. Ich powszechność oznacza, że są one łatwo dostępne i znajdują się w wielu lokalizacjach geograficznych. To kluczowy element skuteczności sCDN, ponieważ umożliwia ona użytkownikom dostęp do treści z lokalnych serwerów skrajnych, co skraca czas odpowiedzi i poprawia ogólną wydajność dostarczania treści.

⁸Statystyki użycia PHP: w3techs.com/technologies/details/pl-php

⁹Ankiety technologii w sieci: w3techs.com/technologies/overview/programming_language

Takie rozwiązanie jest również zgodne z ideą efektywnego wykorzystania istniejących zasobów, eliminując konieczność tworzenia nowej infrastruktury. Wybór serwerów hostingowych obsługujących PHP do budowy programowej sieci CDN ukierunkowuje się więc na gotowe i sprawdzone rozwiązania, co znacznie ułatwia i przyspiesza proces implementacji sieci. Warto również zaznaczyć, że jednym z głównych atutów takiego podejścia do budowy sieci CDN jest to, że administratorzy nie muszą martwić się utrzymaniem i zarządzaniem serwerami skrajnymi. To firma hostingowa bierze na siebie odpowiedzialność za takie zadania jak monitorowanie, naprawy awarii i regularne aktualizacje sprzętu.

Programowe podejście do budowy sieci CDN niesie ze sobą kilka wad, które warto uwzględnić przy rozważaniu tego rozwiązania. Jest to między innymi:

- **Konieczność modyfikacji treści:** W praktyce oznacza to, że treści hostowane na serwerze źródłowym muszą zostać przetworzone tak, aby odniesienia (linki) wskazywały na serwer skrajny. Takie działanie jest wymagane, aby treści były poprawnie serwowane z pożąданiej lokalizacji. Podobnie, pliki cookies muszą być dostosowane, aby współpracowały z serwerem skrajnym, co może wymagać zmian zarówno w samych plikach cookies, jak i w mechanizmach ich obsługi.
- **Wymagane jednorazowe połączenie z serwerem centralnym:** Użytkownik musi nawiązać jednorazowe połączenie z serwerem centralnym, który następnie przekierowuje go do odpowiedniego serwera skrajnego. To dodatkowy krok w procesie dostarczania treści, co może wpływać na czas odpowiedzi i wydajność strony internetowej. Ponadto, w przypadku problemów z serwerem centralnym, może to prowadzić do niedostępności treści dla użytkowników.
- **Dostawcy usług hostingowych mają wgląd w serwer skrajny:** Dostawcy, którzy obsługują serwery skrajne, mają pewien poziom wglądu w przekazywane treści. Ponieważ serwer centralny przekierowuje użytkowników do konkretnych serwerów skrajnych na podstawie geolokalizacji, dostawcy hostingu mogą mieć dostęp do danych przekazywanych z i do serwerów skrajnych. To może rodzić obawy dotyczące prywatności i bezpieczeństwa danych użytkowników.

Warto zwrócić uwagę, że mimo tych wad, programowe podejście do budowy sieci CDN może nadal być atrakcyjne dla administratorów i firm, zwłaszcza tych, które chcą uniknąć inwestycji w infrastrukturę serwerową i skupić się na bardziej elastycznym dostarczaniu treści. Warto jednak dokładnie rozważyć te wady i zalety przed podjęciem decyzji o zastosowaniu tego podejścia.

4. Projekt serwera centralnego

Autor: Maciej Bandura

4.1 Pojęcie serwera centralnego

W programowym podejściu do budowy sieci CDN, serwer centralny pełni klu-
czową rolę w zarządzaniu i kierowaniu ruchem sieci. W przeciwnieństwie do konwencjo-
nalnych sieci CDN, które polegają na serwerach DNS do przekierowywania użytkowni-
ków do odpowiednich serwerów skrajnych, programowa sieć CDN wykorzystuje jeden
scentralizowany serwer główny, znany jako serwer centralny. W tradycyjnych sieciach
CDN, **serwery DNS** (Domain Name System) odgrywają kluczową rolę w kierowa-
niu ruchem użytkowników do odpowiednich serwerów skrajnych. Proces ten przebiega
następującą: w momencie, gdy użytkownik wpisuje adres URL witryny do przeglą-
darki internetowej, urządzenie wysyła żądanie DNS, które ma na celu przekształcenie
nazwy domeny na konkretny adres IP. To pozwala na zidentyfikowanie serwera, na
którym znajduje się treść witryny. W tradycyjnych sieciach CDN żądanie DNS nie
jest kierowane do standardowego serwera DNS, ale trafia do serwera DNS specjalnie
skonfigurowanego w sieci CDN. Ten serwer DNS jest zaprogramowany w taki sposób,
że dynamicznie przekierowuje użytkowników do najbliższych serwerów skrajnych, na
których przechowywana jest zawartość witryny. Działanie serwera DNS w sieci CDN
polega na analizie informacji geolokalizacyjnych oraz innych danych, takich jak ob-
ciążenie serwerów skrajnych, dostępność połączeń i wiele innych dotyczących analizy
wydajności sieci. Na podstawie tych informacji serwer DNS podejmuje decyzję o tym,
który serwer skrajny będzie najbardziej odpowiedni do obsługi danego żądania. Kiedy
użytkownik zostaje przekierowany na odpowiedni serwer skrajny, treść witryny jest
dostarczana z lokalnego źródła. To oznacza, że zamiast pobierania treści z jednego
globalnego - **źródłowego serwera**, użytkownik otrzymuje treść z serwera skrajnego,
który jest najbliżej niego geograficznie. Dzięki temu osiągana jest szybkość dostępu
oraz efektywność dostarczania treści.

Kluczowym zadaniem serwera centralnego jest udostępnienie **interfejsu do za-
rządzania całą siecią** za pomocą interfejsu programistycznego API (Application
Programming Interface). Dzięki temu interfejsowi serwer centralny komunikuje się z
serwerami skrajnymi i dostarcza im istotne informacje oraz zasady działania. API
serwera centralnego umożliwia konfigurację wielu aspektów sieci sCDN, takich jak:

- **Obsługiwane domeny:** Serwer centralny informuje serwery skrajne o domenach, które są obsługiwane w sieci CDN. Dzięki temu serwery skrajne wiedzą, które żądania obsłużyć lub przekierować.
- **Zasady zapisywania zawartości:** Poprzez API serwer centralny określa, jakie treści mają być przechowywane w pamięci podręcznej (cache) serwerów skrajnych. Może to obejmować całe strony internetowe, elementy multimedialne, pliki CSS czy JavaScript.
- **Zasady konwertowania zawartości:** Serwer centralny może kontrolować proces konwertowania zawartości poprzez uruchamianie konwertera zdjęć, czy też minifikatora plików CSS i JavaScript. To istotne, aby zapewnić optymalną wydajność dostarczania treści.
- **Informacje o blokowaniu treści:** Serwer centralny może dostarczać informacje o treściach, które powinny być blokowane, co jest ważne w przypadku treści niezgodnych z regulacjami prawnymi lub polityką firmy.

Dzięki temu interfejsowi API, serwer centralny umożliwia dynamiczne zarządzanie siecią sCDN i dostosowywanie jej działania do bieżących potrzeb i wymagań, co jest kluczowe dla efektywnego dostarczania treści internetowych.

4.2 Proces dostępu do zawartości

W programowym modelu sieci CDN, proces dostępu do zawartości internetowej ulega znaczającej transformacji w porównaniu do tradycyjnego schematu opartego na serwerach DNS. Tutaj, centralną rolę odgrywa serwer centralny, który nie tylko przetwarza żądania DNS, ale również aktywnie zarządza interakcją z użytkownikiem. Gdy użytkownik wpisuje adres URL witryny w swojej przeglądarce, jego urządzenie initiuje żądanie DNS, które jest jednak kierowane bezpośrednio do serwera centralnego. Różnica ta stanowi kluczowy element programowego podejścia do CDN, odmiennie niż w konwencjonalnych systemach, gdzie serwery DNS jedynie przekierowują użytkownika do odpowiednich serwerów skrajnych.

Serwer centralny, otrzymując żądanie, nie ogranicza się do prostej roli pośrednika, lecz przechodzi do bardziej złożonej interakcji. W tym miejscu wkracza nowy element – **"sandbox"**, czyli mini aplikacja przekazywana użytkownikowi. Ten sandbox, realizowany jako strona internetowa z elementem iframe, nie tylko umożliwia

renderowanie zawartości z serwera skrajnego, ale robi to na domenie, z której pochodziło pierwotne żądanie użytkownika. Dzięki temu, z punktu widzenia końcowego odbiorcy, cały proces wydaje się jednolity i niezauważalny, mimo że właściwe przetwarzanie odbywa się w innej lokalizacji. Kluczowym aspektem działania sandboxa jest również pozyskiwanie informacji geolokalizacyjnych użytkownika. Podczas pierwszego kontaktu z sandboxem, aplikacja ta prosi o dostęp do danych geolokalizacyjnych użytkownika, korzystając z mechanizmów JavaScript i API geolokalizacyjnego. Uzyskane w ten sposób koordynaty są następnie przesyłane na serwer centralny, który wykorzystuje te informacje do dalszego przetwarzania żądania.

W przypadku, gdy użytkownik nie udzieli zgody na udostępnienie swojej lokalizacji, serwer centralny podejmuje alternatywną strategię. W tej sytuacji, na serwer przychodzi informacja bez koordynatów lokalizacyjnych użytkownika. Aby poradzić sobie z tym wyzwaniem, serwer wykorzystuje zewnętrzną usługę o nazwie **ip-api**. Ta usługa analizuje adres IP użytkownika i na jego podstawie dostarcza podstawowe informacje, w tym przybliżone dane geolokalizacyjne. Takie podejście pozwala na oszacowanie lokalizacji, nawet jeśli dane nie pochodzą bezpośrednio od użytkownika.

The screenshot shows a web application interface titled "API Demo". At the top, there is a search bar with the placeholder "Search any IP address/domain" and a blue "SEARCH" button. Below the search bar, the IP address "83.28.81.221" is entered. To the right of the search bar is a map of the Kielce region in Poland, with various towns and villages labeled. On the left side of the map, there is a JSON response from the ip-api service:

```
{
  "query": "83.28.81.221",
  "status": "success",
  "continent": "Europe",
  "continentCode": "EU",
  "country": "Poland",
  "countryCode": "PL",
  "region": "26",
  "regionName": "Świętokrzyskie",
  "city": "Kielce",
  "district": "",
  "zip": "25-002",
  "lat": 50.8693,
  "lon": 20.6241,
  "timezone": "Europe/Warsaw",
  "offset": 3600,
  "currency": "PLN",
  "isp": "Orange Polska Spolka Akcyjna",
  "org": "Orange Polska S.A.",
  "as": "AS5617 Orange Polska Spolka Akcyjna",
  "asname": "TPNET",
  "mobile": false,
  "proxy": false,
  "hosting": false
}
```

Rysunek 4.1. Informacje o użytkowniku na podstawie jego adresu IP¹⁰

Na podstawie otrzymanych danych, zarówno z geolokalizacji pozyskanej od użytkownika, jak i z usługi ip-api, serwer centralny podejmuje decyzję o wyborze najodpowiedniejszego serwera skrajnego. W tym miejscu, istotne staje się rozróżnienie między serwerami działającymi na szyfrowanym połączeniu HTTPS a tymi, które tego szyfrowania nie używają.

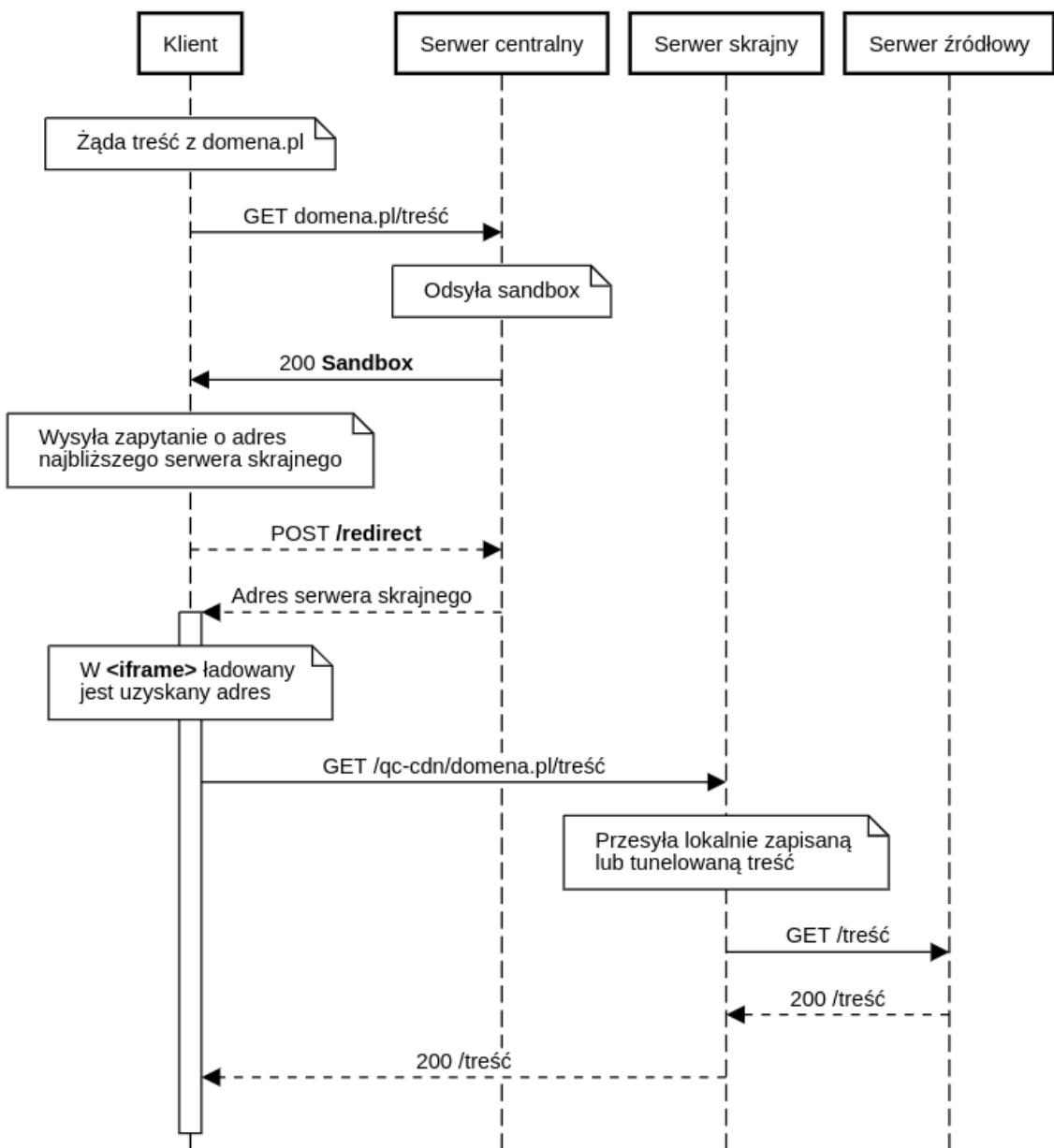
Jeżeli wybrany serwer skrajny używa połączenia HTTPS, jego zawartość jest bezpośrednio wyświetlana w iframe wewnętrz sandboxa. **Iframe** (*inline frame*) jest elementem HTML używanym do osadzenia zawartości z jednej strony internetowej wewnętrz innej. Jego zastosowanie w programowym CDN ma kluczowe znaczenie, gdyż pozwala na bezpieczne wyświetlanie treści z serwera skrajnego wewnętrz sandboxa, bez naruszania polityk bezpieczeństwa i prywatności przeglądarki. Iframe działa jak rodzaj okna, przez które użytkownik może oglądać zawartość znajdującą się na innym serwerze, jednocześnie pozostając na oryginalnej stronie.

Kluczową kwestią tutaj jest obsługa plików cookies. Współczesne przeglądarki internetowe mają zaostrzone zasady dotyczące dostępu do plików cookies przez iframe, szczególnie jeśli pochodzą z różnych domen. Zgłębiając problematykę obsługi plików cookies w kontekście programowego podejścia do budowy sieci CDN, napotykamy na kwestie związane z atrybutem SameSite cookies. Atrybut **SameSite** jest stosunkowo nowym dodatkiem do specyfikacji cookies, który pozwala na kontrolowanie sposobu ich wysyłania w kontekście **żądań międzystronowych** (cross-site). W przeglądarkach domyślnie ustawienie SameSite jest na Lax, co oznacza, że cookies nie są obsługiwane przy żądaniach generowanych z innych domen, co bezpośrednio przekłada się na ich niedostępność w iframe z załadowaną zawartością z innej domeny. Dla iframe, które ładują treści z serwerów skrajnych, może to stanowić problem, ponieważ tradycyjne pliki cookies nie będą dostępne, co uniemożliwia na przykład przekazanie informacji serwerowi centralnemu o tym, jakiej domeny dotyczy aktualne żądanie. Aby cookies mogły być używane w takim scenariuszu, atrybut **SameSite** musi być ustawiony na None, co pozwala na ich wysyłanie w kontekście międzystronowym, w tym do iframe renderującego zawartość z serwera skrajnego. Jednakże, ustawienie **SameSite=None** stawia kolejne wymaganie – konieczność zastosowania atrybutu **Secure**. Atrybut Secure wymusza, aby cookies były przesyłane tylko przez szyfrowane połączenia HTTPS. Jest to wymóg bezpieczeństwa mający na celu ochronę danych przesyłanych cookies przed przechwyceniem przez osoby trzecie w niezabezpieczonych połączeniach.

¹⁰Zewnętrzna usługa ip-api: <https://ip-api.com/>

Kolejnym, wyzwaniem jest zapewnienie, aby użytkownik doświadczał płynnej nawigacji i interakcji, jak gdyby faktycznie przechodził między różnymi stronami. Tutaj kluczową rolę odgrywa JavaScript i technika komunikacji znana jako **window.postMessage()**. Mechanizm, ten umożliwia bezpieczne przekazywanie wiadomości między obiektami typu **Window** pochodząymi z różnych źródeł. W naszym przypadku między stroną wyrenderowaną w iframe a sandboxem. Dzięki temu, gdy użytkownik nawiguje wewnątrz iframe, skrypt JavaScript może informować stronę główną o zmianach. Na przykład, jeśli użytkownik przechodzi na nową stronę wewnątrz iframe, skrypt może wysłać wiadomość do strony głównej, aby zaktualizować tytuł strony na karcie przeglądarki. Ponadto, może to także umożliwić dodanie nowego wpisu do historii przeglądania przeglądarki, co pozwala na korzystanie z przycisków nawigacyjnych takich jak '*wstecz*' i '*dalej*'. Takie podejście jest niezwykle ważne, gdyż użytkownik, choć technicznie pozostaje na oryginalnej stronie w sandboxie, doświadcza zmian treści wewnątrz iframe. Zapewnia to płynne i niezauważalne przejście, które jest kluczowe dla doświadczenia użytkownika w programowym CDN, utrzymując jednocześnie wysoki poziom bezpieczeństwa i prywatności.

W sytuacji, gdy serwer skrajny nie korzysta z szyfrowanego połączenia HTTPS, przeglądarki internetowe mogą blokować lub ograniczać dostęp do plików cookies przez iframe z powodów bezpieczeństwa. W związku z tym, w takich przypadkach sieć sCDN podejmuje decyzję o przekierowaniu klienta bezpośrednio na serwer skrajny. To rozwiązanie pozwala uniknąć potencjalnych problemów z bezpieczeństwem i prywatnością danych, ale wymaga od użytkownika opuszczenia pierwotnie zażądanej domeny. Jest to kompromis między bezpieczeństwem a płynnością użytkowania, który musi być dokonany w ramach programowego CDN.



Rysunek 4.2. Schemat działania sandbox'a

4.3 Dobór serwera skrajnego przez serwer centralny

Głównym punktem analizy procesu doboru serwera skrajnego jest złożony proces decyzyjny uwzględniający wiele czynników. Serwer centralny, analizuje szereg danych, aby wybrać optymalny punkt skrajny do obsługi żądania użytkownika. Proces ten rozpoczyna się od identyfikacji nazwy domeny, która jest kluczowym elementem w powiązaniu użytkownika z zawartością. Serwer centralny wykorzystuje informacje o domenie, aby określić, które serwery skrajne daną treść obsługują. Następnie, biorąc pod uwagę geolokalizację użytkownika, serwer centralny jest w stanie określić, które z dostępnych serwerów skrajnych znajdują się najbliżej geograficznie. Jednocześnie serwer centralny musi wziąć pod uwagę stan sieci, szczególnie to, które serwery skrajne są aktywne i gotowe do działania. Informacje te są istotne, ponieważ nawet serwer geograficznie najbliższy może być chwilowo niedostępny. W takich przypadkach, serwer centralny musi zdecydować się na przekierowanie żądania do innego punktu skrajnego, który zapewni dobre warunki dostępu do treści. Po skompletowaniu listy działających serwerów skrajnych, na których domena jest zarejestrowana, serwer centralny przeprowadza proces wyboru najodpowiedniejszego punktu. Ta selekcja uwzględnia zarówno geolokalizację użytkownika, jak i serwerów skrajnych. W procesie doboru serwera skrajnego przez serwer centralny, krytycznym elementem jest określenie odległości między geolokalizacją użytkownika a lokalizacjami serwerów skrajnych. Teoretycznie, zadanie to polega na znalezieniu najkrótszej drogi między dwoma punktami na powierzchni sferycznej, którą w przybliżeniu jest Ziemia. W tym kontekście, serwer centralny wykorzystuje zestaw informacji geograficznych i matematycznych algorytmów do obliczenia dystansu. Odległość ta jest mierzona przy użyciu formuły haversine, która uwzględnia krzywiznę Ziemi, co jest niezbędne do uzyskania dokładnych pomiarów na globalnej skali. Algorytm haversine wykorzystuje szerokość i długość geograficzną obu punktów, przeliczając je na radiany i stosując trygonometrię sferyczną do określenia najkrótszego łuku między nimi. Z tego powodu, nawet na dużych dystansach, jesteśmy w stanie wyznaczyć relatywnie dokładną odległość między użytkownikiem a serwerem skrajnym.

$$2r \sin^{-1} \left(\sqrt{\sin^2 \left(\frac{\varphi_2 - \varphi_1}{2} \right) + \cos \varphi_1 * \cos \varphi_2 * \sin^2 \left(\frac{\lambda_2 - \lambda_1}{2} \right)} \right)$$

gdzie:

r - promień ziemi

φ_1, φ_2 - szerokość geograficzna 1 i 2 punktu

λ_1, λ_2 - długość geograficzna 1 i 2 punktu

5. Projekt serwera skrajnego

Autor: Marcin Ślusarczyk

5.1 Pojęcie serwera skrajnego

Serwery skrajne (*edge servers*), zwane również serwerami brzegowymi, punktami końcowymi, stanowią fundament działania sieci CDN jak i sCDN. Ich głównym zadaniem jest przechowywanie i dostarczanie treści jak najbliżej geograficznego położenia użytkownika końcowego, co jest kluczowe dla minimalizowania czasu ładowania strony i poprawy ogólnej wydajności przesyłania danych. Serwery te są rozmiieszczone w strategicznych lokalizacjach na całym świecie, co pozwala na dywersyfikację punktów dostępu do treści i zapewnia ich wysoką dostępność oraz niezawodność. Poprzez przechowywanie kopii danych na serwerach skrajnych, sieci CDN mogą znacząco redukować obciążenie serwerów źródłowych oraz czas dostępu do treści, co jest szczególnie istotne w przypadku obsługi dużego ruchu sieciowego i dostarczania treści o dużych rozmiarach, takich jak pliki wideo, grafiki, czy skrypty stron internetowych.

Serwery skrajne są wielofunkcyjnymi elementami architektury sieci sCDN, które pełnią szereg zadań mających na celu optymalizację dostarczania treści do użytkownika końcowego. Jednym z głównych zadań serwera skrajnego jest tunelowanie zapytań, co polega na przyjmowaniu zapytań od użytkownika i kierowaniu ich dalej – czy to do serwera źródłowego, czy też do lokalnie zapamiętanej zawartości. Dzięki temu możliwe jest szybkie i efektywne dostarczenie odpowiedzi bez konieczności każdorazowego odwoływania się do serwera głównego. Kolejną kluczową funkcją jest pobieranie zawartości z serwera źródłowego. Gdy użytkownik żąda określonych danych, serwer skrajny może je pobrać z serwera źródłowego, jeżeli nie posiada ich w swojej pamięci podręcznej. To zadanie jest niezbędne w przypadku pierwszego żądania danej treści lub gdy lokalnie przechowywane dane są przestarzałe. Serwer skrajny zajmuje się również przetwarzaniem pobranej zawartości, co może obejmować takie operacje jak kompresja, minifikacja, lub aplikowanie określonych zasad dostępu. Dzięki temu treść może być dostosowana do struktury sieci programowego CDN, wymagań użytkownika oraz obostrzeń bezpieczeństwa i wydajności.

Lokalne zapamiętywanie zawartości, znane jako pamięć cache serwera, umożliwia serwerom skrajnym przechowywanie kopii popularnych danych, co znacząco przyspiesza ich dostępność przy kolejnych zapytaniach. Jest to fundament działania sieci CDN,

który przyczynia się do redukcji opóźnień i obciążenia sieci. Poza przechowywaniem danych, serwery skrajne odpowiadają za ich przekazywanie do klienta. Oznacza to, że kiedy użytkownik żąda treści, serwer skrajny szybko dostarcza je z lokalnej pamięci podręcznej lub po pobraniu z serwera źródłowego, co zapewnia płynność i szybkość ładowania się stron internetowych i aplikacji.

Dodatkowo, serwery skrajne działają jako punkty dostępu dla klientów, umożliwiając im interakcję z siecią CDN. Są to pierwsze punkty kontaktu dla użytkownika końcowego, co sprawia, że ich wydajność i dostępność mają bezpośredni wpływ na doświadczenie użytkownika.

Serwery skrajne oferują interfejs programistyczny (API), który umożliwia zarządzanie nimi i konfigurację domen. Dostęp do API pozwala na dostosowywanie działania serwerów do indywidualnych potrzeb i wymagań, co jest szczególnie ważne w elastycznym środowisku sieciowym, gdzie szybka adaptacja do zmieniających się warunków jest kluczowa. Użytkownicy oraz administratorzy sieci mają możliwość zarządzania zarejestrowanymi już domenami jak i również mogą dodawać nowe domeny do obsługi przez dany serwer skrajny. Każdy serwer skrajny w architekturze sCDN posiada mechanizmy zabezpieczające komunikację, a jednym z nich jest token autoryzacyjny. Jest to unikalny identyfikator, który musi być dołączony do każdego żądania wysyłanego przez klienta – przesyłany zwykle w nagłówku HTTP jako część protokołu autoryzacji. Token ten służy jako środek weryfikacji uprawnień użytkownika, który próbuje uzyskać dostęp do zasobów serwera skrajnego. Dzięki wykorzystaniu tokenu autoryzacyjnego, serwer skrajny może efektywnie zarządzać dostępem do swoich zasobów, odróżniając legalne żądania od potencjalnie szkodliwych lub nieautoryzowanych prób dostępu. Jeśli żądanie przychodzące do serwera skrajnego nie zawiera prawidłowego tokena w nagłówku, serwer ignoruje takie zapytania. To stanowi podstawową linię obrony przed atakami i nieautoryzowanym dostępem, zapewniając, że tylko usługi z odpowiednimi uprawnieniami mogą korzystać z danych i funkcjonalności oferowanych przez serwer.

5.2 Ścieżki specjalne na serwerze skrajnym

Typowy adres HTTP/HTTPS - URL (*Uniform Resource Locator*) składa się z trzech głównych komponentów:

https:// qc-cdn.pl /pl/zasob

Elementy ścieżki oddzielone znakiem / nazywamy segmentami. Na serwerze skrajnym w programowej sieci CDN istnieją dwie główne ścieżki specjalne, które odpowiadają za różne aspekty działania sieci. Jest to zależne od wartości w pierwszym segmencie. Pierwsza z nich, prefiksowana jako **/qc-api**, stanowi zbiór interfejsów API dedykowanych zarządzaniu i konfiguracji serwera skrajnego. Przez te punkty dostępu, użytkownicy oraz administratorzy mają możliwość przeprowadzania szerokiego zakresu operacji konfiguracyjnych, co obejmuje takie działania jak rejestracja, edycja, czy usuwanie domen na serwerze skrajnym. Ponadto, interfejsy te pozwalają na pobieranie listy plików przechowywanych w lokalnej pamięci cache serwera skrajnego. Umożliwiają one również definiowanie oraz modyfikację reguł cache'owania, które determinują jakie treści mają być przechowywane lub tunelowane. Jak i również zarządzanie regułami konwerterów, które są stosowane do przetwarzania treści dla poszczególnych domen.

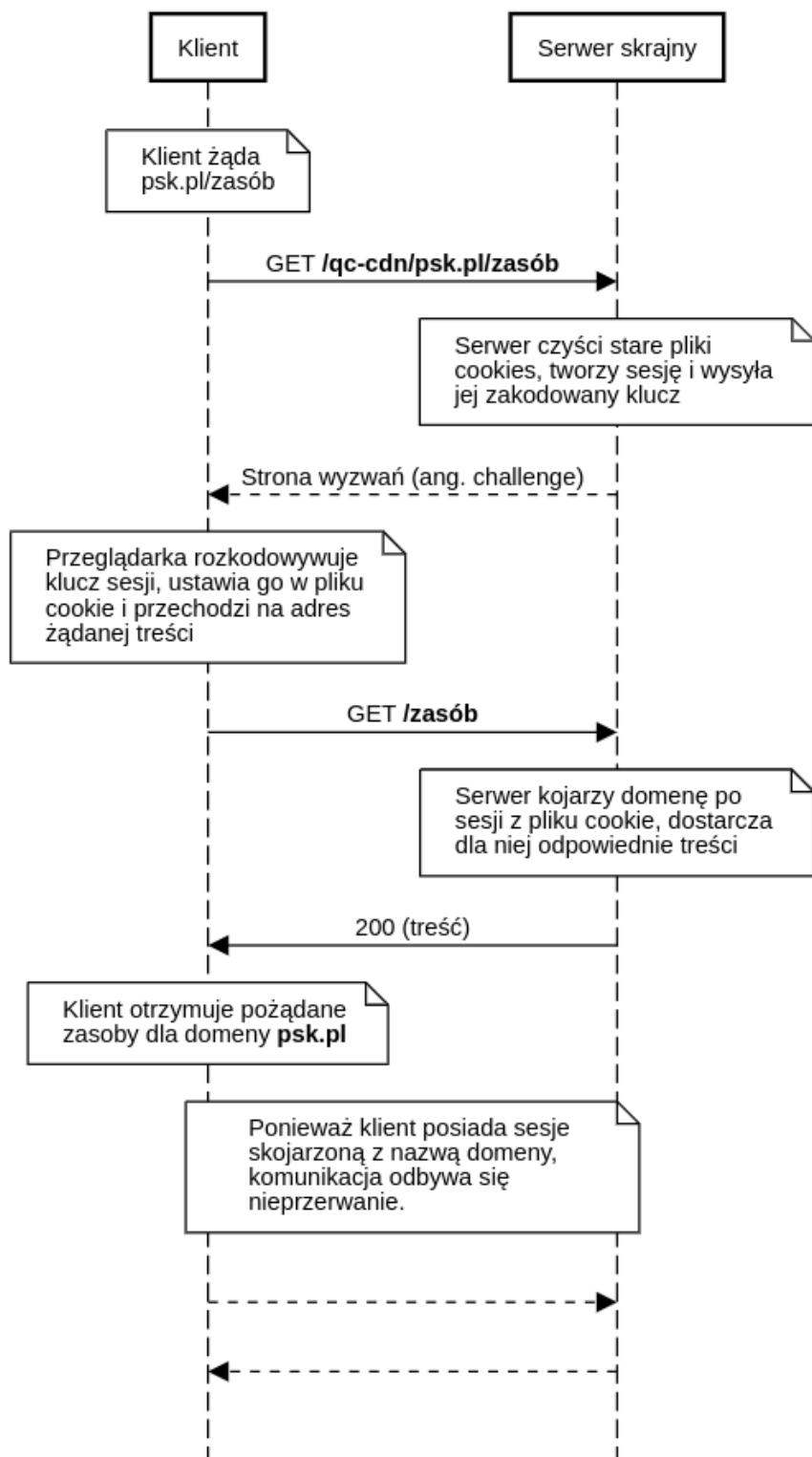
Kolejna ścieżka specjalna, prefiksowana przy pomocy **/qc-cdn**, jest przeznaczona bezpośrednio do serwowania treści. Pracuje ona w oparciu o prostą, lecz efektywną konwencję adresowania. Struktura adresu URL wykorzystuje nazwę domeny oraz ścieżkę do zawartości na serwerze źródłowym, umożliwiając serwerowi skrajnemu lokalizację i dostarczenie żądanej treści użytkownikowi końcowemu. Format adresu URL jest przezroczysty i przewidywalny.

https:// serwer skrajny / qc-cdn / nazwa domeny / ścieżka dostępu
Adres serwera skrajnego na serwerze źródłowym

Warto przy tym podkreślić, że w programowym podejściu do budowy sieci CDN ścieżki takie jak **qc-api** i **qc-cdn** są zarezerwowane dla funkcjonalności sieci i nie mogą być wykorzystywane na serwerze źródłowym. Ograniczenie to zapewnia uniwanie konfliktów nazewnictwa i zachowanie spójności w zarządzaniu i dostarczaniu treści. Dzięki wyraźnemu oddzieleniu ścieżek zarządzania od ścieżek dostarczania treści, sieć sCDN może działać sprawnie i bez zakłóceń, zapewniając użytkownikom szybki i niezawodny dostęp do żądanych zasobów.

5.3 Proces komunikacji pomiędzy klientem a serwerem skrajnym

Proces komunikacji między klientem a serwerem skrajnym w ramach programowej sieci CDN można opisać jako serię kroków, które zapewniają sprawną wymianę treści. Na początku klient wyraża chęć dostępu do określonej domeny i zasobu, na przykład pragnie załadować stronę dostępną pod adresem **psk.pl/zasób**. Żądanie to jest kierowane do serwera skrajnego przez odpowiednio skonstruowany URL, który zawiera prefiks ścieżki **/qc-cdn/**. Serwer skrajny, otrzymując takie żądanie, przeprowadza najpierw operację czyszczenia obcych plików cookies. Jest to krok wstępny, mający na celu zapewnienie, że żadne pliki cookies powiązane z innymi domenami świadczonymi na tym samym serwerze skrajnym, nie będą istniały w bieżącej sesji. Następnie serwer generuje nową sesję i wysyła ją użytkownikowi zakodowaną w formacie strony wyzwania ("*challenge*") w celu weryfikacji przeglądarki użytkownika - jest to zabezpieczenie "anty botowe" powstrzymujące spam i niepożądane zapytania do serwera. Po rozkodowaniu przez przeglądarkę klucza sesji javascript zapisuje go w pliku cookies i dokonuje przekierowania na właściwy adres zasobu, w tym przypadku na ścieżkę **/zasób**. Jest to ważny krok, gdyż zarówno serwer jak i przeglądarka muszą posiadać informacje, jakiej domeny dotyczy dane żądanie. W przypadku programowej sieci CDN ta informacja jest przechowywana w pliku cookies. W dalszej kolejności, serwer skrajny kojarzy domenę na podstawie pliku cookie i dostarcza odpowiednią dla niej treść. Dzięki temu mechanizmowi, treść jest spersonalizowana i dostosowana do potrzeb użytkownika. W przypadku kolejnych żądań o treść dotyczących tej samej domeny, dzięki wcześniejszemu ustawnieniu plików cookie, komunikacja odbywa się bez przeskódek, a klient może kontynuować interakcję z zasobami, tak jakby faktycznie był połączony z serwerem źródłowym. Przechowywanie nazwy domeny w pliku cookies okazało się być najbardziej optymalną strategią do zarządzania sesjami w programowych sieciach CDN. Alternatywne metody, takie jak przekazywanie tej informacji poprzez parametry GET w URL, były rozważane, ale ten sposób uznano za bardziej skomplikowany i generujący więcej wyzwań. Parametry GET mogą komplikować zarządzanie URL-ami lub zwiększać ryzyko wystąpienia błędów w skryptach.



Rysunek 5.1. Proces komunikacji pomiędzy klientem a serwerami skrajnymi

5.4 Mechanizm świadczenia treści na serwerze skrajnym

Mechanizm dostarczania treści przez serwery skrajne w programowej sieci CDN jest zbudowany w taki sposób, aby użytkownik nie doświadczał żadnej różnicy w dostępie do zasobów niezależnie od tego, czy są one serwowane bezpośrednio z serwera źródłowego, czy z serwera skrajnego. Ten niewidoczny dla użytkownika proces zapewnia płynność korzystania z usług internetowych, jednocześnie wykorzystując optymalizację dostarczania danych oferowaną przez rozwiązania CDN. Aby w pełni zrozumieć, jak ten system działa, ważne jest wyjaśnienie kluczowych pojęć związanych z typami zawartości, które serwery CDN obsługują, a mianowicie rozróżnienie między kontentem statycznym a dynamicznym.

Kontent statyczny obejmuje elementy strony, które pozostają niezmienne niezależnie od liczby żądań, takie jak pliki graficzne, arkusze stylów CSS, skrypty JavaScript czy multimedia. Taka zawartość może być efektywnie przechowywana na serwerach skrajnych, dzięki czemu jest dostarczana użytkownikowi z minimalnym opóźnieniem. Ponadto, kontent statyczny nie wymaga częstych aktualizacji, co pozwala na jego długoterminowe przechowywanie na serwerach skrajnych bez konieczności częstego odświeżania, co jest korzystne zarówno pod względem wydajności, jak i optymalizacji wykorzystania zasobów.

Zawartość dynamiczna to treści generowane na żądanie, często personalizowane i zmienne, jak strony kont użytkowników, wyniki wyszukiwania lub informacje na bieżąco aktualizowane z baz danych. W tym przypadku, strategie cache'owania muszą być inteligentne i elastyczne, aby zapewnić aktualność danych przy zachowaniu efektywności dostarczania treści. W kontekście dynamicznej zawartości, równie istotna staje się metoda **tunelowania żądań**. Jest to proces, gdzie serwer skrajny działa jako pośrednik, przekazując zapytania użytkownika do serwera źródłowego i z powrotem. Jest to szczególnie ważne dla treści dynamicznych, które wymagają częstych aktualizacji i interakcji z bazą danych. Jednak rola serwera skrajnego jako pośrednika nie ogranicza się jedynie do przesyłania żądań. Wymaga ona także szczegółowego przekazywania nagłówków HTTP, plików cookie, a także innych informacji niezbędnych do prawidłowego obsłużenia konkretnego żądania. To zadanie jest złożone i wymaga precyzyjnej koordynacji, ponieważ każdy element żądania, od nagłówków po ciasteczka, może mieć kluczowe znaczenie dla dostarczania spersonalizowanej i odpowiedniej treści użytkownikowi. Serwery skrajne muszą zatem nie tylko przekazywać dane, ale również zachować integralność wszystkich przesyłanych informacji,

co jest kluczowe dla zapewnienia spójności doświadczeń użytkowników i optymalnego działania aplikacji internetowych.

Zrozumienie roli tunelowania żądań w kontekście treści dynamicznych prowadzi nas do kolejnego kluczowego elementu sieci CDN - **cache'owania**. Jest to technika przechowywania często żądanych treści w lokalnej pamięci serwera skrajnego, co znacznie skraca czas odpowiedzi na zapytania użytkowników i zmniejsza obciążenie serwerów źródłowych. Dzięki zaawansowanym regułom, system sCDN decyduje, które elementy mogą być cache'owane. W systemie sCDN każda reguła zapisywana treści do pamięci podręcznej serwera skrajnego jest przypisana do konkretnej domeny. Reguły te są konfigurowane z różnymi parametrami, które określają, jak będą one interpretowane. Kluczowymi elementami konfiguracji reguły są:

- **Priorytet:** Określa kolejność, w jakiej reguły są stosowane. Reguły o wyższym priorytecie mają pierwszeństwo przed tymi z niższym, co pozwala na kontrolowanie, które zasady cache'owania są priorytetowe w przypadku nakładania się warunków.
- **Selektor dostępu:** Definiuje kryteria, na podstawie których reguła jest stosowana. Może to być ścieżka do pliku na serwerze, typ zawartości (określony przez MIME type), który umożliwia precyzyjne dopasowanie reguły do określonych rodzajów plików, jak obrazy, arkusze stylów czy skrypty.
- **Wyrażenie regularne (regex):** To potężne narzędzie używane do definiowania złożonych wzorców, które reguła powinna obsługiwać. Dzięki wyrażeniom regularnym można określić, które elementy w ścieżce lub zawartości powinny aktywować daną regułę.
- **Flaga aktywacji:** Wskazuje, czy reguła jest włączona czy wyłączona. Pozwala to na elastyczne zarządzanie regułami, umożliwiając ich szybkie włączanie lub wyłączanie bez konieczności usuwania czy modyfikacji całej reguły.

Dzięki tym parametrom, serwer skrajny jest w stanie dokładnie określić, które elementy strony powinny być wczytywane z lokalnej pamięci, a które wymagają dynamicznego odświeżenia. To zapewnia, że cache'owane są tylko te treści, które rzadko ulegają zmianie, co z kolei przekłada się na szybsze ładowanie stron i mniejsze obciążenie serwera źródłowego. Jest to szczególnie ważne w przypadku dużych serwisów internetowych, gdzie efektywna dystrybucja treści może znacząco wpływać na wydajność całej sieci.

Pojęcie konwertera w programowej sieci CDN pełni kluczową rolę jako manipulatora treści, umożliwiając adaptację i optymalizację przesyłanych danych. Zastosowanie różnych reguł konwersji pozwala na dostosowanie treści do specyficznych wymagań sieci oraz optymalizację ich dla końcowego użytkownika. Poniżej przedstawiono podstawowe reguły, które będą dokładniej omówione w dalszej części:

- **URL REPL**: Ta reguła odpowiada za zamianę adresów URL serwera źródłowego na adres serwera skrajnego. Jest to kluczowe dla zachowania wymagań systemu sCDN, umożliwiając użytkownikom dostęp do treści z serwera najbliższego ich lokalizacji.
- **HT SANDBOX**: Dotyczy dołączania specjalnych skryptów, które umożliwiają efektywną komunikację z „*sandboxem*”. Zapewnia to, że zawartość jest odpowiednio wyświetlana i zarządzana w środowisku użytkownika.
- **IMG MIN**: Kompresor zdjęć, który zmniejsza rozmiar plików graficznych bez znaczącej utraty jakości. Dzięki temu strony ładują się szybciej, co jest szczególnie ważne dla użytkowników w obszarach z wolniejszym dostępem do internetu.
- **JS MIN**: Minifikator JS, który skraca i optymalizuje skrypty JavaScript. Redukuje to czas ładowania stron i poprawia ogólną wydajność.
- **CSS MIN**: Podobnie jak JS_MIN, ta reguła zajmuje się minifikacją arkuszy stylów CSS. Efektem jest szybsze ładowanie się stylów i mniejsze obciążenie sieci.
- **HT ADS***: Reguła ta dotyczy wstawiania reklam na stronach internetowych zarejestrowanych w serwisie sCDN. Ta reguła nie jest dostępna do konfiguracji przez użytkownika. W bezpłatnym pakiecie jest ona zawsze włączona. Zaś w płatnym wyłączona.

Każda z tych reguł ma za zadanie dopracowanie treści przesyłanych przez sieć sCDN w taki sposób, aby były one jak najbardziej dostosowane do architektury i wymagań sieci jak i potrzeb użytkowników końcowych, zarówno pod względem szybkości dostępu, jak i jakości wyświetlania treści.

6. Implementacja serwera centralnego

Autor: Maciej Bandura

6.1 Wykorzystywane technologie

Serwer centralny został zbudowany z wykorzystaniem nowoczesnych i sprawdzonych rozwiązań technologicznych, które zapewniają wysoką wydajność, niezawodność oraz elastyczność w zarządzaniu treścią i komunikacji w sieci. Podstawą technologiczną serwera jest **Java** w wersji 17, która jest jednym z najbardziej popularnych i wszechstronnych języków programowania. Wykorzystanie Javy umożliwia tworzenie niezawodnych, bezpiecznych i łatwo skalowalnych aplikacji, które są idealne do zarządzania złożonymi systemami, jakim z pewnością jest serwer centralny w sieci sCDN. Do budowy serwera centralnego wykorzystano również framework **Spring Boot** w wersji 3.2.0. Spring Boot jest cenionym rozwiązaniem w świecie Java, oferującym szeroką gamę narzędzi do szybkiego i efektywnego tworzenia aplikacji, w tym automatyczną konfigurację, wsparcie dla mikroserwisów oraz łatwość w utrzymaniu i rozbudowie aplikacji. Użycie Spring Boot znacznie przyspiesza rozwój i wdrażanie aplikacji, a także zapewnia wsparcie dla najnowszych technologii i praktyk programistycznych.



Spring Boot, oferuje elastyczne podejście do integracji z bazami danych za pomocą **JDBC** (*Java Database Connectivity*). Connector JDBC w Springu jest kluczowym elementem, który umożliwia aplikacjom nawiązywanie połączeń z bazami danych, wykonując operacje **CRUD** (*Create, Read, Update, Delete*) oraz zarządzając transakcjami. Spring udostępnia proste i przejrzyste metody konfiguracji połączeń JDBC, umożliwia łatwe i szybkie ustawienie połączenia z bazą danych. Konfiguracja jest realizowana za pomocą pliku application.properties. Spring oferuje również klasę **JdbcTemplate**, która upraszcza operacje na bazie danych, eliminując konieczność pisania *boilerplate code*. JdbcTemplate zajmuje się zarządzaniem zasobami, jak

otwieranie i zamykanie połączeń, a także oferuje wygodne metody do wykonywania zapytań i aktualizacji. Spring upraszcza przetwarzanie wyników zapytań, eliminując potrzebę ręcznego przypisywania wartości z rekordów bazy danych do obiektów Java. Wykorzystywany do tego jest **BeanPropertyRowMapper**.

Hostowanie serwera Java w środowisku chmurowym, korzystając z serwisów takich jak Railway, Heroku czy AWS, stanowi dynamiczne rozwiązanie dla aplikacji internetowych, oferując elastyczność i skalowalność niezbędną do efektywnego zarządzania zasobami. Te platformy zapewniają nie tylko **skalowalność horyzontalną** - zdolność do dodawania dodatkowych serwerów lub instancji w odpowiedzi na wzrost zapotrzebowania - ale również całą gamę narzędzi i usług, które ułatwiają zarządzanie, monitorowanie i automatyzację aplikacji. Dzięki tym cechom, hostowanie w chmurze staje się coraz bardziej popularnym wyborem dla aplikacji Java, oferując nie tylko wysoką dostępność i niezawodność, ale również co jest powiązane z myślą przewodnią programowego CDN'a, uwalnia deweloperów od obowiązków związanych z utrzymaniem infrastruktury.

Baza danych wybrana do tego projektu to **MariaDB**, która jest wysokowydajną, bogatą w funkcje i skalowalną bazą danych, zapewniającą niezbędną prędkość i niezawodność dla systemu sCDN. MariaDB, jako system zarządzania bazą danych, wyróżnia się na tle innych popularnych rozwiązań SQL szeregiem charakterystycznych cech. Porównując go do innych znanych systemów, takich jak MySQL, PostgreSQL czy Microsoft SQL Server, można wyróżnić kilka kluczowych aspektów:

- **Kompatybilność z MySQL:** MariaDB początkowo powstała jako fork MySQL, co oznacza, że jest w pełni kompatybilna z MySQL. To sprawia, że migracja z MySQL do MariaDB jest zwykle prosta i bezproblemowa, a narzędzia i aplikacje stworzone dla MySQL zazwyczaj działają bez zmian na MariaDB.
- **Wydajność i Skalowalność:** MariaDB jest znana z lepszej wydajności w niektórych scenariuszach, szczególnie przy obsłudze dużych baz danych i zapytań. Została zoptymalizowana pod kątem szybkości i efektywności, oferując zaawansowane mechanizmy cache'owania i indeksowania.
- **Otwarte Źródło i Społeczność:** Jako projekt open source, MariaDB ma aktywną społeczność deweloperów, co sprzyja ciągłeemu rozwojowi i innowacjom. Jest to znacząca przewaga nad rozwiązaniami komercyjnymi, gdzie rozwój jest zależny od pojedynczej firmy.

- **Bezpieczeństwo:** MariaDB kładzie duży nacisk na aspekty bezpieczeństwa, oferując zaawansowane funkcje, takie jak szyfrowanie danych, zarówno w społeczeństku, jak i w transmisji, co jest ważne w przypadku wrażliwych danych.

Serwer centralny wykorzystuje **Redis** (*Remote Dictionary Server*) do zarządzania sesjami użytkowników. Ta nierelacyjna baza danych, przechowująca informacje w formacie klucz-wartość, oferuje wyjątkową szybkość i wydajność dzięki przechowywaniu danych w pamięci RAM, a nie na dysku. Jego szybkość w odczycie i zapisie danych znacznie przekłada się na szybkość działania aplikacji, co jest istotne przy zarządzaniu dużą liczbą sesji. Redis zapewnia także łatwą skalowalność, umożliwiając efektywne radzenie sobie ze wzrostem ruchu sieciowego. Ponadto, jego mechanizmy replikacji i trwałości danych zwiększały niezawodność systemu. Wykorzystanie Redisa w systemie logowania na serwerze centralnym, nie tylko poprawia wydajność zarządzania sesjami, ale również odciąża główną bazę danych, co może przyczyniać się do lepszego wykorzystania zasobów i obniżenia kosztów operacyjnych.



Rysunek 6.1. Porównanie prędkości odczytu różnych baz danych¹¹

Graficzny interfejs użytkownika oparty został na jednej z czołowych bibliotek JavaScript - React. Jest to nie tylko biblioteka, ale prawdziwy ekosystem dla twórców front-endowych, wyróżnia się swoim unikalnym podejściem do projektowania UI. Zamiast klasycznego podejścia do tworzenia stron, React skupia się na modularnym

¹¹Porównane baz danych: profil-software.com/blog/development/database-comparison-sql-vs-nosql-mysql-vs-postgresql-vs-redis-vs-mongodb

i komponentowym modelu, gdzie każdy element interfejsu jest niezależną jednostką, co znacząco upraszcza proces tworzenia i utrzymania skomplikowanych aplikacji webowych. Głównymi zaletami tego framework'u są między innymi, komponentowy model budowy. React opiera się na komponentach, co oznacza, że interfejs użytkownika składa się z wielu niezależnych, izolowanych i ponownie wykorzystywanych elementów. Każdy komponent jest odpowiedzialny za renderowanie części UI i może posiadać własny stan i logikę. Dodatkowo React wykorzystuje **wirtualny DOM** (*Document Object Model*), który pozwala na efektywne zarządzanie aktualizacjami interfejsu użytkownika. Zmiany w stanie komponentów są najpierw aplikowane do wirtualnego DOM, a następnie, za pomocą algorytmu porównującego, minimalnie modyfikują rzeczywisty DOM przeglądarki. Warto również wspomnieć, iż React jest rozwijany i utrzymywany przez Meta - firmę odpowiedzialną za Facebooka oraz dużą społeczność deweloperów, co gwarantuje regularne aktualizacje, nowe funkcjonalności i szybkie reagowanie na znalezione błędy.

Każda z użytych technologii wnosi unikalny wkład w funkcjonowanie serwera centralnego, czyniąc go nie tylko wydajnym, ale i elastycznym w adaptacji do różnych wymagań oraz łatwym w utrzymaniu i rozbudowie. Wspólnie tworzą one mocną i zaawansowaną infrastrukturę, która jest w stanie sprostać wyzwaniom stawianym przez wymagania programowej sieci CDN.

6.2 Struktura aplikacji

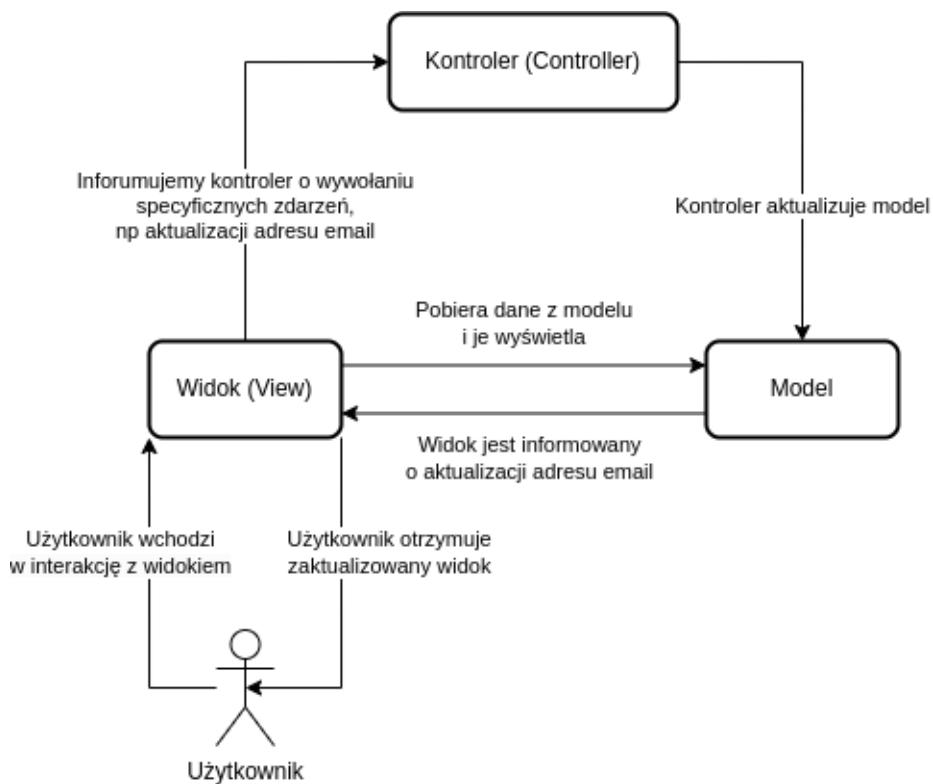
Specyfika projektu systemu sCDN wymagała jasnego i efektywnego sposobu komunikacji pomiędzy serwerem centralnym a serwerami skrajnymi. Jako, iż są to dwa oddzielne projekty w jednym musiał zostać precyzyjnie rozgraniczona. Aby sprostać temu wyzwaniu, został wykorzystany styl architektoniczny **REST API**, który zapewnia przejrzystość, skalowalność oraz łatwość integracji. Takie podejście pozwala na wyraźne oddzielenie różnych warstw systemu, zgodnie z wzorcem Model-View-Controller (MVC), co przekłada się na wyższą efektywność w zarządzaniu danymi i procesami.

Istotnym aspektem infrastruktury serwera centralnego jest zastosowanie wspomnianego wzorca MVC. W obecnych czasach, gdy technologie informatyczne szybko się rozwijają, odpowiednia struktura i architektura aplikacji są niezwykle ważne. Dzięki nim, aplikacje są bardziej wydajne, łatwiejsze w skalowaniu i prostsze w ob-

służbe. Sam **wzorzec MVC** (*Model-View-Controller*) to wzorzec architektoniczny szeroko stosowany w projektowaniu aplikacji internetowych, znany ze swojej skuteczności w oddzielaniu logiki biznesowej aplikacji od interfejsu użytkownika. Wzorzec ten dzieli aplikację na trzy główne komponenty: **Model**, **View** i **Controller**, co zapewnia modularną strukturę, ułatwiając zarządzanie, rozwój i testowanie aplikacji.

W kontekście REST-API wzorzec MVC jest adaptowany w unikalny sposób, różniący się od jego klasycznego zastosowania w aplikacjach webowych. Chociaż podstawowe komponenty MVC pozostają te same, ich rola i funkcjonalność są dostosowane do specyfiki REST-API.

Zaczniemy od Modelu. W REST-API model nadal odpowiada za reprezentację danych oraz logikę biznesową. Składają się na niego struktury danych i klasy encji, a także warstwa dostępu do danych, czyli wszystko to, co stanowi o „mózgu” aplikacji. **Model** zajmuje się przetwarzaniem żądań, takich jak operacje **CRUD** (Create, Read, Update, Delete), które są podstawą każdej interakcji z danymi. Przejdźmy teraz do Widoku. W tradycyjnym MVC, **widok** zajmuje się prezentacją danych, jednak w REST-API ta rola zostaje przeddefiniowana. Zamiast generować strony HTML, REST-API wysyła dane w formacie JSON. Oznacza to, że „widok” w tym kontekście to sposób prezentacji danych, które są łatwo konsumowalne przez aplikację kliencką (panel zarządzania) napisaną w REACT. Na koniec mamy Kontroler. W REST-API **kontrolery** są sercem interakcji, działając jako punkty końcowe dla żądań HTTP od klientów. Kontrolery przyjmują żądania, takie jak GET, POST, i koordynują przepływ informacji między widokiem a modelem. Są odpowiedzialne za przetwarzanie wejścia, wywoływanie odpowiednich operacji w modelu i zwracanie danych do klienta, pełniąc rolę kluczowego pośrednika.

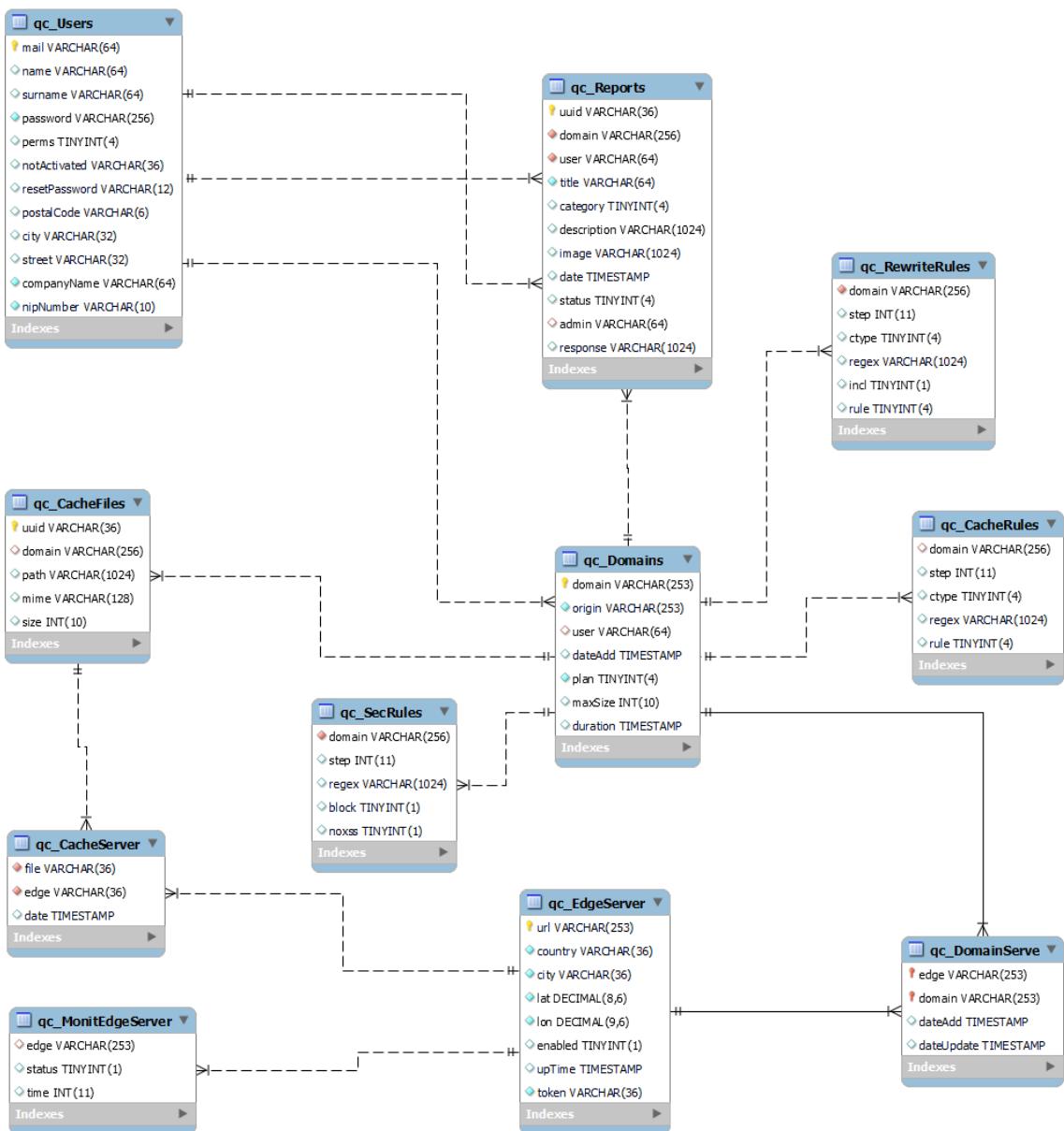


Rysunek 6.2. Struktura aplikacji w modelu MVC¹²

6.3 Schemat bazy danych

Projektowanie schematu bazy danych dla serwera centralnego zostało przeprowadzone z zachowaniem zasad normalizacji, kluczowego elementu w tworzeniu efektywnych i spójnych struktur danych. W szczególności, baza danych w znacznej większości tabel spełnia kryteria **1. postaci normalnej** (*1NF*), co zapewnia, że każdy atrybut w rekordzie jest atomowy i każdy rekord zawiera unikalne informacje. Jest to istotne, aby uniknąć problemów z redundancją i sprzecznością danych. W bazie danych każda tabela posiada jasno zdefiniowany klucz główny, co gwarantuje unikalność każdego rekordu. Dodatkowo, stosowanie kluczy obcych pozwala na utrzymanie integralności danych poprzez zdefiniowanie relacji między tabelami. Struktura bazy danych została skonstruowana tak, aby odpowiadała specyficznym potrzebom i funkcjonalnościom systemu sCDN. Każdy element – od zarządzania użytkownikami i domenami, po serwery skrajne i zasady bezpieczeństwa – został starannie zaprojektowany, aby sprostać wymaganiom systemu.

¹²Na podstawie: <https://medium.com/analytics-vidhya/implementation-of-mvc-rest-apis-in-expressjs-c7eb6a097b9f>



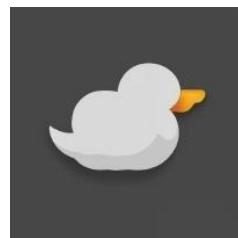
Rysunek 6.3. Schemat bazy danych serwera centralnego

Struktura tabel w bazie danych odzwierciedla kluczowe komponenty systemu. Główne elementy na które należy zwrócić uwagę podczas analizy schematu bazy danych:

- Tabela **qc_Users**: Zawiera informacje o użytkownikach, w tym adresy e-mail, hasła, uprawnienia oraz dane osobowe i firmowe. Jest to centralny punkt dla zarządzania kontami użytkowników.
- Tabela **qc_EdgeServer**: Reprezentuje serwery skrajne w sieci, zawierając informacje takie jak URL, lokalizacja geograficzna, token autoryzacyjny i inne.
- Tabela **qc_Domains**: Przechowuje informacje o zarejestrowanych domenach, wskazując ich pochodzenie, właścicieli i powiązane plany. Ta tabela jest ważna dla zarządzania domenami użytkowników i ich konfiguracją.
- Tabela **qc_DomainServe**: Tworzy powiązanie między serwerami skrajnymi a domenami, umożliwiając śledzenie, które domeny są obsługiwane przez poszczególne serwery skrajne.
- Tabele **qc_CacheRules**, **qc_RewriteRules**, **qc_SecRules**: Zawierają zasady dotyczące zapisywania lokalnej zawartości na serwerach skrajnych, przekierowań i optymalizacji treści dla każdej domeny, umożliwiając szczegółową konfigurację działania systemu.
- Tabela **qc_Reports**: Umożliwia użytkownikom zgłaszanie problemów lub pytań, przechowując informacje o zgłoszeniach, ich statusach i odpowiedziach administratorów.
- Tabela **qc_MonitEdgeServer**: Służy do monitorowania stanu serwerów skrajnych, przechowując informacje o ich statusie i czasie działania.
- Tabele **qc_CacheFiles** i **qc_CacheServer**: Zarządzają plikami przechowywanymi w cache na serwerach skrajnych, zapisując ich lokalizację, rozmiar i typ. Ta zawartość jest aktualizowana, w momencie gdy użytkownik wybierze opcję "odśwież zawartość" w panelu administracyjnym.

6.4 Identyfikacja wizualna

Elementem identyfikacji wizualnej jest chociażby logo, które jak wszystkie obrazy umieszczone w panelu, zostało stworzone przy użyciu innowacyjnych technologii AI, konkretnie przez Bing Image Creator. Generowanie obrazów wykorzystuje technologię **DALL · E**, opracowaną przez OpenAI. Jest to zaawansowany model AI, który tworzy szczegółowe, wysokiej jakości obrazy na podstawie opisów tekstowych. Ta technologia pozwala na syntezę obrazów obejmujących różne style i tematy, od realistycznych zdjęć po abstrakcyjne koncepty. Zdolność DALL · E do interpretowania i wizualizacji złożonych pomysłów czyni go niezwykle wszechstronnym narzędziem dla grafików, artystów i projektantów, otwierającym nowe możliwości w kreowaniu unikalnych i innowacyjnych wizualizacji.



Rysunek 6.4. Logo serwisu sCDN

Wykorzystanie sztucznej inteligencji do generowania loga posiada szereg zalet, między innymi AI ma zdolność tworzenia unikalnych wzorów i koncepcji, co przekłada się na oryginalność logo i wyróżnienie marki na rynku. Ważnym aspektem jest przyspieszenie procesu projektowania. AI umożliwia eksperymentowanie z różnymi stylami i kolorami, szybko dostarczając różnorodne propozycje. Dodatkowo logo stworzone przez AI, w odróżnieniu od tradycyjnych metod projektowania, może oferować większą jasność co do praw autorskich i licencji. Ponieważ jest to generowane automatycznie, ryzyko naruszenia praw autorskich istniejących wzorów jest znacznie zredukowane.

Panel zarządzania siecią CDN został zaprojektowany z wykorzystaniem popularnego motywów **SB Admin 2**, dostępnego na stronie startbootstrap.com/theme/sb-admin-2 pod licencją **MIT**. Oparty jest na **Bootstrap**, który jest jednym z najbardziej popularnych framework'ów front-endowych. Według danych na dzień 30 grudnia 2023, jest on używany przez 25,4% wszystkich stron internetowych, których bibliotekę JavaScript znamy. Stanowi to 20,8% wszystkich stron internetowych¹³.

¹³Statystyki użycia Bootstrap: <https://w3techs.com/technologies/details/js-bootstrap>

6.5 Konfiguracja serwera centralnego

6.5.1 Application.properties

Plik *application.properties* w aplikacjach Spring Boot to tekstowy plik konfiguracyjny, który stosuje format **klucz=wartość** do definiowania różnych ustawień. Jest to główny sposób na dostosowanie zachowania aplikacji Spring bez konieczności zmiany kodu. Plik ten jest ładowany podczas startu aplikacji i jest umieszczony w katalogu *src/main/resources*, co pozwala na łatwe pakowanie go razem z aplikacją. Można w nim definiować różnorodne ustawienia, takie jak: konfigurację połączenia z bazą danych, parametry serwera, ustawienia bezpieczeństwa, konfigurację serwera pocztowego i inne ustawienia zależne od potrzeb aplikacji. Klucze używane w pliku są zdefiniowane przez Spring Boot i związane z nim biblioteki. Plik ten jest szczególnie ważny w środowiskach produkcyjnych, gdzie konfiguracje, takie jak adresy URL baz danych, dane uwierzytelniające i inne zmienne środowiskowe, różnią się od tych używanych podczas rozwoju i testowania. Używając zewnętrznych plików konfiguracyjnych, można dostosować aplikację do różnych środowisk bez konieczności przebudowywania aplikacji.

```
1 spring.datasource.driver-class-name=org.mariadb.jdbc.Driver
2 spring.datasource.url = jdbc:mariadb://127.0.0.1:3306/central
3 #spring.datasource.url = jdbc:mariadb://mariadb:3306/central
4 spring.datasource.username = root
5 spring.datasource.password = 1234
```

Listing 6.1: Konfiguracja konektora JDBC

W konfiguracji *application.properties*, linie odpowiadające za ustawienia konektora JDBC są kluczowe do zarządzania komunikacją z bazą danych MariaDB. Dwie różne konfiguracje połączenia są uwzględnione: jedna zakomentowana, przeznaczona do użycia w środowisku Docker, oraz druga, aktywna, dedykowana dla połączenia w środowisku lokalnym. Zakomentowanie linii za pomocą `#` pozwala na łatwą zmianę między konfiguracjami bez konieczności wprowadzania większych modyfikacji w pliku.

```
1 jedis.host=127.0.0.1
2 jedis.port=6379
3 paypal.client.secret=#####
4 paypal.client.id=AagNrRbUPIg1268P6qfpb...
```

Listing 6.2: Konfiguracja konektora jedis oraz API PayPal

W ten sam sposób można konfigurować połączenia do innych konektorów, np. połączenia z Redis. Plik ten służy również do przechowywania innych kluczowych danych niezbędnych dla funkcjonowania systemu, takich jak klucze publiczne i prywatne wymagane do integracji z zewnętrznymi interfejsami API, na przykład w systemie płatności PayPal.

```
1 spring.mail.host=smtp.gmail.com
2 spring.mail.port=587
3 spring.mail.username=qc.scdn@gmail.com
4 spring.mail.password=#####
5 spring.mail.properties.mail.smtp.auth=true
6 spring.mail.properties.mail.smtp.starttls.enable=true
```

Listing 6.3: Konfiguracja JavaMailSender

Plik konfiguracyjny oferuje możliwość parametryzacji ustawień dla różnych bibliotek i modułów wykorzystywanych przez centralny serwer. Na przykład, za pomocą tego pliku można skonfigurować **JavaMailSender**, który jest używany do wysyłania e-maili. Dzięki temu, parametry takie jak serwer SMTP, port, dane uwierzytelniające i inne opcje związane z wysyłaniem wiadomości e-mail mogą być łatwo dostosowywane i zarządzane, zapewniając elastyczność oraz ułatwiając utrzymanie i aktualizację systemu.

6.5.2 pom.xml

Plik *pom.xml*, czyli Project Object Model, stanowi fundament każdego projektu opartego o Java i Spring Boot Framework, gdy wykorzystuje się narzędzie **Maven** do zarządzania projektami i zależnościami. Maven, jeden z kilku popularnych systemów zarządzania pakietami w świecie Java, wyróżnia się swoją prostotą i efektywnością w zarządzaniu zależnościami oraz konfiguracją projektu. Główną funkcją pliku pom.xml jest opis projektu, jego struktury, zależności, konfiguracji procesu budowania, używanych pluginów i wersji poszczególnych komponentów. Dzięki temu plikowi, Maven jest w stanie automatycznie pobrać i zintegrować potrzebne biblioteki i narzędzia, które nie są domyślnie dostarczane przez Java czy Spring.

W projekcie serwera centralnego zdecydowano się na użycie kilku dodatkowych **bibliotek**, aby wzbogacić funkcjonalność oraz ułatwić proces rozwoju aplikacji. Te dodatkowe biblioteki to: gson, jedis i lombok.

```
1 <dependencies>
2   (...)

3   <dependency>
4     <groupId>com.google.code.gson</groupId>
5     <artifactId>gson</artifactId>
6     <version>2.8.9</version>
7   </dependency>
8   <dependency>
9     <groupId>redis.clients</groupId>
10    <artifactId>jedis</artifactId>
11    <version>4.4.3</version>
12  </dependency>
13  <dependency>
14    <groupId>org.projectlombok</groupId>
15    <artifactId>lombok</artifactId>
16    <optional>true</optional>
17  </dependency>
18  (...)

19 </dependencies>
```

Listing 6.4: Fragment pom.xml integrujący dodatkowe biblioteki

W ramach projektu serwera centralnego, wykorzystano IntelliJ IDEA, który oferuje szerokie wsparcie dla Mavena, w tym **dedykowaną konsolę** do zarządzania projektem. Dzięki tej integracji, IntelliJ IDEA umożliwia łatwe i intuicyjne zarządzanie zależnościami Mavena, wykonanie poleceń budowania, czyszczenia oraz innych operacji związanych z cyklem życia projektu. Ta wygodna funkcjonalność pozwala na sprawne dodawanie, aktualizowanie i zarządzanie dodatkowymi bibliotekami, co znacznie upraszcza i przyspiesza proces rozwoju oprogramowania.

6.6 Funkcjonalności serwera centralnego

6.6.1 Wymagania funkcjonalne

- **Przekierowywanie żądań o treści**
 - Odnajdywanie najbliższego serwera skrajnego.
 - Dostarczanie klientowi aplikacji - **sandbox**.
 - Estymacja geolokalizacji klienta.
- **Zarządzanie domenami przez użytkowników**
 - Możliwość: dodania, edycji, usunięcia domeny.
 - Możliwość wysłania, usunięcia domeny na serwery skrajne.
 - Możliwość dokupienia planu PRO dla konkretnej domeny.
 - Możliwość konfiguracji reguł lokalnego zapamiętywania zawartości na serwerach skrajnych.
 - Możliwość konfiguracji reguł konwerterów.
 - Możliwość blokowania dostępu do zawartości.
 - Przeglądanie ewidencji treści na serwerach skrajnych.
- **Zarządzanie serwerami skrajnymi przez administratorów**
 - Możliwość: dodania, edycji, usunięcia, wyłączenia serwera skrajnego.
 - Możliwość: podglądu lokalizacji serwera na mapie.
 - Możliwość wyświetlenia domen zarejestrowanych na serwerze skrajnym.
 - Możliwość wysłania żądania o przywrócenie serwera skrajnego do ustawień fabrycznych.
- **Zarządzanie kontem użytkownika**
 - Możliwość edycji podstawowych danych o użytkowniku, takich jak: dane personalne, informacje adresowe oraz dane do faktur.
 - Możliwość zmiany hasła.
 - Możliwość aktywacji konta przez link z maila.
 - Możliwość wyświetlenia listy użytkowników.
 - Możliwość zmiany uprawnień dla konta użytkownika.
 - Możliwość zresetowania hasła do konta przez administratora.
 - Możliwość usunięcia konta przez administratora.

- **Zarządzanie zgłoszeniami**
 - Możliwość wysłania zgłoszenia przez użytkownika wraz z opisem swojego problemu.
 - Możliwość przeglądania publicznych, rozwiązań problemów.
 - Możliwość obsługi zgłoszeń przez administrację.
- **Zbieranie informacji o infrastrukturze**
 - Cykliczne odpytywanie i monitorowanie statusu serwerów skrajnych.
 - Możliwość pobrania i podglądu informacji konfiguracyjnych o serwerze skrajnym.
 - Możliwość zarządzania statusem serwera skrajnego (włączanie i wyłączenie).
- **Obsługa systemu płatności**
 - Możliwość zakupienia płatnego planu dla domeny.
 - Cykliczne monitorowanie stanu planów, aktualizacja ich statusu oraz informowanie użytkowników za pomocą wiadomości email o zmianach.
 - Informowanie serwerów skrajnych o zmianach planu.

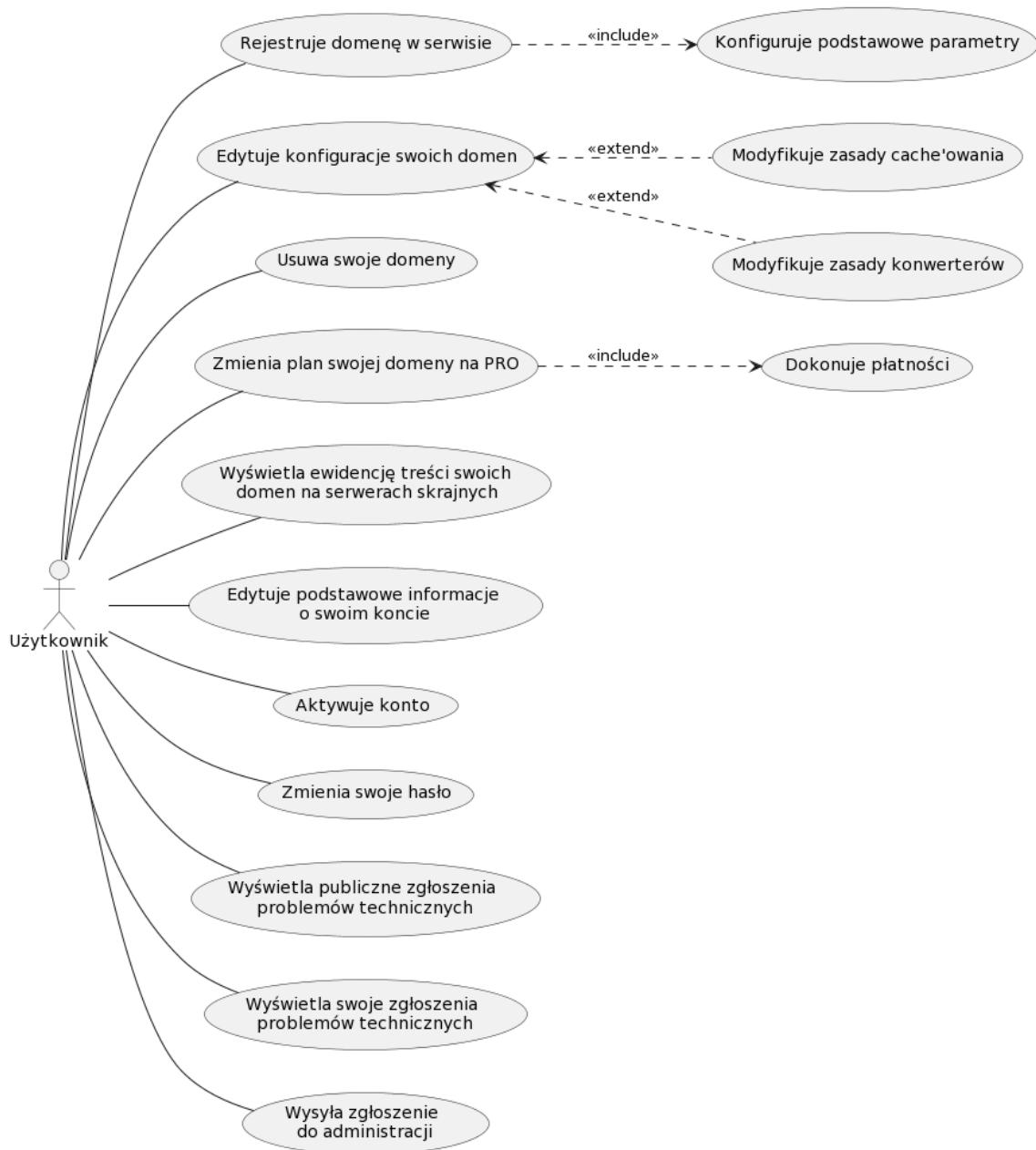
6.6.2 Wymagania niefunkcjonalne

- System powinien być dostępny i funkcjonalny przez większość czasu, minimalizując ryzyko przestojów.
- Stabilne działanie na różnych przeglądarkach internetowych i platformach.
- Informacje wrażliwe powinny być przechowywane w sposób bezpieczny (szifrowanie).
- Zapewnienie, że interfejs użytkownika jest intuicyjny, dostępny i łatwy w użyciu dla różnych grup użytkowników.
- Zachowanie integralności danych z serwerami skrajnymi.
- Wykorzystanie skalowania horyzontalnego.

6.6.3 Diagramy przypadków użycia



Rysunek 6.5. Diagram przypadków użycia - perspektywa administratora

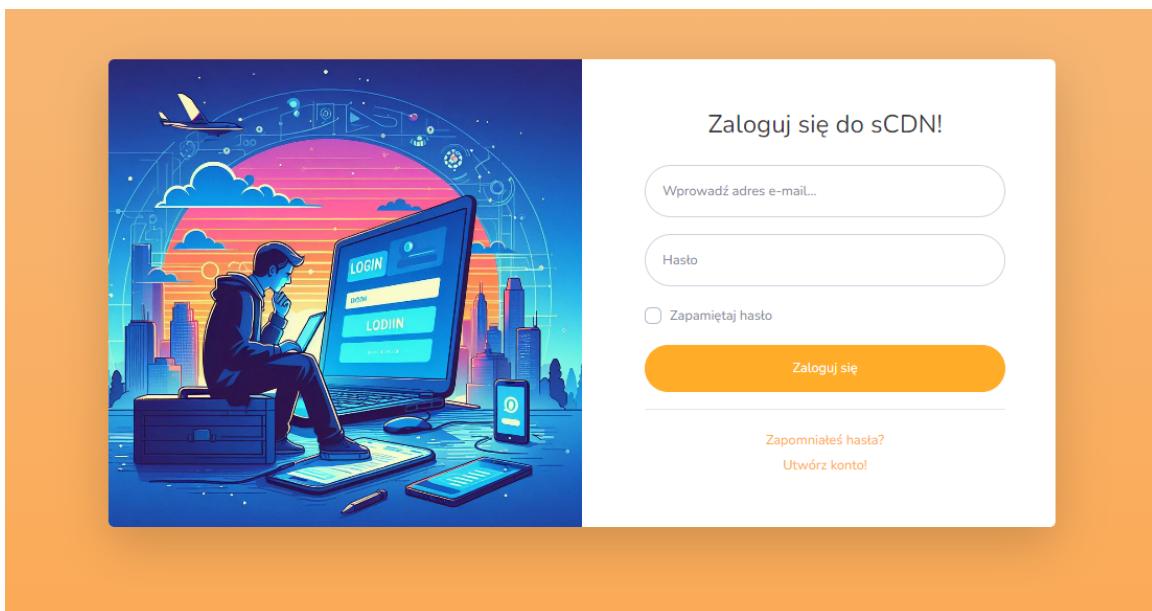


Rysunek 6.6. Diagram przypadków użycia - perspektywa użytkownika

6.7 Prezentacja funkcjonalności serwera centralnego

6.7.1 Logowanie i rejestracja

Prezentację procesu logowania do serwisu sCDN należy rozpocząć od najważniejszego elementu tego procesu, czyli **adresu email**. Jest to unikalny identyfikator dla każdego użytkownika. Wybór adresu email jako głównego środka identyfikacji wynika z jego powszechności, łatwości weryfikacji oraz bezpieczeństwa. W naszym systemie, każdy adres email może być przypisany tylko do jednego konta użytkownika. Jest to standardowa praktyka w większości systemów internetowych, mająca na celu zapewnienie unikalności i integralności danych użytkownika.



Rysunek 6.7. Logowanie do panelu zarządzania

Obiekt konta w bazie danych, reprezentowany przez tabelę **qc_Users**, został zaprojektowany, by zawierać kompleksowe dane o użytkownikach. W tej tabeli znajduje się adres email, służący jako unikalny identyfikator użytkownika. Dla zapewnienia bezpieczeństwa, hasła są przechowywane w formie zaszyfrowanej, przy użyciu **algoritmu SHA256** oraz **techniki solenia**. Status aktywacji konta jest rejestrowany w polu **notActivated**, co ma kluczowe znaczenie dla procesu autoryzacji. Tabela ta obejmuje także istotne dane, takie jak informacje adresowe oraz pole **resetPassword**, które jest używane podczas resetowania hasła użytkownika. Struktura tego obiektu jest optymalizowana pod kątem efektywnego przechowywania i szybkiego dostępu do danych, co jest kluczowe dla płynnego funkcjonowania systemu logowania.

W kontekście implementacji w **Javie**, obiekt konta użytkownika jest reprezentowany jako klasa zawierająca pola odpowiadające atrybutom w bazie danych. Ta klasa nie tylko ułatwia manipulację danymi użytkownika w aplikacji, ale również zapewnia warstwę abstrakcji, która oddziela logikę aplikacji od bezpośredniego dostępu do bazy danych.

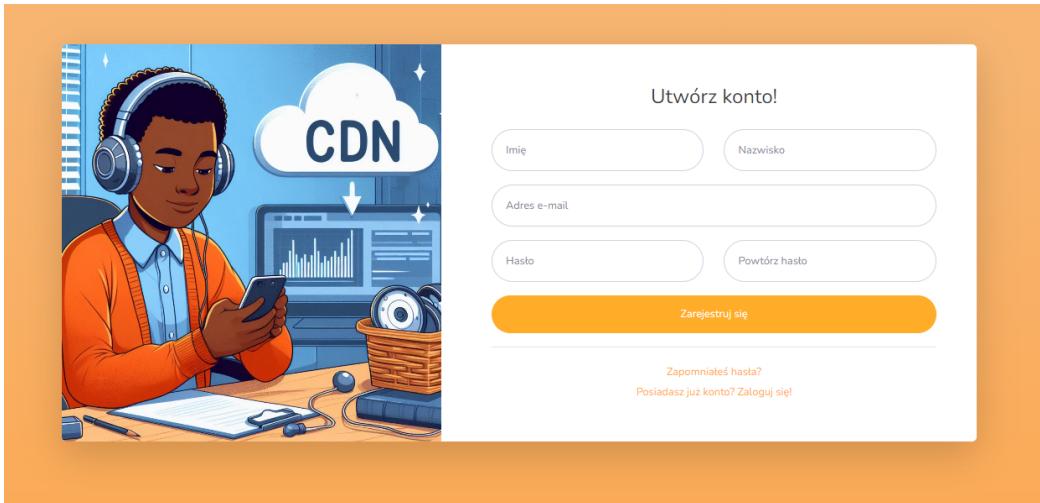
```
1  @Data
2  public class qc_Users
3  {
4      String mail;
5      String name;
6      String surname;
7      String password;
8      Integer perms;
9      String postalCode;
10     String notActivated;
11     String resetPassword;
12     String city;
13     String street;
14     String companyName;
15     String nipNumber;
16
17
18     public static qc_Users fromJSON(String body)
19     {
20         return (new Gson())
21             .fromJson(body, qc_Users.class);
22     }
23 }
```

Listing 6.5: Odzwierciedlenie rekordów z tabeli jako obiekt użytkownika w Java

Powyższy kod to definicja klasy w Java, reprezentująca użytkownika. Klasa ta zawiera atrybuty odzwierciedlające rekord z bazy danych. Warto wyróżnić dwa główne elementy: dekorator `@Data` z biblioteki **Lombok** oraz metodę `fromJSON`. Lombok to biblioteka, która jest popularnym narzędziem w ekosystemie Javy, służącym do redukcji *boilerplate code*. `@Data` jest adnotacją, która automatycznie generuje typowe metody, które zazwyczaj są ręcznie tworzone w klasach Java, takie jak: `getters`, `setters`, `toString()`, `equals()` i `hashCode()`. Wykorzystanie tej biblioteki znacznie upraszcza klasę, ponieważ eliminuje potrzebę ręcznego pisania tych metod.

Drugim rozwiązaniem, o którym należy wspomnieć, jest metoda `fromJSON()`, służy ona do tworzenia instancji tej klasy na podstawie łańcucha **JSON**. Wykorzystuje ona bibliotekę Gson, która jest popularnym narzędziem do parsowania i generowania JSON w aplikacjach Javy. Gson umożliwia mapowanie obiektów Java na ich reprezentacje JSON i odwrotnie. **JSON** (*JavaScript Object Notation*) to format wymiany informacji wykorzystywany głównie do przesyłania danych między serwerem a aplikacjami internetowymi. Jest to tekstowy format reprezentujący dane w postaci par klucz-wartość, będący jednocześnie łatwym do odczytu dla człowieka i prostym do przetwarzania dla maszyn. JSON jest niezależny od języka programowania i może być używany w połączeniu z wieloma różnymi językami.

Proces rejestracji użytkownika w serwisie jest zaprojektowany tak, aby był **intuicyjny**, a jednocześnie zapewniał odpowiedni poziom **bezpieczeństwa** i weryfikacji. Na początkowym etapie, użytkownik jest proszony o podanie swoich podstawowych danych osobowych, które obejmują imię, nazwisko, adres email oraz hasło. Po za rejestraniu, użytkownik otrzymuje dostęp do podstawowych funkcjonalności serwisu. Jednakże, pełny dostęp do wszystkich funkcji jest możliwy dopiero po przejściu przez **proces aktywacji konta**. Aktywacja konta ma na celu potwierdzenie, że podany adres email należy do rejestrującego się użytkownika. Proces ten odbywa się poprzez wysłanie na podany adres email wiadomości zawierającej link aktywacyjny. Użytkownik musi kliknąć w ten link, co jest sygnałem dla systemu, że adres email jest prawidłowy. Taki proces rejestracji i aktywacji konta jest standardową praktyką w większości serwisów internetowych. Zapewnia on odpowiednią równowagę między łatwością rejestracji a potrzebą zabezpieczenia konta i weryfikacji tożsamości użytkownika. Poprzez zastosowanie aktywacji z użyciem email, serwis minimalizuje ryzyko nadużyć i fałszywych rejestracji, jednocześnie utrzymując proces rejestracji prostym i przyjaznym dla użytkownika.



Rysunek 6.8. Panel rejestracji użytkownika

```
1  @SneakyThrows
2  @PostMapping("/api/user/add")
3  public String apiUserAdd
4  (
5      @RequestBody String body,
6      HttpServletResponse httpServletResponse,
7      @CookieValue(value = "sess", defaultValue = "") String sess
8  )
9  {
10     APIResp apiResp = new APIResp();
11     Session ses = sessionService.fromToken(sess);
12     qc_Users usr = qc_Users.fromJSON(body);
13     ...
14     if (UsersController.fetchByMail(jdbcTemplate, usr.getMail()) != null)
15         return apiResp.fail("Użytkownik o takim mail'u już istnieje");
16
17     usr.setPassword(SHA256.hashLogin(usr));
18     usr.setPerms(EnumPermissionLevel.USER.ordinal());
19     usr.setNotActivated(UUID.randomUUID().toString());
20
21     if (!UsersController.insert(jdbcTemplate, usr))
22         return apiResp.fail("Wystąpił błąd, proszę spróbować ponownie później");
23
24     GmailActivate.send(gmailService, usr);
25     return apiResp.success();
26 }
```

Listing 6.6: Proces rejestracji

Zaprezentowany fragment kodu służy do obsługi dodawania nowego użytkownika, reaguje na żądania POST na ścieżce `/api/user/add`. Kluczowym krokiem jest deserializacja ciała żądania JSON do obiektu `qc_Users` przy użyciu metody `fromJSON(body)`. Następnie następuje weryfikacja, czy wszystkie wymagane dane użytkownika są obecne i poprawne. Jeśli jakiekolwiek dane są niekompletne lub użytkownik o podanym adresie email już istnieje, metoda zwraca komunikat o błędzie. Po pomyślnej walidacji danych, hasło użytkownika jest haszowane. Ustawiane są również domyślne uprawnienia użytkownika i generowany jest unikalny identyfikator do aktywacji konta. W kolejnym kroku, użytkownik jest dodawany do bazy danych. Jeśli ta operacja się nie powiedzie, zwracany jest odpowiedni komunikat o błędzie. Na koniec, jeśli użytkownik został pomyślnie dodany, wysyłany jest do niego email aktywacyjny za pomocą serwisu **GmailActivate**.

```
1  public class GmailActivate extends GmailMail
2  {
3      public static void send (GmailService gmailService, qc_Users user)
4      {
5          String ht = load("login");
6          ht = ht.replaceAll("%s_name", user.getName());
7          ht = ht.replaceAll("%s_surname", user.getSurname());
8          ht = ht.replaceAll("link_do_aktywacji",
9          String.format("http://cdn.localhost/api/user/activate/%s",
10             ↳ user.getNotActivated())
11      );
12
13      prep(gmailService, user.getMail(), "Aktywacja konta", ht);
14  }
```

Listing 6.7: Wysłanie maila aktywacyjnego



Dzień dobry, Maciej Bandura,

Dziękujemy za rejestrację w serwisie sCDN! Jesteśmy podekscytowani, że dołączyłeś/aś do naszej społeczności.

Aby aktywować swoje konto i czerpać pełne korzyści z naszych usług, prosimy kliknąć poniższy przycisk:

[Aktywuj konto](#)

Jeśli nie możesz kliknąć powyższego linku, skopiuj i wklej poniższy adres URL do przeglądarki:

<http://cdn.localhost/api/user/activate/c5a97c8a-7f6c-4e55-a5e5-a229a93ad0d8>

Dziękujemy jeszcze raz za dołączenie do sCDN. Jeśli masz jakiekolwiek pytania lub potrzebujesz pomocy, nie wahaj się skontaktować z naszym zespołem wsparcia pod adresem qc.scdn@gmail.com.

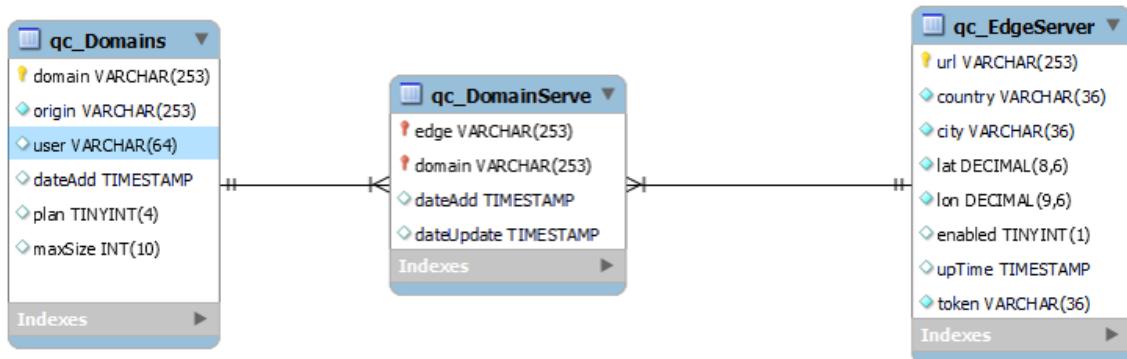
qc - sCDN 2023 © Maciej Bandura, Marcin Ślusarczyk | [Strona](#) | [Panel zarządzania](#)

Rysunek 6.9. Email aktywacyjny

Klasa **GmailActivate**, która jest rozszerzeniem klasy **GmailMail**, zawiera metodę **send**, służącą do wysyłania emaili aktywacyjnych do użytkowników. Metoda ta przyjmuje dwa parametry: **GmailService gmailService** i **qc_Users user**. GmailService to serwis używany do wysyłania emaili, natomiast qc_Users user to obiekt reprezentujący użytkownika, któremu ma zostać wysłany email. W ciele metody send, najpierw ładowana jest treść szablonu emaila za pomocą metody **load** z klasy **GmailMail**. Metoda ta ładuje szablon HTML, w przypadku maila aktywacyjnego, jest to plik o nazwie "*activate*", znajdujący się w folderze z zasobami serwera. Następnie, w szablonie tym, dokonywana jest podmiana kluczowych fragmentów tekstu, aby spersonalizować email. Podobnie jak w przypadku innych emaili wysyłanych przez system, konsekwentnie używa się wspólnego szablonu, który zapewnia spójność nagłówka i stopki w każdej wiadomości. Podmieniana jest jedynie centralna zawartość maila (div **content**). Końcowy efekt to email, który jest jednocześnie **spersonalizowany** (dzięki treści specyficznej dla danego użytkownika) i **spójny** (dzięki ujednolicenemu wygląowi zapewnionemu przez szablon).

6.7.2 Rejestracja domeny w serwisie

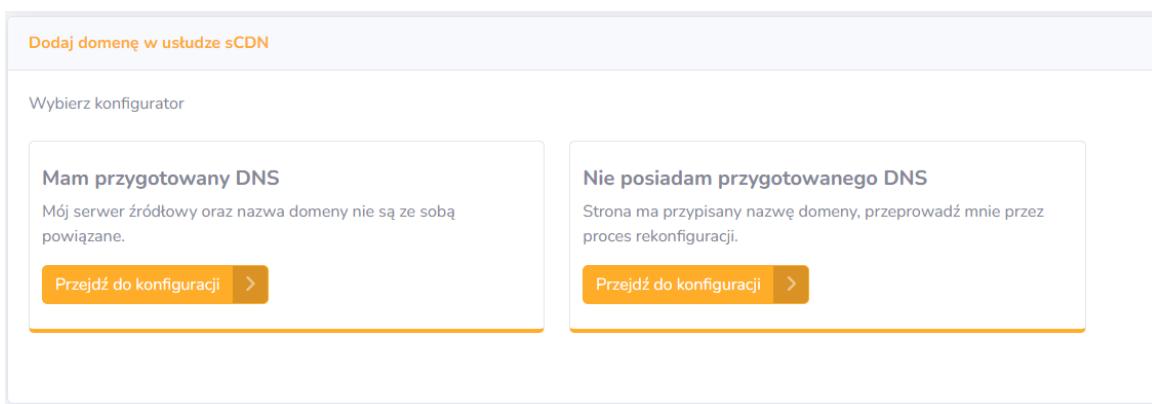
Rejestracja domeny w serwisie sCDN jest kluczowym procesem dla użytkowników serwisu. Domena w sCDN to nie tylko nazwa wskazująca na serwer centralny, ale również zestaw konfiguracji i ustawień, które definiują sposób dystrybucji i optymalizacji treści strony internetowej. W sCDN jedna domena może być zarejestrowana na wielu serwerach skrajnych. Z drugiej strony, jeden serwer skrajny może obsługiwać wiele różnych domen. Taka relacja jest oparta na tabeli **qc_DomainServe** będącą swoistym łącznikiem domeny z serwerami skrajnymi.



Rysunek 6.10. Relacja pomiędzy domeną a serwerem skrajnym

Proces rejestracji domeny składa się z dwóch głównych sposobów dodania domeny. Zależnie od tego, czy posiada on już skonfigurowany DNS, czy też nie.

- 1. Rejestracja z gotowym DNS:** W tym przypadku, użytkownik podaje nazwę domeny i adres serwera źródłowego.
- 2. Rejestracja bez skonfigurowanego DNS:** W tym scenariuszu, użytkownik dodaje domenę, ale nie dysponuje jeszcze pełnym DNS. Po wprowadzeniu podstawowych informacji, serwer automatycznie wyszukuje adres serwera źródłowego. Po poprawnym odnalezieniu adresu jest on ustawiany dla dodawanej domeny. Jeżeli jednak nie uda się odnaleźć adresu serwera źródłowego, użytkownik otrzymuje instrukcję, w jaki sposób skonfigurować DNS aby był on kompatybilny z siecią sCDN. Jest to konfigurator zbudowany na podstawie istniejącego już rozwiązania dystrybuowanego przez firmę Cloudflare. Zakłada, że użytkownik tylko przełączy rekord w DNS na serwer centralny po dokonaniu wstępnej konfiguracji.



Rysunek 6.11. Dostępne konfiguratory dodania domeny

Kolejnym krokiem jest wybór optymalizacji. W zależności od potrzeb, użytkownik może wybrać spośród różnych poziomów optymalizacji, które oferują różnorodne kombinacje usług. Obejmuje to pełne pokrycie lokalnych kopii strony, różne strategie tunelowania zapytań, a także różne narzędzia do optymalizacji treści, takie jak minifyery czy kompresja obrazów. Każdy z tych poziomów optymalizacji jest dostosowany do różnych wymagań i celów użytkowników, od tych poszukujących kompleksowej optymalizacji i wydajności, po tych, którzy potrzebują tylko podstawowej dystrybucji treści bez dodatkowych narzędzi optymalizacyjnych.

Zarejestruj swoją domenę w qc-sCDN

Nazwa domeny: strona.pl

Serwer źródłowy: http://127.0.0.1/

Typ zawartości:

Statyczna strona
 Statyczny kontent

Rodzaj optymalizacji: Pełna optymalizacja

Petna optymalizacja strony polega na:

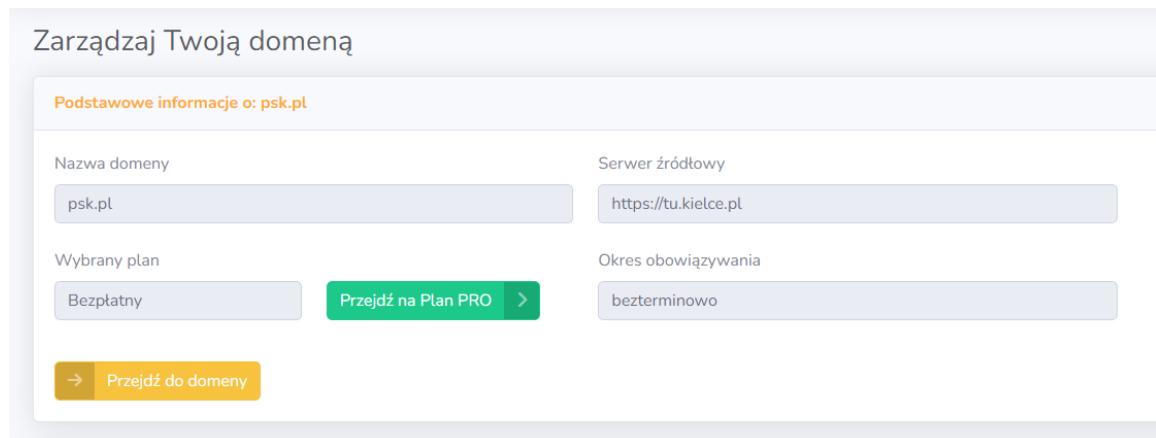
- Pełne pokrycie lokalnych kopii strony
- Tunelowanie tylko zapytań POST, PUT, DELETE
- Pobranie zawartości tylko przy pierwszym zapytaniu na serwerze
- Optymalizacja zdjęć
- Wymuszanie formatu zdjęć.webp
- Minifier JS
- Minifier CSS
- Minifier HTML

Dodaj domenę

Rysunek 6.12. Formularz rejestracji domeny z wyborem poziomu optymalizacji

6.7.3 Konfiguracja domeny

Panel zarządzania domeną w serwisie sCDN zapewnia użytkownikom między innymi dostęp do podstawowych informacji o ich domenie. W pierwszej sekcji, użytkownicy mogą znaleźć dane takie jak nazwa domeny, adres serwera źródłowego oraz typ wybranego planu subskrypcji. Stąd również mają możliwość zmiany planu na płatny.



Rysunek 6.13. Panel zarządzania domeną

Użytkownik ma możliwość zarejestrowania domeny na serwerze skrajnym. Operacja ta jest realizowana w sekcji *domena na serwerach skrajnych*. Znajduje się tu lista aktualnie dostępnych serwerów skrajnych zawierająca: możliwość rejestracji domeny (1), funkcję wyświetlenia domeny na wybranym serwerze skrajnym (2), resetowanie plików dotyczących wybranej witryny (3), możliwość wyrejestrowania - usunięcia domeny z wybranego serwera skrajnego (4) oraz funkcję grupowych operacji (5). W celu przyspieszenia procesu zarządzania domeną, użytkownik może wykonać pewne działania na wszystkich serwerach skrajnych.

Domena psk.pl na serwerach skrajnych				
Serwer skrajny	Data dodania	Ostatnia aktualizacja	Operacje	
http://edge3.j.pl	2023-12-26 21:51	2023-12-26 21:51		
http://edge6.j.pl	2023-12-26 21:51	2023-12-26 21:51		
http://edge7.j.pl	2023-12-26 21:54	2023-12-26 21:54		
https://edge1.000webhostapp.com	2023-12-26 21:01	2023-12-26 21:13		
https://edge2.5v.pl	2023-12-26 21:11	2023-12-26 21:13		
https://edge4.atwebpages.com	2023-12-26 21:56	2023-12-26 21:56		
https://edge8.qc-cdn.pl	2023-12-26 22:18	2023-12-26 22:18		
http://daleszyce.cdn.localhost				
https://edge5.cloudns.biz				
				1
+ Dodaj do wszystkich Wyczyszczać cache Usuń ze wszystkich				

Rysunek 6.14. Sekcja informacji o domenie

Istotnym elementem panelu zarządzania domeną sCDN jest moduł **reguły cache**. Użytkownicy definiują, jakie elementy ich strony internetowej powinny być zapisywane w pamięci lokalnej serwerów skrajnych. W tej sekcji, użytkownicy mogą korzystać z dwóch głównych selektorów do określania, które elementy mają być cache'owane: **ścieżki** (umożliwia określenie konkretnych ścieżek URL na stronie) oraz **Content-Type** (pozwala na wybór typów danych na podstawie MIME type). Użytkownicy mogą stosować wyrażenia regularne (regex) w polu reguły, aby precyzyjnie określić, które elementy pasują do danej zasady cache. To zapewnia dużą elastyczność w definiowaniu polityki cache. Istotnym aspektem zarządzania regułami cache jest ich priorytet - im zasada znajduje się niżej, tym jest traktowana jako ważniejsza. Pozwala to na nadpisywanie zasad, co jest szczególnie przydatne w sytuacjach, gdy różne reguły mogą się ze sobą pokrywać lub gdy chcemy określić wyjątki od ogólnych zasad.

Reguły cache				
Priorytet	Źródło	Regex / wyrażenie regularne ścieżki	Czy zastosować cache?	Usuń
~ ~	Ścieżka	/*	NIE	
~ ~	Content-Type	image/*	TAK	
~ ~	Content-Type	text/css*	TAK	
+ Dodaj regułę Zapisz zmiany				

Rysunek 6.15. Sekcja reguł cache

W sekcji reguł konwertera w panelu zarządzania domeną sCDN, użytkownicy mają możliwość precyzyjnego kontrolowania działania konwerterów na konkretnych zasobach. Ta sekcja umożliwia zarówno włączenie, jak i wyłączenie działania określonych konwerterów, co pozwala na bardziej szczegółową optymalizację treści. Podobnie jak w przypadku reguł cache, reguły konwerterów są definiowane na podstawie typu źródła (takiego jak ścieżka URL lub typ MIME) oraz wyrażeń regularnych (regex), które pozwalają na precyzyjne określenie zasobów objętych działaniem konwertera. System priorytetyzacji działa jak w powyższym przypadku. Użytkownicy mają do wyboru pięć różnych konwerterów. Szczegółowe opisy działania tych konwerterów oraz ich zastosowanie znajdą się w rozdziale poświęconym serwerom skrajnym.

Zarządzaj działaniami dla konkretnych zasobów					
W tej sekcji masz możliwość definiowania szczegółowych działań dla różnych rodzajów zasobów na swojej domenie. Dla wybranych ścieżek lub typów plików. Istnieje możliwość indywidualnie ustawiać minifikację dla zdjęć, CSS oraz JS, co pozwala na elastyczne zarządzanie procesem kompresji dla poszczególnych typów zasobów. Szczegółowy opis poszczególnych funkcji znajdziesz w zakładce pomoc . Uwaga: Starannie dostosuj te ustawienia, aby zoptymalizować wydajność strony, jednocześnie zachowując integralność treści.					
Priorytet	Źródło	Regex / wyrażenie regularne ścieżki	Zasada	Włącz	Usuń
	Content-Type	text/html*	HT_SANDBOX	<input checked="" type="checkbox"/>	
	Content-Type	text/html*	URL_REPLACE	<input checked="" type="checkbox"/>	
	Content-Type	text/css*	URL_REPLACE	<input checked="" type="checkbox"/>	
	Content-Type	application/javascript*	URL_REPLACE	<input checked="" type="checkbox"/>	
^ ^	Content-Type	image/*	IMG_MIN	<input checked="" type="checkbox"/>	
^ ^	Content-Type	text/css*	CSS_MIN	<input checked="" type="checkbox"/>	
^ ^	Content-Type	application/javascript*	JS_MIN	<input checked="" type="checkbox"/>	
+ Dodaj regułę Zapisz zmiany					

Rysunek 6.16. Sekcja reguł konwertera

W sekcji zarządzania zasadami bezpieczeństwa w sCDN, użytkownicy mają możliwość zastosowania środków mających na celu ochronę swoich zasobów internetowych. Ta sekcja umożliwia wprowadzenie reguł, które mogą zablokować dostęp do określonych treści na stronie. Pierwszą dostępną opcją jest blokada dostępu do danych treści znajdujących się pod konkretną ścieżką. Ta funkcja pozwala użytkownikowi na wyłączenie dostępu do specyficznych zasobów, które mogą być uznane za wrażliwe, niebezpieczne lub niepożądane. Przykładowo, jeśli na stronie znajduje się zawartość, która nie powinna być publicznie dostępna, użytkownik może zablokować do niej dostęp, zwiększać w ten sposób bezpieczeństwo swojej witryny (np. wyłączenie dostępu do panelu administratora (**wp-admin**) w systemach opartych na Wordpress - użytkownicy zewnętrzni nie potrzebują dostępu do panelu zarządzania witryną). Mechanizm ten dostarcza dodatkową warstwę ochrony. W tej sekcji również dostępny

jest opcja blokowania ataków typu **Cross-Site Scripting** (XSS), która usuwa potencjalnie szkodliwe skrypty z **parametrów GET** w żądaniach HTTP.

Ostatnia sekcja w panelu zarządzania domeną w sCDN jest bardziej informacyjna i skierowana na wyświetlenie zawartości przechowywanej na serwerach skrajnych. Ta część interfejsu użytkownika zapewnia szczegółowy wgląd w pliki i zasoby związane z wybraną domeną, które są przechowywane na różnych serwerach skrajnych. Użytkownicy mogą przeglądać listę wszystkich plików, z dokładnym wskazaniem, na którym serwerze skrajnym dany plik się znajduje. Główną funkcją tej sekcji jest umożliwienie użytkownikom łatwego dostępu do szczegółowych informacji o każdym pliku. Po wybraniu konkretnego pliku, użytkownicy mają możliwość przeglądania jego szczegółów i, co ważniejsze, mogą rekonfigurować zasady dotyczące jego przechowywania i dostępu. Obejmuje to dostosowywanie zasad zapisywania zawartości, reguł konwertera oraz reguł blokowania. Sekcja ta jest szczególnie przydatna dla mniej zaawansowanych użytkowników, którzy mogą napotykać trudności w stosowaniu selektorów i wyrażeń regularnych (regex) do zarządzania zawartością. Zamiast polegać na skomplikowanych regułach regex, użytkownicy mogą wykorzystać intuicyjny interfejs graficzny, gdzie za pomocą kilku kliknięć mogą dostosować poszczególne reguły dla każdego pliku. Ta funkcjonalność sprawia, że zarządzanie treścią na serwerach skrajnych staje się bardziej dostępne i mniej złożone technicznie.

The screenshot displays the 'Content' section of the sCDN domain management interface. At the top, there's a header bar with a search field and a 'Wyszukaj' button. Below it is a table showing file details:

Ścieżka	Typ MIME	Rozmiar	Wystąpienia	Szczegóły
/wp-content/plugins/contact-form-7/includes/css/styles.css?ver=5.8.5	text/css	2.37 KB	1	🔗 🕒 🕒
/wp-includes/js/dist/vendor/wp-polyfill.min.js?ver=3.15.0	application/javascript	115.13 KB	5	🔗 🕒 🕒

Below the table are three panels:

- Serwery skrajne**: A table showing external servers with their last update times:

Serwer	Ostatnia aktualizacja
http://edge3.j.pl	2024-01-01 21:51
http://edge6.j.pl	2024-01-01 21:51
http://edge7.j.pl	2024-01-01 21:51
https://edge4.atwebpages.com	2024-01-01 21:51
https://edge8.qc-cdn.pl	2024-01-01 21:52
- Lista reguły**: A table showing rules with 'Dodaj' and 'Blokuj' buttons:

Nazwa	Operacje
URL Rewrites	Dodaj Blokuj
Minifier js'a	Dodaj Blokuj
- Firewall**: A panel with a checkbox for 'Wyłącz dostęp do zasobu na wszystkich serwerach skrajnych' and 'Zezwalaj' and 'Blokuj' buttons.

Rysunek 6.17. Pliki domeny na serwerach skrajnych

6.7.4 Zarządzanie serwerami skrajnymi

Zarządzanie serwerami skrajnymi w programowej sieci CDN to funkcjonalności dostępne wyłącznie dla administratorów sieci. Jedną z kluczowych operacji jest możliwość dodania nowego serwera skrajnego do sieci sCDN. Proces ten rozpoczyna się od wypełnienia formularza, który wymaga podania następujących informacji:

- **URL serwera skrajnego:** adres internetowy nowego serwera, który będzie częścią sieci sCDN.
- **Położenie geograficzne:** szerokość i długość geograficzna lokalizacji serwera.
- **Token autoryzacyjny:** klucz służący do weryfikacji i autoryzacji zapytań na serwerze w sieci sCDN.

Opcjonalnie, administrator może również podać dodatkowe informacje, takie jak kraj i miasto, w którym serwer się znajduje. Te dane mogą być wykorzystane do lepszego zrozumienia geograficznego rozlożenia infrastruktury. Po weryfikacji i potwierdzeniu poprawności podanych danych, nowy serwer skrajny jest dodawany do sieci sCDN. Od tego momentu, serwer staje się dostępny dla użytkowników, którzy mogą rejestrować na nim swoje domeny. Dzięki temu, sieć sCDN jest w stanie dynamicznie rosnąć i adaptować się do zmieniających się potrzeb użytkowników oraz zwiększającego się ruchu w sieci.

The form consists of several input fields and a status indicator. At the top, there's a button labeled "Dodaj nowy serwer skrajny". Below it, there are two main columns of fields. The left column contains "URL serwera skrajnego" (Edge server URL) and "Kraj" (Country). The right column contains "Data uruchomienia" (Launch date), "Miasto" (City), "Lat" (Latitude), and "Lon" (Longitude). Below these, there's a section for the "Token autoryzacyjny" (Authorization token) and a "Stan serwera" (Server status) field containing a checked checkbox labeled "Włączony" (Enabled). At the bottom, there's a green "Dodaj" (Add) button with a checkmark icon.

Rysunek 6.18. Formularz dodania nowego serwera skrajnego do sieci

Administrator ma dostęp do panelu, zawierającego listę wszystkich serwerów skrajnych w sieci wraz z kluczowymi informacjami o każdym z nich. W tej liście zawarte są dane takie jak: adres serwera skrajnego, informacje lokalizacyjne, data uruchomienia, a także aktualny status serwera, informujący o jego działaniu lub ewen-

tualnych problemach. Z poziomu tej listy, administrator ma możliwość wykonywania różnych operacji i kontroli nad serwerami skrajnymi. Może wyświetlić lokalizację serwera na mapie (1). Istnieje także opcja sprawdzenia połączenia z serwerem (2), co jest kluczowe dla zapewnienia jego prawidłowego funkcjonowania. W przypadku, gdy serwer nie odpowiada, administrator otrzymuje odpowiedni komunikat, co pozwala na szybką reakcję i rozwiązywanie problemów. Jeśli serwer działa poprawnie, administrator ma dostęp do szczegółowych informacji debugerskich, w tym spisu konfiguracji serwera. Ponadto, administrator może przejść do edycji konkretnego serwera, co pozwala na dokonywanie niezbędnych zmian w ustawieniach i konfiguracji (3). Istnieje również opcja wyłączenia serwera (4), co może być używane w sytuacjach wymagających konserwacji lub gdy występują problemy techniczne.

```

1      const [list, setList] = useState([]);
2      useEffect(() => {
3          fetch(`$__server__/api/edge/list/domains`, { ... })
4              .then((response) => response.json()).then((resp) => {
5                  ...
6                  setList(resp.data);
7              });
8      }, []);
9
10     return (
11         ...
12         <tbody>
13             {list.map((dom) => (
14                 <tr>
15                     <td>{dom.domain}</td>
16                     <td>{dom.user}</td>
17                     <td>{dom.plan == 0 ? 'Darmowy' : 'Płatny'}</td>
18                     <td>{dom.maxSize == null ? 'Niedostępne' : `${convertBytes(dom.maxSize)} MB`}</td>
19                     <td className="d-flex justify-content-around">
20                         ...
21                         </td>
22                     </tr>
23                 )));
24         </tbody>
25         ...
26     )

```

Listing 6.8: Wyrenderowanie listy serwerów skrajnych

Ten fragment kodu to część komponentu React, który renderuje listę domen w panelu administratora sieci sCDN. Używa on stanu **list** do przechowywania informacji o domenach, które są pobierane z serwera za pomocą funkcji **fetch** wewnątrz hooka **useEffect**. Dane są pobierane z endpointu API **/api/edge/list/domains** i następnie zapisywane w stanie **list** po ich konwersji na format JSON. Kod ten korzysta z metody **map**, aby wygenerować wiersze tabeli (**<tr>**) dla każdej domeny zawartej w stanie list.

Kod ten jest typowym przykładem renderowania dynamicznych danych w aplikacji sCDN w panelu zarządzania, gdzie dane są pobierane z API, przechowywane w stanie komponentu i wyświetlane w formie tabeli w interfejsie użytkownika.

Lista serwerów skrajnych					
Adres serwera	Kraj	Miasto	Data uruchomienia	Status	
http://daleszyce.cdn.localhost	Polska	Daleszyce	2023-12-27 20:46	Włączony	
http://edge3.j.pl	Holandia	Amsterdam	2023-12-23 20:56	Włączony	
http://edge6.j.pl	Belgia	Antwerpia	2023-12-26 22:51	Włączony	
http://edge7.j.pl	Holandia	Groningen	2023-12-26 22:54	Włączony	
http://kielce.cdn.localhost	Polska	Kielce	2023-12-13 19:49	Wyłączony	
https://edge1.1000webhostapp.com	Niemcy	Wallen	2023-12-26 21:52	Włączony	
https://edge2.5v.pl	Francja	Lille	2023-12-23 20:56	Włączony	
https://edge4.atwebpages.com	Bulgaria	Sofia	2023-12-13 19:52	Włączony	
https://edge5.cloudns.biz	Polska	Olsztyn	2023-12-13 19:52	Włączony	
https://edge8 qc-cdn.pl	Polska	Warszawa	2023-12-26 22:17	Włączony	

Rysunek 6.19. Lista serwerów skrajnych dostępnych w sieci sCDN

Panel edycji serwera skrajnego jest podzielony na kilka sekcji, z których każda umożliwia wykonanie różnych operacji. W pierwszej sekcji (1), administrator ma możliwość rekonfiguracji podstawowych parametrów serwera. Obejmuje to aktualizację lokalizacji serwera oraz tokena autoryzacyjnego, co jest istotne dla prawidłowego działania i zabezpieczenia serwera w ramach sieci sCDN. Kolejna sekcja (2) to spis wszystkich zarejestrowanych domen na wybranym serwerze skrajnym. Daje to administratorowi przegląd domen korzystających z tego serwera. Sekcja oznaczona cyfrą 3 pozwala na przywrócenie serwera skrajnego do stanu fabrycznego. Ta funkcja obejmuje zresetowanie bazy danych serwera, co może być użyteczne w przypadku problemów technicznych lub konieczności ponownej konfiguracji serwera. Ostatnia sekcja, umożliwia administratorowi usunięcie serwera skrajnego z sieci. Jest to działanie **nie-odwracalne** i skutkuje usunięciem wszystkich danych i informacji powiązanych z danym serwerem, w tym zawartości lokalnie przechowywanej na serwerze.

Edycja serwera: http://daleszyce.cdn.localhost

1

URL serwera skrajnego	Data uruchomienia
http://daleszyce.cdn.localhost	2023-12-27 20:46
Kraj	Miasto
Polska	Daleszyce
Lat	Lon
50.803611	20.807500
Token autoryzacyjny	Stan serwera
[REDACTED]	<input checked="" type="checkbox"/> Włączony

Zapisz zmiany

2

Obsługiwane domeny na serwerze skrajnym

Domena	Użytkownik	Wybrany plan	Wykorzystane zasoby	Operacje
repulse.pl	[REDACTED]	Platny	6.68 MB / 100MB	
psk.pl	[REDACTED]	Bezplatny	699.54 KB / 10MB	
test.com	[REDACTED]	Platny	Niedostępne	

3

Reset stanu serwera skrajnego

Zresetowanie serwera skrajnego powoduje usunięcie całej zawartości jaką dotychczas dany serwer posiadał. Baza danych zostanie przywrócona do fabrycznych wartości (wszystkie dane zostaną usunięte). Operacja ta jest nieodwracalna, co oznacza, że wszystkie dane i treści związane z tym serwerem skrajnym zostaną trwale usunięte.

Uwaga: Przed zresetowaniem serwera skrajnego upewnij się, że jest to niezbędne.

Zresetuj stan serwera

4

Usunięcie

Usunięcie serwera skrajnego może prowadzić do destabilizacji sieci. Operacja ta jest nieodwracalna, co oznacza, że wszystkie dane i treści związane z tym serwerem skrajnym zostaną trwale usunięte.

Uwaga: Przed usunięciem serwera skrajnego upewnij się, że jest to niezbędne.

Usuń serwer

Rysunek 6.20. Panel edycji serwera skrajnego

6.7.5 Komunikacja z użytkownikami - raportowanie błędów

W systemie sCDN użytkownicy mają możliwość raportowania napotkanych błędów lub proszenia o pomoc w kwestiach związanych z usługami. Każde zgłoszenie, które użytkownik może wysłać do administracji, dotyczy konkretnej domeny. W zgłoszeniu zawarte są następujące informacje:

- **Tytuł zgłoszenia:** Krótki opis problemu lub zapytania.
- **Rodzaj zgłoszenia:** Użytkownik wybiera z dostępnych kategorii, takich jak:
 - Problemy z dostępnością usług CDN,
 - Problemy z optymalizacją treści,
 - Błędy konfiguracyjne domeny,
 - Problemy z wydajnością CDN,
 - Zapytania dotyczące faktur i płatności.

- **Opis problemu:** Szczegółowy opis sytuacji, problemu lub pytania.

Po wysłaniu, zgłoszenie otrzymuje status „**wysłano**” i staje się widoczne dla administratorów systemu sCDN. Administratorzy mają możliwość przeglądania zgłoszeń, a następnie mogą je odrzucić lub rozwiązać. W przypadku, gdy administrator uznaje zgłoszenie za ważne i mające szersze znaczenie, może ono zostać udostępnione w sekcji „*popularne problemy*”, co umożliwia innym użytkownikom zapoznanie się z rozwiązaniami podobnych kwestii.

W systemie należy wyróżnić następujące statusy zgłoszeń:

- **Wysłano:** oznacza, że zgłoszenie zostało pomyślnie wysłane do systemu i oczekuje na przegląd przez zespół administracyjny. W tym stanie zgłoszenie jest zarejestrowane w systemie, ale jeszcze nie zostało przypisane do odpowiedzialnej osoby.
- **Oczekuje:** status ten wskazuje, że zgłoszenie zostało przypisane do odpowiedniego administratora i oczekuje na rozwiązanie. Może to oznaczać, że zgłoszenie jest aktualnie w trakcie przeglądu lub wymaga dodatkowych informacji.
- **Rozwiążano:** zgłoszenie zostało zakończone i problem został rozwiązany przez zespół wsparcia. Ten status oznacza, że na zgłoszenie udzielono odpowiedzi lub podjęto niezbędne działania, aby rozwiązać zgłoszony problem.
- **Rozwiążano (publiczne):** oznacza, że zgłoszenie zostało rozwiązane, a jego szczegóły zostały udostępnione publicznie. Jest to stosowane w przypadku, gdy rozwiązanie problemu może być przydatne dla innych użytkowników systemu, którzy mogą napotkać podobne kwestie.

Administrator dysponuje trzema zakładkami, które umożliwiają efektywne zarządzanie zgłoszeniami. Pierwsza zakładka, "Nowe zgłoszenia", zawiera listę zgłoszeń oznaczonych jako "**Wysłane**", które nie zostały jeszcze przypisane do konkretnego administratora. Zgłoszenia te są świeże i wymagają podjęcia działań. Administrator ma tutaj możliwość przeglądania tych zgłoszeń i podejmowania się ich rozpatrzenia. W momencie, gdy administrator wybierze dane zgłoszenie, jest mu ono automatycznie przypisywane i zmienia status na "**Oczekuje**".

W kolejnej zakładce, "Podjęte zgłoszenia", znajdują się problemy, nad którymi administrator obecnie pracuje. Ta sekcja daje mu szybki dostęp do bieżących zadań i umożliwia efektywne zarządzanie procesem rozwiązywania problemów.

Ostatnia zakładka, "Rozwiążane zgłoszenia", to miejsce, gdzie administrator może przeglądać historię zgłoszeń, które zostały już rozwiązane. Jest to użyteczne do śledzenia wykonanej pracy, analizowania często występujących problemów oraz jako archiwum rozwiązań, które mogą być przydatne w przyszłości.

Twoje zgłoszenia						
Domena	Tytuł	Data zgłoszenia	Status	Osoba odpowiedzialna	Odpowiedź	Szczegóły
test.com	Kiedy orzymam fakturę za miesiąc grudzień?	2024-01-02 18:06	Rozwiążano	admin@admin.com	Fakturę otrzyma Pan na m	
strona.pl	Mój adres strony przenosi na normalną stronę	2024-01-02 18:02	Rozwiążano	admin@admin.com	Prawdopodobnie rekordy D	
maciej-bandura.j.pl	Problem z reklamami na stronie	2024-01-02 17:31	Odrzucono	admin@admin.com	Należy zresetować cache na wszystkich serwerach skrajnych	

Rysunek 6.21. Sekcja rozwiązań zgłoszeń

Zgłoszenia w systemie sCDN, chociaż nie są bezpośrednio związane z podstawową funkcją sieci CDN, czyli dostarczaniem treści, odgrywają kluczową rolę dla ogólnego działania serwisu. Świadczenie usług CDN wykracza poza samą dystrybucję treści, obejmując również zapewnienie odpowiedniego zaplecza technicznego. W związku z tym powyższy proces wymaga: właściwego zarządzania infrastrukturą, optymalizacji wydajności, skutecznego rozwiązywania problemów technicznych poprzez obsługę zapytań klientów. Zgłoszenia te są więc niezbędne dla utrzymania wysokiej jakości usług i zapewnienia ich niezawodności, co jest kluczowe dla satysfakcji użytkowników i sukcesu serwisu sCDN.

6.7.6 Zarządzanie kontem użytkownika

Podobnie jak zarządzanie zgłoszeniami, zarządzanie kontem użytkownika jest integralną częścią mechanik serwisu sCDN, oferując użytkownikom kontrolę w zarządzaniu ich osobistymi i biznesowymi informacjami. Użytkownicy, którzy pomyślnie aktywowali swoje konta, mają dostęp do poszerzonej ilości opcji konfiguracyjnych. Mogą edytować swoje dane osobowe i adresowe, co jest szczególnie ważne dla procesów fakturowania. Dostępne opcje edycji obejmują zarówno podstawowe dane, takie jak dane osobowe, jak i bardziej szczegółowe informacje, na przykład dane do faktur, które są kluczowe dla transakcji biznesowych. Dodatkowo, mechanizm resetowania hasła stanowi istotny element bezpieczeństwa w serwisie sCDN. Użytkownicy mają możliwość zmiany swojego hasła w dowolnym momencie, co jest szczególnie przydatne w przypadku podejrzeń o nieautoryzowany dostęp do ich konta lub po prostu jako rutynowa praktyka zwiększająca bezpieczeństwo.

Rysunek 6.22. Widok ustawień konta użytkownika dla klienta sieci sCDN

Administrator ma dostęp do funkcji umożliwiających nie tylko przeglądanie szczegółowych informacji o kontach użytkowników, ale także wprowadzanie zmian w ich uprawnieniach. Może to obejmować nadawanie uprawnień administracyjnych wybranym użytkownikom, co jest szczególnie przydatne w sytuacjach, gdy chcemy, aby w systemie był więcej niż jeden administrator.

The screenshot shows a user management interface with two main sections:

- Użytkownicy**: A table listing users with columns: Mail, Imię i nazwisko, Typ konta, and Szczegóły. The table contains four rows:

Mail	Imię i nazwisko	Typ konta	Szczegóły
admin@admin.com	Admin 1	Administrator	edit remove lock unlock reset password
banduram1@gmail.com	Maciek Bandura	Użytkownik	edit remove lock unlock reset password
marcindev@int.pl	Marcin Ślusarczyk	Użytkownik	edit remove lock unlock reset password
webdesign@repulse.pl	Jan Kowalski	Użytkownik	edit remove lock unlock reset password
- Informacje o użytkowniku banduram1@gmail.com**: A detailed view for the user Maciek Bandura, showing personal information, address details, and payment terms.

Informacje personalne: Imię: Maciek, Nazwisko: Bandura.

Szczegóły adresowe: Kod pocztowy: 26-021, Miasto: Daleszyce, Ulica i numer: Kilińskiego.

Dane do faktur: Nazwa firmy: [redacted], Numer NIP: [redacted].

Rysunek 6.23. Panel zarządzania użytkownikami serwisu

Administrator posiada również zdolność do resetowania haseł użytkowników. W przypadku, gdy użytkownik zapomni swojego hasła lub zjadzie potrzeba jego resetowania z innych przyczyn, administrator może inicjować proces, w wyniku którego użytkownik otrzymuje tymczasowe hasło drogą mailową. To zapewnia ciągły dostęp do konta przy jednoczesnym zachowaniu odpowiednich środków bezpieczeństwa.



Dzień dobry, Maciej Bandura,

Otrzymujesz tego maila, ponieważ poprosiłeś/aś o zresetowanie hasła do swojego konta w serwisie sCDN.

Oto Twoje nowe tymczasowe hasło:

[REDACTED]
Zalecamy natychmiastową zmianę hasła po zalogowaniu się. Możesz to zrobić w panelu zarządzania kontem.

Jeśli nie prosiłeś/aś o reset hasła, prosimy zignorować tego maila.

Jeśli masz jakiekolwiek pytania lub potrzebujesz pomocy, skontaktuj się z naszym zespołem wsparcia pod adresem qc.scdn@gmail.com.

qc - sCDN 2023 © Maciej Bandura, Marcin Ślusarczyk | [Strona](#) | [Panel zarządzania](#)

Rysunek 6.24. Mail z tymczasowym hasłem dla użytkownika

6.7.7 Dostarczanie treści - sandbox

W kontekście omawianego wcześniej sandboxa, realizowanego jako strona internetowa z wykorzystaniem elementu <**iframe**>, ważnym jest, aby umożliwiał on renderowanie zawartości z serwera skrajnego, ale robił to na domenie, z której pochodziło pierwotne żądanie użytkownika. W ten sposób odbiorca może być przekonany, że otrzymał treść z serwera źródłowego, a nie serwera skrajnego w sieci sCDN.

```
1   const _iframe = document.querySelector('iframe');
2   const redirect = function (coords = {lat: null, lon: null})
3   {
4       fetch('/redirect',
5       {
6           method: 'POST',
7           body: ....,
8           credentials: 'include'
9       })
10      .then((response) => response.json())
11      .then((resp) =>
12      {
13          const url = new URL(resp.data);
14          const edge = `${url.protocol}//${url.host}`;
15
16          if (url.protocol == 'http:')
17              window.location.href = resp.data;
18
19          const domain = (new URL('http://' + url.pathname.replace('qc-cdn/',
20             ''))).host;
21          _iframe.src = resp.data;
22          ...
23      });
24 }
```

Listing 6.9: Renderowanie treści z serwerów skrajnych u klienta

Ten fragment kodu JavaScript wykorzystuje **fetch** do wysyłania żądania **POST** do endpointu **/redirect**, przekazując niezbędne informacje, takie jak chociażby współrzędne geograficzne. Po odebraniu odpowiedzi, skrypt analizuje otrzymany URL, identyfikuje serwer skrajny i odpowiednio przekierowuje użytkownika. W przypadku protokołu **HTTP**, skrypt realizuje bezpośrednie przekierowanie, podczas gdy dla **HTTPS** zawartość jest ładowana do iframe. Ten sposób przekierowania pozwala na płynne i transparentne dostarczanie treści z różnych lokalizacji, jednocześnie utrzymując wrażenie jednolitej i niezmienionej domeny z perspektywy użytkownika. Jest to kluczowe dla zachowania spójnego doświadczenia użytkownika oraz dla optymalizacji procesów związanych z dystrybucją treści w sieci sCDN.

```
1 window.addEventListener('message', function (e)
2 {
3     document.title = e.data.b;
4     history.pushState({}, '',
5         ↳ `${window.location.protocol}//${window.location.host}${new
6             ↳ URL(e.data.a).pathname}`);
7 });
8
9 window.addEventListener('popstate', function(event)
10 {
11     history.back();
12     _iframe.src = `${_iframe.dataset['edge']}${window.location.pathname}`;
13 });

```

Listing 6.10: Renderowanie treści z serwerów skrajnych u klienta

Powyzszy fragment kodu jest częścią mechanizmu zarządzania stanem i nawigacją przeglądarki w kontekście strony wykorzystującej iframe. Skrypt nasłuchuje na dwa rodzaje zdarzeń: **message** i **popstate**, które są niezbędne do synchronizacji stanu aplikacji sandbox'a z zawartością ładowaną w iframe.

Pierwsza część kodu, nasłuchująca na zdarzenie **message**, odpowiada za aktualizację tytułu dokumentu na podstawie danych otrzymanych z iframe. Gdy iframe przesyła dane za pomocą **postMessage**, skrypt przechwytuje te informacje i aktualizuje tytuł dokumentu. Ponadto, skrypt modyfikuje aktualny stan historii przeglądarki, wykorzystując metodę **history.pushState**, aby odzwierciedlić zmiany w ścieżce dostępu bez konieczności przeładowania strony. Dzięki temu, zmiany w adresie URL są zgodne z zawartością wyświetlana w iframe.

Druga część kodu, reagująca na zdarzenie **popstate**, jest wyzwalana, gdy użytkownik wykonuje akcję nawigacyjną, taką jak kliknięcie przycisku *wstecz* w przeglądarce. W takim przypadku, skrypt powoduje powrót do poprzedniego stanu historii przeglądarki i aktualizuje źródło iframe, aby zachować spójność nawigacji.

Powyzsze mechaniki zapewniają płynną i spójną interakcję użytkownika z dynamicznie ładowanymi treściami iframe, jednocześnie utrzymując integralność i spójność doświadczenia nawigacyjnego na głównej stronie, gdzie jest załadowany sandbox. Jest to kluczowe dla zapewnienia, że użytkownicy mogą swobodnie nawigować i otrzymywać aktualne informacje, niezależnie od tego, gdzie aktualnie się znajdują w strukturze strony.

6.8 Komunikacja z serwerami skrajnymi

Komunikacja z grupą serwerów skrajnych, wykorzystuje REST API. Na początku, ustawiona jest flaga `allow` na `true`, co służy do śledzenia, czy operacje na wszystkich serwerach skrajnych zakończyły się powodzeniem. Następnie, iterując przez listę serwerów skrajnych (`List<qc_EdgeServer> dest`), wykonywane są określone działania dla każdego serwera. Jeżeli jest on aktywny (`edge.getEnabled() == 0`), konstruowany jest specyficzny endpoint API, do którego będą wysyłane żądania. Klasa `HttpHeaders` wykorzystana jest do ustawienia nagłówków żądania, w tym typu zawartości (`MediaType.APPLICATION_JSON`) oraz tokena uwierzytelniającego. Następnie tworzony jest obiekt reprezentujący ciało żądania, który jest konwertowany do formatu JSON.

Wykorzystując obiekt `RestTemplate`, który jest częścią framework'a Spring do obsługi żądań HTTP, wysyłane jest żądanie POST do wcześniej zbudowanego endpointu. Odpowiedź od serwera jest następnie przetwarzana, a jej zawartość wypisywana i konwertowana z powrotem z formatu JSON na obiekt Java. Jeżeli w trakcie komunikacji wystąpi wyjątek, jest on wypisywany na konsolę, a flaga `allow` jest ustawiana na `false`, co sygnalizuje problemy w procesie komunikacji z serwerami skrajnymi.

Komunikacja odbywa się w sposób uniwersalny. W większości przypadków zmieniają się tylko dane oraz ich obsługa, ale sam rdzeń komunikacji zostaje niezmienny. W przypadku, gdy wysyłamy informacje do wielu serwerów skrajnych i na jednym z nich dojdzie do błędu, cała operacja nie jest przerywana, a użytkownik zostaje o tym poinformowany odpowiednim komunikatem.

```
1  // Flaga informująca czy operacja powiodła się na każdym serwerze skrajnym
2  boolean allOk = true;
3
4  /** Komunikacja z grupą serwerów skrajnych w liście List<qc_EdgeServer> dest
5   * /
6  for (qc_EdgeServer edge : dest)
7  {
8      if (edge.getEnabled() == 0)
9          continue;
10
11     String apiEndpoint = String.format("%s/qc-api/domain/add", edge.getUrl());
12     HttpHeaders headers = new HttpHeaders();
13     headers.setContentType(MediaType.APPLICATION_JSON);
14     headers.set("Token", edge.getToken()); /* <-- token uwierzytelniający */
15     data_DomainAdd adder = new data_DomainAdd();
16     /** adder.set(...) - uzupełniamy ciało zapytania informacjami */
17     HttpEntity<String> requestEntity = new HttpEntity<>(adder.toJSON(), 
18         headers);
19
20     try
21     {
22         ResponseEntity<String> responseEntity = new RestTemplate()
23             .postForEntity(apiEndpoint, requestEntity, String.class);
24
25         String response = responseEntity.getBody();
26         System.out.println(response);
27
28         /** resp to otrzymana odpowiedź z serwera skrajnego */
29         APIResp resp = APIResp.fromJSON(response);
30
31         // ...
32     }
33     catch (Exception e)
34     {
35         e.printStackTrace();
36         allOk = false;
37     }
38
39     // ...
40 }
```

Listing 6.11: Komunikacja serwera centralnego z serwerami skrajnymi

6.9 Mechanizm wysyłania maili

Moduł odpowiedzialny za wysyłanie maili ma zasadnicze znaczenie, ponieważ informuje użytkowników o różnych ważnych działaniach związanych z ich kontem i domenami. Centralną częścią tej mechaniki jest serwis `GmailService`, który służy do wysyłania przygotowanych wiadomości e-mail. Dzięki temu serwisowi, proces wysyłania maili jest uproszczony i zautomatyzowany. Kluczową rolę odgrywa również ogólna klasa `GmailMail`, która stanowi bazę dla szeregu specjalizowanych klas reprezentujących różne rodzaje maili, które system może wysłać. Te specjalistyczne klasy są dostosowane do konkretnych potrzeb komunikacyjnych, takich jak potwierdzenie rejestracji, zmiana hasła czy powiadomienia o ważnych aktualizacjach dotyczących konta użytkownika lub jego domen.

```
1 @Service
2 public class GmailService
3 {
4     @Autowired
5     private JavaMailSender javaMailSender;
6
7     @SneakyThrows
8     public void sendMail (String to, String subject, String content)
9     {
10         MimeMessage mimeMessage = javaMailSender.createMimeMessage();
11         MimeMessageHelper helper = new MimeMessageHelper(mimeMessage, true, "UTF-8");
12
13         helper.setTo(to);
14         helper.setSubject(subject);
15         helper.setText(content, true);
16
17         javaMailSender.send(mimeMessage);
18     }
19 }
```

Listing 6.12: Serwis `GmailService` zajmujący się wysyłaniem wiadomości

Metoda `sendMail` w serwisie odpowiedzialnym za wysyłanie wiadomości na serwerze centralnym odgrywa istotną rolę w personalizacji i dostosowywaniu komunikacji z użytkownikami. Ta metoda umożliwia wysyłanie wiadomości email na określony adres, z konkretnym tematem i zawartością w formacie HTML. Dzięki temu, wiadomości mogą być dostosowywane pod kątem wizualnym i stylistycznym, co jest standardową

praktyką w wielu firmach, zapewniając spójność komunikacji z wizerunkiem systemu sCDN. Chociaż usługa nosi nazwę `GmailService`, jest ona uniwersalna i może być używana do współpracy z różnymi dostawcami usług email. W projekcie sCDN, dla celów demonstracyjnych i testowych, utworzono adres email `qc.scdn@gmail.com` - stąd pochodzi nazwa tego serwisu. Konfiguracja poczty, z której wysyłane są wiadomości znajduje się w `application.properties`.

```
1  @Service
2  public class GmailMail
3  {
4      @SneakyThrows
5      public static void prep (GmailService service, String to, String subject, String
6          ↪ content)
7      {
8          ClassPathResource resource = new ClassPathResource("mails/body.html");
9          byte[] fileData = FileCopyUtils.copyToByteArray(resource.getInputStream());
10         String body = new String(fileData);
11         body = body.replace("%content", content);
12         service.sendMail(to, String.format("[qc - sCDN] %s", subject), body);
13     }
14
15     @SneakyThrows
16     public static String load (String name)
17     {
18         ClassPathResource resource = new
19             ↪ ClassPathResource(String.format("mails/%s.html", name));
20         byte[] fileData = FileCopyUtils.copyToByteArray(resource.getInputStream());
21         return new String(fileData);
22     }
23 }
```

Listing 6.13: Klasa po której dziedziczą możliwe maile

Klasa `GmailMail` pełni funkcję bazową dla różnych rodzajów wiadomości email, które mogą być wysyłane do użytkowników. Kluczowe metody tej klasy, `load(String name)` i `prep(...)`, odgrywają ważną rolę w procesie tworzenia i wysyłania spersonalizowanych wiadomości email.

Metoda **load** jest odpowiedzialna za **załadowanie szablonu** wiadomości z zasobów aplikacji. Dzięki temu procesowi, różne rodzaje wiadomości, od potwierdzeń rejestracji po powiadomienia o zmianach w serwisie, mogą korzystać z predefiniowanych i spójnych wzorców. Metoda **prep**, z kolei, zajmuje się przygotowaniem ogólnego szablonu dla wszystkich wiadomości. Proces ten obejmuje integrację załadowanego wcześniej specyficznego szablonu wiadomości z ogólnym szablonem, formatowanie tytułu maila, a także wykorzystanie serwisu **GmailService** do wysłania odpowiednio przygotowanej wiadomości do użytkownika. Dzięki temu, wiadomości email wysyłane przez system sCDN są nie tylko spersonalizowane, ale także utrzymane w jednolitym stylu.

```
1 public class GmailActivate extends GmailMail
2 {
3     public static void send (GmailService gmailService, qc_Users user)
4     {
5         String ht = load("login");
6         ht = ht.replaceAll("%s_name", user.getName());
7         ht = ht.replaceAll("%s_surname", user.getSurname());
8         ht = ht.replaceAll("link_do_aktywacji",
9             String.format("http://cdn.localhost/api/user/activate/%s",
10                ↳ user.getNotActivated())
11        );
12        prep
13        (
14            gmailService, user.getMail(), "Aktywacja konta", ht
15        );
16    }
17 }
```

Listing 6.14: Klasa reprezentująca mail aktywacyjny

Klasa reprezentująca mail aktywacyjny w systemie sCDN stanowi znakomity przykład wykorzystania zintegrowanego mechanizmu wysyłania emaili na serwerze centralnym. Proces tworzenia i wysyłania maila aktywacyjnego przebiega w kilku kluczowych etapach.

Na początku, klasa ta wykorzystuje odziedziczoną metodę **load** do pobrania specyficznego szablonu emaila. Szablon ten, znajdujący się w lokalizacji `.resources-mails/activate.html`, jest specjalnie zaprojektowany do treści maila aktywacyjnego. W tym szablonie następuje uzupełnienie kluczowych danych, takich jak link aktywacyjny oraz inne informacje personalizujące wiadomość dla konkretnego użytkownika. Po przygotowaniu i uzupełnieniu szablonu, przygotowana treść maila jest przekazywana do kolejnej odziedziczonej metody – `prep`. W tej fazie, treść maila jest łączona z ogólnym szablonem mailowym. Metoda `prep` także odpowiada za nadanie tematu wiadomości – w tym przypadku "*Aktywacja konta*" – oraz końcowe wysłanie złożonego emaila na adres użytkownika za pomocą serwisu `GmailService`.

```
1 @PostMapping("/api/user/add")
2 public String apiUserAdd (...)

3 {
4     APIResp apiResp = new APIResp();
5     (...)

6
7     GmailActivate.send(gmailService, usr);
8     return apiResp.success();
9 }
```

Listing 6.15: Przykład wysłania maila

Dzięki starannie zaprojektowanej abstrakcji systemu mailowego, proces wysyłania wiadomości staje się znacznie uproszczony i bardziej efektywny. W praktyce, aby wysłać email, wystarczy wywołać metodę **send** z odpowiednimi parametrami. Metoda ta wymaga instancji serwisu `GmailService` oraz dodatkowych danych niezbędnych do **personalizacji** treści maila. W przypadku wysyłania maila aktywacyjnego wymagany jest model użytkownika zawierający kluczowe informacje, takie jak adres email i dane do personalizacji wiadomości. Cały proces wysyłania maila aktywacyjnego sprowadza się więc do **pojedynczej linii** kodu, która uruchamia skomplikowany proces tworzenia, formatowania i wysyłania wiadomości email.

System wysyłania maili jest integralną częścią sCDN, ponieważ zapewnia niezbędny kanał komunikacji między usługą a jej użytkownikami. Jest to szczególnie ważne w kontekście zarządzania kontem i domenami, gdzie szybka i skuteczna komunikacja może mieć znaczący wpływ na doświadczenia użytkowników.

6.10 Obsługa systemu płatności

System biznesowy w sCDN oferuje użytkownikom dwa plany: darmowy oraz płatny, różniące się zakresem oferowanych usług.

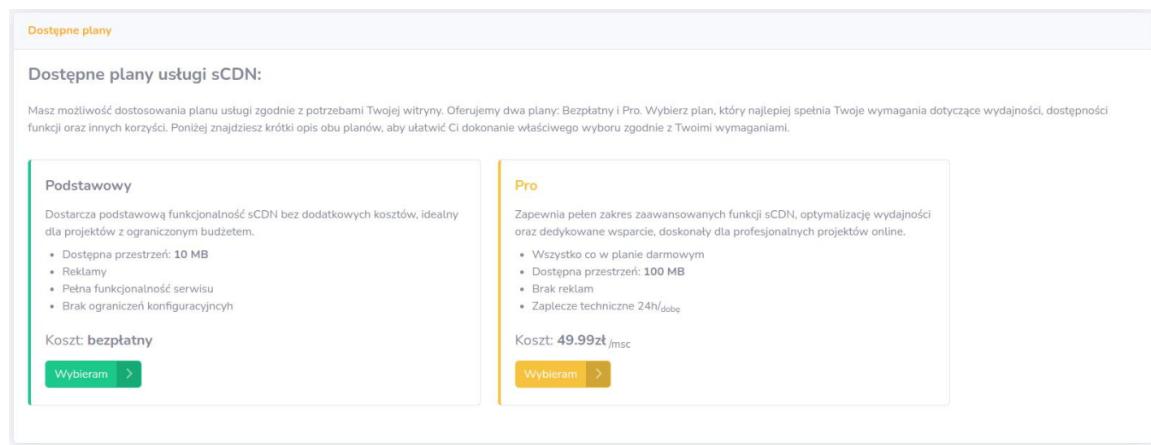
1. Plan podstawowy (bezpłatny)

- Użytkownicy korzystający z darmowego planu otrzymują **10 MB** przestrzeni dyskowej na serwerze skrajnym.
- Zapewniona jest pełna funkcjonalność serwisu, bez jakichkolwiek ograniczeń konfiguracyjnych.

Kluczową cechą darmowego planu jest jednak wyświetlanie reklam typu popup na stronach użytkowników. Reklamy te są narzucone przez system jako forma monetyzacji dla bezpłatnej usługi.

2. Plan PRO (49.99zł/mies)

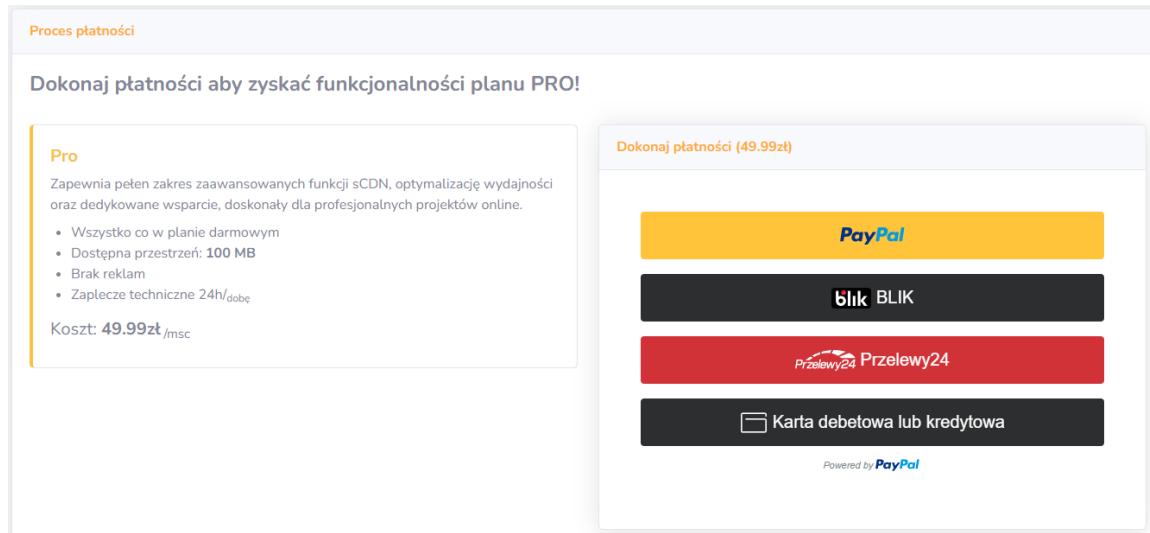
Plan płatny rozszerza ofertę darmowego planu, dostarczając użytkownikom więcej miejsca na serwerze skrajnym - do **100 MB**. Zaletą jest brak reklam, co oznacza, że strony użytkowników są wolne od treści reklamowych, zapewniając lepsze doświadczenie zarówno dla właścicieli stron, jak i ich odwiedzających.



Rysunek 6.25. Porównanie planów usługi sCDN

Wprowadzenie systemu płatności do sCDN wymagało odpowiedniego doboru operatora płatności po stronie serwera centralnego, który spełniłby zarówno wymagania techniczne, jak i biznesowe. Wybór padł na firmę **PayPal**, co było podyktowane kilkoma istotnymi czynnikami. PayPal oferuje łatwą w integracji i przejrzystą bibliotekę, co umożliwiło sprawną i efektywną implementację mechanizmów płatniczych. Kluczowe było także zapewnienie bezpieczeństwa transakcji, co PayPal osiąga,

przetwarzając wszystkie wrażliwe dane finansowe bezpośrednio po swojej stronie. To znacząco zwiększa bezpieczeństwo transakcji i minimalizuje ryzyko dla sCDN. Dodatkowo, PayPal udostępnia funkcję umożliwiającą łatwe sprawdzenie statusu płatności za pomocą pojedynczego endpointu, co pozwala na szybką weryfikację, czy transakcja została zrealizowana pomyślnie. Nie bez znaczenia jest także dostępność trybu testowego (sandbox), który pozwala na testowanie procesów płatniczych bez konieczności przeprowadzania rzeczywistych transakcji finansowych.



Rysunek 6.26. Proces płatności

```

1 <PayPalScriptProvider options={{currency: "PLN", "client-id": __client_id}}>
2   <PayPalButtons
3     createOrder={(data, actions) => {
4       return actions.order.create({
5         purchase_units: [
6           {
7             amount: { value: 49.99 },
8           },
9         ],
10        onCancel={onCancel}
11        onApprove={onApprove}
12      />
13    </PayPalScriptProvider>

```

Listing 6.16: Element płatności z paczki paypal

Element <**PayPalScriptProvider**>, jest odpowiedzialny za załadowanie i konfigurację skryptów PayPal. W opcjach tego komponentu określona jest waluta (w tym przypadku polski złoty, "PLN") oraz identyfikator klienta, który jest unikalnym identyfikatorem konta PayPal. Następnie, użyty jest komponent <**PayPalButtons**>, który renderuje przyciski płatności PayPal. Wewnątrz tego komponentu zdefiniowane są funkcje obsługujące różne etapy procesu płatności. Funkcja `createOrder` jest wywoływana, aby utworzyć zamówienie z określoną kwotą (tutaj 49.99 PLN). To właśnie w tej funkcji określa się szczegóły transakcji. Dodatkowo, kod zawiera obsługę zdarzeń **onCancel** i **onApprove**. Pierwszy z nich jest uruchamiany, gdy użytkownik anuluje proces płatności. Określa, co ma się stać po anulowaniu transakcji.

```
1 const onApprove = function (data, actions)
2 {
3     return actions.order.capture().then((details) =>
4     {
5         fetch(`$__server__/api/paypal/checkout/${subpanel}/${details.id}`,
6         {
7             method: 'GET',
8             credentials: 'include'
9         })
10        .then((response) => response.json())
11        .then((resp) =>
12        {
13            if (!resp.ok)
14                throw new Error(resp.msg);
15            window.location.href = `/${mail}/domains/${subpanel}`;
16        });
17    });
18 }
```

Listing 6.17: Metoda wywoływana po dokonaniu płatności

Po pomyślnym zatwierdzeniu płatności przez PayPal, funkcja **onApprove** aktywuje się i inicjuje interakcję z serwerem centralnym. W praktyce, `onApprove` wywołuje określony endpoint na backendzie, przekazując mu istotne informacje o transakcji. Kluczowe elementy przesyłane do serwera to identyfikator transakcji oraz nazwa domeny, dla której dokonano płatności. Identyfikator transakcji jest wykorzystywany do weryfikacji i potwierdzenia płatności po stronie serwera centralnego, co zapewnia spójność danych. Natomiast nazwa domeny jest używana do skojarzenia płatności z odpowiednią domeną w systemie.

Endpoint `/api/paypal/checkout/{domain}/{id}`, po otrzymaniu informacji z funkcji `onApprove`, wykonuje serię działań mających na celu przetworzenia transakcji płatniczej. Pierwszym krokiem jest weryfikacja płatności w systemie PayPal, gdzie identyfikator transakcji jest używany do potwierdzenia jej statusu. Po potwierdzeniu i zatwierdzeniu płatności przez PayPal, endpoint aktualizuje informacje w bazie danych serwera centralnego. Po aktualizacji danych, informacje te są przesyłane na wszystkie serwery skrajne, co zapewnia spójność i aktualność danych w całym systemie.

```
1  @GetMapping("/api/paypal/checkout/{domain}/{id}")
2  public String apiPlansPaid (@PathVariable("domain") String domain,
3      @PathVariable("id") String id)
4  {
5      APIResp apiResp = new APIResp();
6      (...)

7      String trDetailsJSON = paypalService.getTransactionDetails(id);
8      FsdPayPalTransaction fsdPaypalTransaction
9          = FsdPayPalTransaction.fromJSON(trDetailsJSON);

10     if (!fsdPaypalTransaction.getStatus().equals("COMPLETED"))
11         return apiResp.fail("Nie można potwierdzić płatności...");

12     (...)

13     if (!DomainsController.update(jdbcTemplate, dom))
14         return apiResp.fail("Wewnętrzny błąd serwera...");

15     (...)

16     List<qc_EdgeServer> edges = DomainsController
17         .fetchEdgeServersByDomianName(jdbcTemplate, domain);
18     for (qc_EdgeServer edge : edges)
19     {
20         String apiEndpoint = String
21             .format("%s/qc-api/domain/edit", edge.getUrl());
22         (...)

23     }

24     return apiResp.success();
25 }
```

Listing 6.18: Endpoint wywoływany po dokonaniu płatności

W ramach projektu zdecydowano się na zaimplementowanie własnej integracji z API PayPal, korzystając bezpośrednio z endpointów dostarczanych przez ten system płatności. Kluczowym elementem tej integracji jest serwis **PayPalService**, który zawiera dwie metody: `getAccessToken()` oraz `getTransactionDetails(...)`. Pierwsza z nich wykorzystuje REST API PayPal do uzyskania tokenu sesji, który jest niezbędny do autoryzacji i wykonania różnych zapytań w systemie PayPal. Proces ten zapewnia bezpieczne i efektywne zarządzanie sesją, umożliwiając serwerowi centralnemu komunikację z PayPal. Z kolei metoda `getTransactionDetails(...)` jest wykorzystywana do pozyskiwania szczegółowych informacji o konkretnych transakcjach. Na podstawie identyfikatora transakcji, ta metoda umożliwia pobranie wszelkich niezbędnych danych, takich jak status płatności, kwota czy dane odbiorcy, co jest kluczowe dla procesu weryfikacji.

```
1 @Data
2 public class FsdPayPalTransaction
3 {
4     String id;
5     String status;
6     Float amount;
7
8     public static FsdPayPalTransaction fromJSON (String json)
9     {
10         FsdPayPalTransaction fsdPaypalTransaction = new FsdPayPalTransaction();
11         JsonParser parser = new JsonParser();
12         JsonObject jsonObject = parser.parse(json).getAsJsonObject();
13         fsdPaypalTransaction.setId(jsonObject.get("id").getAsString());
14         fsdPaypalTransaction.setStatus(jsonObject.get("status").getAsString());
15         String strAmount = jsonObject
16             .getAsJsonArray("purchase_units").get(0).getAsJsonObject()
17             .get("amount").getAsJsonObject().get("value").getAsString();
18
19         fsdPaypalTransaction.setAmount(Float.parseFloat(strAmount));
20         return fsdPaypalTransaction;
21     }
```

Listing 6.19: Obiekt reprezentujący transakcję z systemu PayPal

Otrzymane z serwisu PayPal informacje o płatnościami są dostarczane w formacie JSON. W celu ułatwienia ich manipulacji i zwiększenia abstrakcji w kontekście języka obiektowego Java, te dane są mapowane na obiekt klasy `FsdPayPalTransaction`.

7. Implementacja serwera skrajnego

Autor: Marcin Ślusarczyk

7.1 Wykorzystywane technologie

W kontekście tworzenia serwera skrajnego dla programowego CDN, wybór technologii został dokonany z precyzyjnym uwzględnieniem założeń projektu oraz aktualnych trendów rynkowych. Głównym celem było wykorzystanie istniejącej infrastruktury, która jest powszechnie dostępna i sprawdzona w praktyce. W tym celu, podjęto decyzję o zastosowaniu technologii, które są szeroko stosowane, jednocześnie zapewniając wydajność, stabilność i elastyczność niezbędną do obsługi dynamicznego ruchu w sieci CDN. W tym kontekście, technologią, która wyróżnia się jako filar serwera skrajnego, jest **PHP**. Jest to język, który przez lata zdobył solidną reputację, zwłaszcza w kontekście budowania dynamicznych stron internetowych i aplikacji, czyniąc go naturalnym wyborem dla podstaw projektu serwera skrajnego. Rozpowszechnienie PHP w środowisku komercyjnych usług hostingowych stanowi jeden z kluczowych powodów jego wyboru jako podstawy serwera skrajnego. PHP, jako język skryptowy do aplikacji internetowych, jest nie tylko szeroko dostępny, ale także oferuje bogaty zestaw funkcji, które są idealnie dopasowane do potrzeb szybkiego i efektywnego przetwarzania danych. Jego elastyczność i łatwość integracji z różnymi systemami baz danych oraz innymi technologiami webowymi czynią go idealnym wyborem dla serwerów skrajnych w programowej sieci dostarczania zawartości.

W ramach projektowania serwera skrajnego dla systemu sCDN, istotnym aspektem jest wybór technologii, które zapewniają wydajną i bezpieczną wymianę danych. W tym kontekście, kluczową rolę odgrywa biblioteka **cURL**. Wykorzystanie cURL w projekcie serwera skrajnego jest ściśle powiązane z jego fundamentalnymi funkcjami i celami. Serwery skrajne, będące nieodzownym elementem architektury sieci sCDN, mają za zadanie przechowywanie i dostarczanie treści jak najbliżej użytkownika końcowego. Kluczem do realizacji tego zadania jest efektywne pobieranie i wysyłanie danych, co wymaga zastosowania narzędzia, które jest nie tylko wszechstronne, ale również zapewnia szybką i niezawodną komunikację. cURL, dzięki swojej wszechstronności i wsparciu dla wielu protokołów, staje się idealnym narzędziem w takim środowisku. Biblioteka ta dostarcza łatwo parametryzowany interfejs. Jest to niezwykle istotne, gdyż umożliwia to łatwe manipulowanie nagłówkami HTTP oraz zarządzanie plikami cookies, co jest niezbędne w programowym podejściu do CDN.

Elastyczność cURL w obszarze zarządzania nagłówkami pozwala na precyzyjne kontrolowanie procesu przesyłania danych, co jest kluczowe w zapewnieniu efektywności komunikacji między serwerami skrajnymi a serwerami źródłowymi i klientami końcowymi. Ta zdolność do szczegółowego konfigurowania żądań i odpowiedzi, w połączeniu z wydajnością i niezawodnością cURL, czyni go idealnym wyborem dla potrzeb serwera skrajnego w systemie sCDN. W scenariuszu, gdzie szybkość i niezawodność są kluczowe, cURL zapewnia, że żądania są przetwarzane szybko, a dane dostarczane są bez opóźnień. Jest to szczególnie istotne w kontekście tunelowania zapytań i pobierania zawartości, które są głównymi zadaniami serwera skrajnego. cURL umożliwia serwerom skrajnym wydajne zarządzanie zapytaniami od użytkowników, kierując je albo do lokalnie przechowywanej zawartości, albo do serwera źródłowego. To sprawia, że cURL jest nie tylko narzędziem do transferu danych, ale również istotnym elementem w procesie optymalizacji dostarczania treści. Ważnym aspektem są również zasady użycia tej biblioteki, jest ona wydana na licencji **MIT**, która jest jedną z najbardziej liberalnych i elastycznych licencji w świecie **oprogramowania open-source**. Licencja MIT pozwala na szerokie wykorzystanie, modyfikację oraz dystrybucję biblioteki cURL, zarówno w projektach prywatnych, jak i komercyjnych, bez konieczności opłacania licencji czy uwzględniania kodu źródłowego. To sprawia, że cURL jest atrakcyjnym wyborem dla projektu o takiej charakterystyce jak sCDN.



Rysunek 7.1. Główne technologie serwera skrajnego

Wybór MariaDB jako systemu zarządzania bazą danych dla serwera skrajnego w systemie sCDN był wynikiem starannej analizy potrzeb projektu oraz panujących trendów na rynku hostingowym. MariaDB, będąc forkiem MySQL, cieszy się dużą popularnością wśród dostawców usług hostingowych, co stanowi jeden z głównych powodów jej wyboru. Oprócz powszechnego wykorzystania w portalach hostingowych, istotnym aspektem jest również licencja MariaDB. Jest ona dostępna na warunkach otwartego oprogramowania (open-source), co sprawia, że jej używanie jest bezpłatne i otwarcie na modyfikacje. Dzięki temu serwery skrajne w systemie sCDN mogą korzystać z pełnych możliwości MariaDB bez dodatkowych opłat licencyjnych, co jest

szczególnie ważne w kontekście ograniczania kosztów operacyjnych jak i samego charakteru projektu. MariaDB oferuje również wysoką wydajność i niezawodność, co jest niezbędne dla zapewnienia stabilnego i efektywnego zarządzania danymi w dynamicznym środowisku sieci CDN. Jej zaawansowane funkcje, takie jak wsparcie dla dużych ilości danych, replikacja czy wydajne mechanizmy indeksowania, przyczyniają się do zapewnienia optymalnej wydajności bazy danych.

7.2 Struktura aplikacji

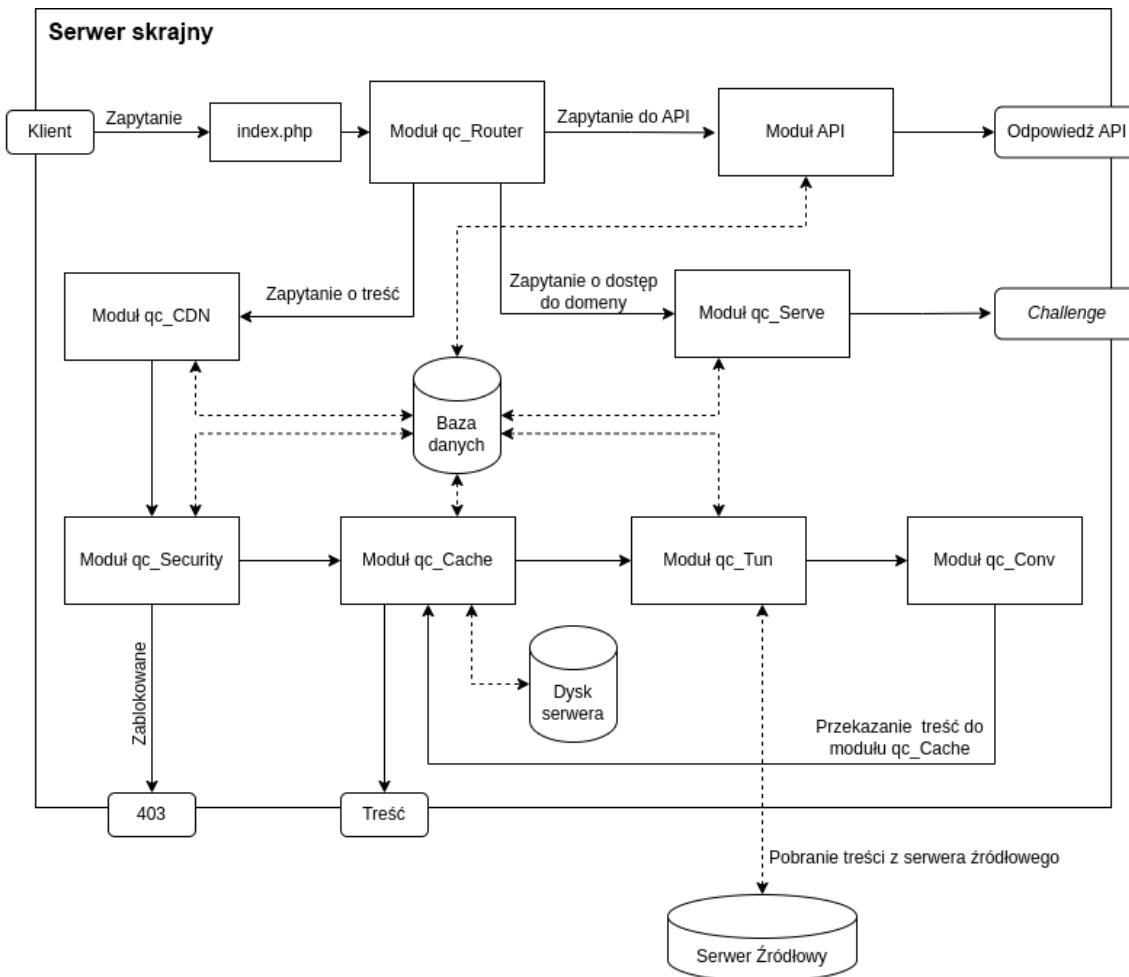
Centralną częścią architektury serwera skrajnego jest fakt, że wszystkie zapytania kierowane są do jednego punktu wejścia – pliku **index.php**. Takie podejście pozwala na uporządkowaną i centralnie zarządzaną obsługę wszystkich żądań, co jest niezwykle ważne w kontekście obsługi różnorodnych zapytań wyprofilowanych pod kątem serwerów źródłowych. Ważnym aspektem jest utrzymanie zasady, że klient komunikujący się z serwerem skrajnym nie powinien odczuwać różnicy w porównaniu z komunikacją z serwerem źródłowym. W pliku index.php tworzone są dwie główne zmienne globalne, które odgrywają fundamentalną rolę w działaniu aplikacji. Pierwsza z nich **\$_config** – jako zmienna globalna jest odpowiedzialna za obsługę pliku konfiguracyjnego serwera. Dzięki niej, aplikacja ma dostęp do wszystkich niezbędnych ustawień i parametrów. Zmienna ta zapewnia łatwy dostęp do informacji konfiguracyjnych, takich jak ustawienia połączenia z bazą danych, tokeny autoryzacyjne, informacje o konwerterach treści, a także inne specyficzne ustawienia serwera skrajnego. Kolejną zmienną jest **\$_dbc** – ta zmienna jest odpowiedzialna za komunikację z bazą danych. Zapewnia ona połączenie z bazą danych, umożliwia wykonywanie zapytań oraz obsługę wyników.

W procesie działania serwera skrajnego w systemie sCDN, istotną rolę pełni **qc_Router** – odpowiedzialny za efektywne zarządzanie routingiem w aplikacji. Moduł ten jest kluczowy, gdyż to on decyduje, w jaki sposób różnorodne żądania są przetwarzane i kierowane do właściwych części systemu. **qc_Router** charakteryzuje się szczególnym podejściem do obsługi ścieżek. W strukturze routingu wyróżnione są dwie specjalne ścieżki, które odgrywają istotną rolę w działaniu serwera:

- **/qc-api:** Ścieżka ta jest przeznaczona dla interfejsu programistycznego (API) serwera skrajnego. Dzięki niej możliwe jest wykonywanie operacji zarządzających i monitorujących działanie serwera.

- **/qc-cdn:** Przy pomocy tej ścieżki klient inicjuje dostęp do wybranej domeny, którą serwer skrajny będzie świadczyć.

Pozostałe ścieżki w systemie routingu są traktowane jako zapytania o treści na świadczonej domenie. Oznacza to, że żądania do tych ścieżek są obsługiwane tak, jakby były bezpośrednio związane z zawartością wybranej domeny.



Rysunek 7.2. Schemat architektury serwera skrajnego

W architekturze serwera skrajnego w systemie sCDN można wyróżnić kilka klu-czowych modułów, które wspólnie zapewniają pełną funkcjonalność i bezpieczeństwo systemu. Każdy z tych modułów odgrywa unikalną rolę w procesie obsługi żądań i zarządzania treściami.

Moduł API jest "sercem" interfejsu programistycznego serwera. Przechowuje on listę dostępnych endpointów API wraz z możliwymi żądaniami i jest odpowiedzialny za autoryzację oraz wykonanie odpowiednich operacji. **Moduł qc_Serve** odgrywa

kluczową rolę w dostarczaniu użytkownikom dostępu do świadczonych domen. Dwórzysy sesje użytkowników i weryfikuje przeglądarki poprzez wysyłanie tzw. "challenge". Moduł **qc_CDN** jest odpowiedzialny za weryfikację sesji użytkowników oraz pobieranie kluczowych informacji o świadczonych domenach z bazy danych. W kontekście świadczenia treści jest to kluczowy moduł, z którego korzystają inne moduły. Moduł **qc_Security** zapewnia bezpieczeństwo systemu, blokując dostęp do zastrzeżonych treści na podstawie zasad ustawionych przez właścicieli domen, a także chroni przed atakami typu **XSS** w parametrach GET. Stanowi on główny element w ochronie zarówno serwera, jak i użytkowników końcowych. Moduł **qc_Cache** zarządza zapisem lokalnej zawartości, decydując, które treści powinny być cache'owane, a także zajmuje się ich zapisem i odczytem. Moduł **qc_Tun** jest odpowiedzialny za komunikację z serwerem źródłowym, wymianę plików cookies oraz przekazywanie ważnych informacji w nagłówkach, takich jak **User-Agent** czy **Content-Type**. Dzięki temu modułowi, serwer skrajny może efektywnie komunikować się z serwerem źródłowym i zapewniać spójność treści. Moduł **qc_Conv** odpowiada za modyfikacje i optymalizację treści dostarczanych przez serwer. W planie bezpłatnym dodatkowo "dokleja" reklamy, co stanowi część modelu biznesowego systemu.

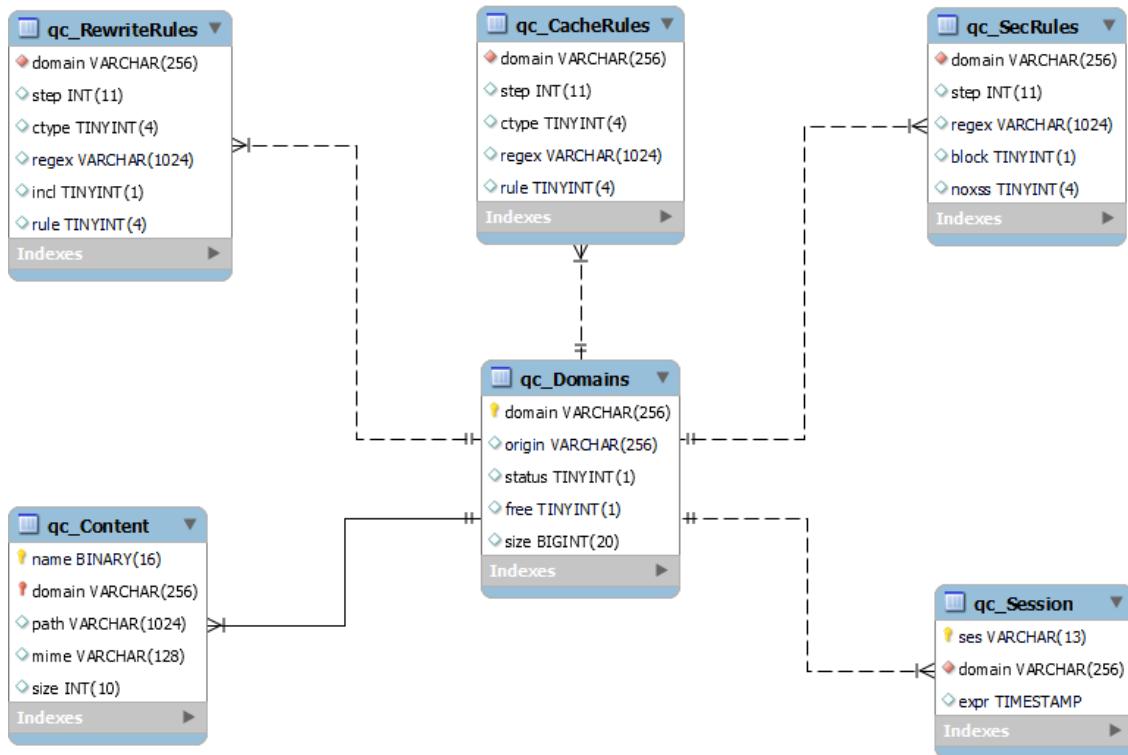
W procesie projektowania struktury kodu serwera skrajnego w systemie sCDN, przyjęto w dużej mierze zasadę **clean code** oraz technikę **guard clauses**, co miało znaczący wpływ na jego jakość i funkcjonalność. Zasady clean code koncentrują się na tworzeniu kodu, który jest zarówno czytelny, jak i łatwy w utrzymaniu. Oznacza to, że kod powinien być prosty, dobrze zorganizowany, a jego intencje – jasno wyrażone, co umożliwia łatwą współpracę i rozwój projektu przez różnych programistów. Z kolei **guard clauses** (warunki strażnicze) to technika programistyczna polegająca na wczesnym zakończeniu wykonywania funkcji, jeśli nie są spełnione określone warunki, co zwiększa czytelność i bezpieczeństwo kodu, redukując zagnieździenia i poprawiając jego strukturę. Kod serwera skrajnego jest silnie obiektowy, co zapewnia modularność i ułatwia zarządzanie złożonymi systemami. Jednak w niektórych miejscach, specyfika języka PHP wymaga pewnych odstępstw od tych zasad, szczególnie w obszarach takich jak renderowanie wyzwania ("challenge") czy generowanie stron błędów. W tych sytuacjach, praktyczność i funkcjonalność mają priorytet, choć ogólna struktura kodu nadal pozostaje zgodna z zasadami clean code i wykorzystuje guard clauses tam, gdzie to możliwe. Dzięki temu, kod jest nie tylko wydajny, ale także odporny na błędy, czytelniejszy i łatwiejszy w utrzymaniu.

Podczas implementacji, kod serwera skrajnego został zaprojektowany z silnym naciskiem na obiektowość, co jest zgodne z tymi zasadami. Jednak w niektórych obszarach, takich jak renderowanie wyzwań czy generowanie stron błędów, specyfika języka PHP wymusiła pewne odstępstwa od tych zasad. Mimo to, ogólny projekt kodu utrzymuje wysoki standard czytelności i organizacji, co jest niezbędne dla efektywnego i bezproblemowego działania systemu sCDN.

7.3 Schemat bazy danych

Projektowanie schematu bazy danych dla serwera skrajnego w systemie sCDN zostało wykonane z naciskiem na skuteczność i spójność strukturalną, co jest kluczowe dla efektywnego zarządzania danymi w dynamicznym środowisku sieci CDN. Podobnie jak w przypadku serwera centralnego została zastosowana 1. postać normalna (1NF) w bazie danych serwera skrajnego, która zapewnia, że każdy atrybut w rekordzie jest atomowy, a każdy rekord zawiera unikalne informacje, co jest niezbędne dla uniknięcia problemów z redundancją i sprzecznością danych. W procesie projektowania bazy danych dla serwera skrajnego w systemie sCDN, istotne było także zapewnienie, że struktura danych będzie spójna i tożsama z tą na serwerze centralnym. Takie podejście ma na celu ułatwienie integracji między różnymi częściami systemu, minimalizując różnice w zarządzaniu i przetwarzaniu danych na obu typach serwerów. Dzięki temu, operacje na danych mogą być przeprowadzane w sposób bardziej jednolity i efektywny, bez konieczności stosowania skomplikowanych mechanizmów adaptacyjnych czy konwersji danych. Stworzenie częściowo tożsamyh schematów bazy danych na serwerach skrajnych i centralnym pozwala na łatwiejsze zarządzanie, synchronizację i wymianę danych między różnymi elementami systemu. To z kolei przekłada się na większą efektywność operacyjną, zmniejszając ryzyko błędów i ułatwiając wspólne procesy, takie jak chociażby zarządzanie regułami lokalnych kopii i konwerterów.

Tabele zostały zaprojektowane w taki sposób, aby odpowiadać specyficznym potrzebom serwera skrajnego. Na przykład, tabela **qc_Domains** przechowuje informacje o domenach obsługiwanych przez serwer, w tym ich pochodzenie i status. Jest to niezbędne dla zarządzania i dystrybucji treści do odpowiednich lokalizacji. Kolejna tabela, **qc_Content**, odpowiada za przechowywanie danych o konkretnych treściach, takich jak ścieżka, typ MIME i rozmiar plików, co umożliwia efektywne zarządzanie zawartością.



Rysunek 7.3. Schemat bazy danych serwera skrajnego

Tabele **qc_CacheRules**, **qc_RewriteRules** i **qc_SecRules** odgrywają kluczową rolę w definiowaniu zasad cache, przekierowań i zabezpieczeń dla domen. Pozwalają one na szczegółowe ustalanie, jak treści są przechowywane, przetwarzane i zabezpieczane na serwerach skrajnych. Wykorzystanie kluczy obcych w tych tabelach zapewnia integralność danych poprzez utrzymanie spójnych i zaktualizowanych relacji między różnymi elementami bazy danych. Tabela **qc_Session** przechowuje informacje o aktywnych sesjach użytkowników. Każda sesja jest aktywna przez minimum 24 godziny.

7.4 Funkcjonalności serwera skrajnego

7.4.1 Wymagania funkcjonalne

- **Świadczenie interfejsu programistycznego**
 - Autoryzacja zapytań do API.
 - Możliwość usunięcia zapisanej zawartości z serwera skrajnego przypisanej do konkretnej domeny.
 - Pobranie listy zapisanej lokalnie zawartości dla wybranej domeny.
 - Ustawienie zasad cache'owania dla wybranej domeny.
 - Możliwość: rejestracji, edycji i usunięcia domeny.
 - Możliwość pobrania informacji konfiguracyjnych oraz systemowych.
 - Możliwość zresetowania serwera do stanu fabrycznego.
 - Ustawienie zasad konwertera dla konkretnej domeny.
 - Ustawienie zasad bezpieczeństwa dla wybranej domeny.
- **Świadczenie punktu dostępu do domeny**
 - Tworzenie oraz zarządzanie sesjami użytkowników.
 - Weryfikacja przeglądarki użytkownika.
 - Usuwanie plików cookies innych świadczonych domen.
- **Modyfikacja świadczonych treści**
 - Kompresja udostępnianych zdjęć.
 - Minifikacja skryptów js oraz css.
 - Komunikacja z sandbox'em.
 - Zamiana ścieżek dostępu w dostarczanych dokumentach na adres serwera skrajnego.
 - "Doklejanie" reklam w planie bezpłatnym.
- **Tunelowanie zapytań**
 - Tunelowanie zapytań o zawartość do serwera klienta.
 - Odtwarzanie plików cookies.
 - Przekazywanie informacji z nagłówków zapytań.
- **Lokalne zapamiętywanie zawartości**
 - Tworzenie fizycznej kopii pliku na dysku serwera.

- Decydując o zapisie danej zawartości na podstawie reguł ustalonych przez użytkownika.
 - Przekazywanie lokalnie zapisanej zawartości.
- **Firewall**
 - Aplikowanie zdefiniowanych przez użytkownika zasad bezpieczeństwa.
 - Blokowanie dostępu do zastrzeżonych treści.
 - Usuwanie ataków XSS z parametrów GET.
 - **Komunikaty błędów**
 - Zapewnienie deskryptywnych komunikatów błędów.
 - Zapewnienie obsługi i wykrywania błędów na każdym poziomie systemu.

7.4.2 Wymagania niefunkcjonalne

- Stabilne działanie na różnych serwerach hostingowych PHP.
- Współpraca z możliwie największą ilością systemów bazodanowych oferowanych przez zewnętrznych dostawców usług hostingowych.
- Zapewnienie łatwej rozbudowy systemu o nowe moduły i funkcjonalności.
- Zabezpieczenie użytkowników i systemu przed różnymi atakami sieciowymi (XSS, boty, DDoS).
- Zachowanie integralności danych z serwerem centralnym.
- Zapewnienie dużej odporności na błędy.

7.5 Plik konfiguracyjny i .htaccess

W środowisku serwerów skrajnych systemu sCDN, plik konfiguracyjny stanowi centralne miejsce do przechowywania i zarządzania wszystkimi kluczowymi ustawieniami i parametrami, które są niezbędne do prawidłowego funkcjonowania serwera. Zastosowanie jednolitego pliku konfiguracyjnego ma szereg zalet, w tym:

- **Centralizacja zarządzania:** Umożliwia administratorom dostęp do wszystkich konfiguracji serwera w jednym miejscu, co znacząco ułatwia zarządzanie i monitorowanie.
- **Łatwość aktualizacji:** Ułatwia aktualizowanie ustawień, ponieważ zmiany można wprowadzać centralnie, bez konieczności modyfikacji wielu plików lub ustawień na różnych poziomach systemu.
- **Zabezpieczenie dostępu:** Centralizacja danych konfiguracyjnych umożliwia skuteczniejsze zabezpieczenie tych informacji. Dzięki temu, istotne ustawienia i dane są chronione przed nieautoryzowanym dostępem z zewnątrz. W celu zapewnienia bezpieczeństwa, plik konfiguracyjny jest chroniony przez reguły zapisane w pliku **.htaccess**.

```
1  {
2      "ver": "",
3      "name": "",
4      "token": "",
5      "ads": "",
6      "convs": {
7          "img_min": true,
8          "js_min": true,
9          "css_min": true
10     },
11     "mariadb": {
12         "url": "",
13         "login": "",
14         "password": ""
15     },
16     "cache": "",
17     "tlr": [ "" ],
18     "dbinit": "",
19     "sched": true
20 }
```

Listing 7.1: Plik konfiguracyjny serwera skrajnego

Poszczególne elementy pliku konfiguracyjnego:

- **ver** - wersja oprogramowania: Przechowuje informacje o wersji oprogramowania serwera skrajnego. Jest to ważne dla administratorów do identyfikacji bieżącej wersji i ewentualnych aktualizacji.
- **name** - nazwa serwera: Określa nazwę serwera skrajnego, co ułatwia jego identyfikację w sieci.
- **token** - token autoryzacyjny: Używany do uwierzytelniania zapytań przychodzących. Poprawny token w nagłówku zapytania jest wymagany, aby serwer skrajny akceptował i przetwarzał żądanie.
- **ads** - adres operatora reklam: Pole używane w celu dystrybuowania reklam na domenach w pakiecie darmowym.
- **convs** - Optymalizatory treści: Zawiera informacje o włączonych optymalizatorach treści, takich jak **img_min**, **js_min**, **css_min**. Niektóre serwery mogą mieć ograniczoną moc obliczeniową i mogą wymagać wyłączenia niektórych konwerterów.
- **MariaDB** - dane logowania do bazy danych: Zawiera dane potrzebne do połączenia z bazą danych, takie jak URL, login i hasło. Mimo, że nazwa może wskazywać, że serwer skrajny obsługuje tylko połączenia z bazą danych MariaDb, to jednak jest to pozostałość po pierwotnej wersji pliku konfiguracyjnego. W aktualnej wersji serwer ma możliwość łączenia się z innymi systemami DBMS (*Database Management System*), nie tylko MariaDB. W pierwotnej wersji developerskiej aplikacja korzystała z connectora MariaDB, co związane z konfiguracją Dockera. Niemniej jednak, samo połączenie do bazy danych umożliwia komunikację z różnymi systemami.
- **cache** - lokalizacja zapamiętywanej zawartości: Określa lokalizację, w której ma być zapisywana lokalna zawartość.
- **tlr** - Reguły Top Level Rewrites: Zestaw ciągów znaków, dla modułu TLR.
- **dbinit** - inicjacja bazy danych: Wskazuje na ścieżkę do pliku SQL, który służy do inicjalizacji bazy danych.

Wspomniany już plik **.htaccess**, jest używany na serwerach Apache i odgrywa istotną rolę w konfiguracji serwerów skrajnych w systemie sCDN, biorąc pod uwagę zwłaszcza ich działanie na istniejącej infrastrukturze hostingowej. Jest to plik konfiguracyjny, który umożliwia lokalne zarządzanie ustawieniami serwera na poziomie

katalogu, co jest kluczowe, gdyż wiele usług hostingowych umożliwia jego modyfikację. Jednym z głównych zastosowań .htaccess w kontekście serwerów skrajnych jest przekierowywanie wszystkich żądań do jednego głównego pliku - **index.php**. Dzięki temu, możliwe jest centralne i uporządkowane zarządzanie żądaniami, co jest niezwykle ważne w kontekście efektywności i bezpieczeństwa. Plik ten pozwala również na blokowanie bezpośredniego dostępu do innych plików i katalogów, co jest szczególnie istotne dla plików konfiguracyjnych zawierających wrażliwe informacje. **RewriteRule** w pliku .htaccess niesie ze sobą konsekwencje, które utrudniają obsługę błędów na poziomie tego pliku. To oznacza, że nie można skonfigurować niestandardowych stron błędów bezpośrednio za pomocą .htaccess. W praktyce, obsługa błędów musi odbywać się na poziomie aplikacji.

```
1 RewriteEngine On
2
3 # Wersja dla fizycznej infrastruktury
4 # RewriteRule ^(.*)$ src/index.php [L]
5
6 #Wersja dla serwerów lokalnych w kontenerach Docker
7 RewriteRule ^(.*)$ index.php [L]
```

Listing 7.2: Plik konfiguracyjny serwera apache - .htaccess

502

Wystąpił problem podczas komunikacji z serwerem źródłowym!

6599d1ffc900c - 2024-01-06 22:19
© Marcin Ślusarczyk, Maciej Bandura qc-CDN 2023/2024

Rysunek 7.4. Widok komunikatu błędu zwracany przez serwer skrajny

7.6 Połączenie serwera z bazą danych

W PHP istnieje kilka podejść do nawiązywania połączenia z bazą danych, które różnią się między sobą pod względem funkcjonalności, wydajności i bezpieczeństwa. Najpopularniejsze metody to:

1. **MySQLi (MySQL Improved)** MySQLi jest rozszerzeniem PHP dedykowanym dla baz danych MySQL. Zapewnia programistom zarówno proceduralny, jak i obiektowy interfejs do pracy z bazą danych. MySQLi jest zalecaną opcją, gdy pracuje się z bazami danych MySQL, ponieważ oferuje funkcje niedostępne w starszych rozszerzeniach, takie jak przygotowane zapytania (prepared statements), które zwiększa bezpieczeństwo poprzez zapobieganie atakom SQL Injection.
2. **PDO (PHP Data Objects)** PDO to interfejs do baz danych, który umożliwia jednolite połączenie z różnymi systemami baz danych. PDO nie jest specyficzny dla jakiegokolwiek typu bazy danych, co czyni go bardziej elastycznym w porównaniu do MySQLi, który jest charakterystyczny dla MySQL. PDO również obsługuje przygotowane zapytania, zapewniając bezpieczeństwo i optymalizację.

W kontekście serwera skrajnego w systemie sCDN, zastosowano specjalnie zaprojektowaną klasę **qc_DBC** do zarządzania połączeniem z bazą danych. Ta klasa reprezentuje bardziej złożone podejście, oferując zaawansowane funkcje i zarządzanie połączeniem z bazą danych. W ramach dostosowania do specyfiki, wymagań i potrzeb serwera skrajnego, zdecydowano się na wykorzystanie metody **MySQLi** jako podstawy połączenia z bazą danych.

```
1 public function __construct ()  
2 {  
3     global $__config;  
4     $this->dbName = $__config->get("name");  
5     $this->dbAuth = $__config->get("mariadb");  
6     mysqli_report(MYSQLI_REPORT_OFF);  
7     $this->enable();  
8 }
```

Listing 7.3: Konstruktor klasy qc_dbc

Konstruktor klasy inicjalizuje połaczenie przy pomocy funkcji **enable**, korzystając z globalnych konfiguracji do pobrania danych uwierzytelniających. Metoda **exec** w klasie umożliwia wykonanie przygotowanych zapytań, co jest kluczowe dla bezpieczeństwa. Przygotowane zapytania przy pomocy metody **bind_param** minimalizują ryzyko **ataków SQL Injection**. Metoda ta obsługuje parametry zapytania i pozwala na łatwą manipulację danymi w bazie.

```
1 public function exec ($prep, ...$binds)
2 {
3     $stmt = $this->connector->prepare($prep);
4     if (is_bool($stmt))
5         qc_Error::render(500);
6
7     $type = "";
8     foreach ($binds as $param)
9         $type .= "s";
10
11    $stmt->bind_param($type, ...$binds);
12    if ($stmt->execute() == false)
13        return false;
14
15    $res = $stmt->get_result();
16    if (is_bool($res))
17        return [];
18
19    return $res->fetch_all(MYSQLI_ASSOC);
20 }
```

Listing 7.4: Metoda wykonująca zapytania do bazy danych

Pierwszym argumentem funkcji **exec** jest kod SQL, w którym miejsca parametrów są oznaczone znakami zapytania. Pozostałe argumenty funkcji są używane do przekazywania wartości, które zostaną podstawione w miejsce tych parametrów. Funkcja ta jest zaimplementowana jako funkcja ze **zmienną liczbą argumentów** (*variadic function*), co oznacza, że może przyjąć dowolną liczbę argumentów po pierwszym parametrze. Następnie funkcja próbuje przygotować zapytanie SQL za pomocą metody **prepare**, a następnie powiązać parametry zapytania za pomocą metody **bind_param**. Typy parametrów są określone na podstawie typu s (*string*), gdyż jest to typ uniwersalny i będzie działał ze wszystkimi typami danych wykorzystywanymi w aplikacji serwera skrajnego.

Jeśli wykonanie zapytania SQL nie powiedzie się (w przypadku operacji **INSERT** i **UPDATE**), funkcja zwraca **false**, wskazując na niepowodzenie wykonania zapytania. Jeśli wykonanie zapytania SQL zakończy się sukcesem, funkcja próbuje pobrać wynik za pomocą metody **get_result**. Jeśli wynik jest dostępny, zostanie zwrócony w formie tablicy asocjacyjnej (*MYSQLI_ASSOC*). Jeśli wynik nie jest dostępny, funkcja zwróci pustą tablicę.

```
1 $__dbc->exec("SELECT * FROM qc_Session WHERE sess = ?", $_COOKIE['qc_domain']);
```

Listing 7.5: Przykładowe wywołanie funkcji exec

```
1 /var/www/html/cdn/cdn.php:62:
2 array (size=1)
3   0 =>
4     array (size=3)
5       'sess' => string '6599b61e64c2e' (length=13)
6       'domain' => string 'domena.localhost' (length=16)
7       'expr' => string '2024-01-07 20:20:46' (length=19)
```

Listing 7.6: Przykładowe dane zwracane przez funkcję exec

W początkowej wersji klasy **qc_DBC**, stałe połączenia z bazą danych generowały problemy, szczególnie na hostingach z ograniczeniami przepustowości i szybkości obsługi bazy danych oraz z restrykcjami dotyczącymi maksymalnej liczby połączeń. Wykonywanie intensywnych blokowych operacji, takich jak pobieranie plików przez cURL czy zapisywanie ich do pamięci lokalnej serwera, choć nie angażowało bezpośrednio bazy danych, miało wpływ na efektywność zarządzania połączeniem z bazą. Te operacje były czasochłonne i w efekcie, utrzymywane połączenia bazodanowe były blokowane, co nie było efektywne ani potrzebne. Taki model prowadził do nieefektywnego wykorzystania zasobów serwera, co mogło skutkować wykorzystaniem dostępnej liczby połączeń do bazy danych. Ponadto, na wielu hostingach, możliwości zwiększenia limitu maksymalnej liczby połączeń z bazą danych były ograniczone, a modyfikacja kluczowych parametrów konfiguracyjnych bazy danych, takich jak **max_connections**, często była niemożliwa. Rozwiązaniem tego problemu było wprowadzenie metod **enable** i **disable** do zarządzania połączeniami bazodanowymi.

```
1 public function enable ()  
2 {  
3     $conn = @new mysqli  
4     (  
5         $this->dbAuth->url ,  
6         $this->dbAuth->login,  
7         $this->dbAuth->password,  
8         $this->dbName  
9     );  
10  
11    if ($conn->connect_error)  
12        exit;  
13  
14    $this->connector = $conn;  
15 }
```

Listing 7.7: Metoda tworząca połączenie z bazą danych

Metoda **enable** służy do otwierania połączenia z bazą danych tylko wtedy, gdy jest to konieczne, a **disable** do jego zamknięcia zaraz po zakończeniu wymaganej operacji. Dzięki temu, połączenia z bazą danych nie są utrzymywane niepotrzebnie podczas wykonywania operacji, które nie wymagają interakcji z bazą, jak pobieranie czy zapisywanie plików. To podejście znacznie poprawiło wydajność i stabilność serwerów skrajnych, pozwalając na lepsze zarządzanie zasobami i unikanie przeciążeń.

Klasa **qc_DBC** stanowi nieocenione narzędzie, które pozwala na uproszczenie procesu nawiązywania połączenia z bazą danych. Dzięki jej zastosowaniu, można uniknąć powtarzania tego samego kodu w wielu miejscach aplikacji, co prowadzi do znacznie czytelniejszego i bardziej modułowego kodu źródłowego. To z kolei ułatwia zarządzanie i utrzymanie aplikacji, a także przyczynia się do poprawy jej jakości.

7.7 Challenge page

Dzięki wprowadzeniu metody podobnej do stosowanej przez inne firmy świadczące usługi CDN, takiej jak Cloudflare, wprowadziliśmy stronę wyzwań, zwaną również jako strona CAPTCHA (*Captcha page*). Jest to skuteczna metoda poprawy zabezpieczeń i zarządzania dostępem do zawartości. Głównym celem strony wyzwań jest uwierzytelnienie użytkownika przed udzieleniem mu dostępu do żądanej zawartości. Mechanizm "*challenge*" sprawdza, czy zapytanie do serwera pochodzi od rzeczywistego użytkownika, korzystającego z przeglądarki internetowej, a nie od botów lub narzędzi automatyzujących zapytania (takich jak cURL czy Python). Działa on w następujący sposób:

1. Serwer, po otrzymaniu zapytania od klienta, aktywuje mechanizm "*challenge*", który polega na wysłaniu do klienta specjalnego skryptu.
2. Skrypt ten jest uruchamiany po stronie przeglądarki użytkownika i korzysta z operacji na **drzewie DOM** (*Document Object Model*) oraz języka JavaScript.
3. Informacje o kluczu sesji są zaszyfrowane i ukryte w elementach DOM strony internetowej, którą przeglądarka otrzymuje. Nie są one bezpośrednio dostępne dla użytkownika.
4. Skrypt JavaScript działający po stronie przeglądarki odczytuje i dekoduje te informacje, używając odpowiednich selektorów i operacji na drzewie DOM.
5. Jeśli proces dekodowania zakończy się sukcesem, klucz sesji zostaje zapisany w pliku cookies przeglądarki. To pozwala na przesyłanie klucza sesji w każdym kolejnym zapytaniu HTTP jako nagłówek cookies.
6. Kiedy klient ponownie wysyła zapytanie do serwera, klucz sesji jest odczytywany z pliku cookies i weryfikowany przez serwer, aby potwierdzić, że użytkownik jest prawdziwy.

Sam proces kodowania i dekodowania klucza sesji w mechanizmie "*challenge*" jest złożony i obejmuje kilka kroków:

1. Kodowanie klucza sesji w pakietach

- Klucz sesji jest podzielony na mniejsze jednostki - pakiety.
- Każdy pakiet jest reprezentowany przez element **div** o klasie "a".
- W atrybutach **data-b** i **data-c** umieszczane są dane związane z pakietem, które stanowią część klucza sesji. W pierwszym z nich umieszczony jest znak klucza sesji, a w drugim jego kolejność.

2. Doklejanie pakietów do DOM asynchronicznie

- Pakiety danych są asynchronicznie dołączane do drzewa DOM przy użyciu funkcji **setTimeout** przy wykorzystaniu metody **insertAdjacentHTML**.

```
1  setTimeout(() =>
2  {
3      document.body.insertAdjacentHTML('beforeend', '<div class="a" data-b="Y"
4                                     → data-c="8"></div>');
5      document.querySelector('.title').innerHTML = `Trwa weryfikacja
6                                     → przeglądarki ${1 + Math.round(((1000) - (600)) / 1000)}s`;
7  }, 600);
```

Listing 7.8: Dynamiczne modyfikowanie DOM

- Pakiety są dodawane w określonych interwałach czasowych.

3. Dekodowanie klucza sesji

- Proces dekodowania klucza sesji zostaje uruchomiony po upływie określonego czasu (kiedy wszystkie pakiety zostaną doklejone do DOM).
- Wartość **expr**, reprezentująca datę ważności pliku cookie, jest ustawiana na 24 godziny od aktualnej daty (analogicznie jak w instancji sesji w bazie danych).
- Funkcja dekoduje klucz sesji na podstawie zawartości pakietów w drzewie DOM. Pakiety są sortowane na podstawie atrybutu **data-c** w celu zachowania odpowiedniej kolejności.
- Następnie dane z atrybutów **data-b** pakietów są łączone w jeden ciąg znaków, który jest dekodowany za pomocą funkcji **atob** (dekodowanie z Base64).
- Zdekodowany klucz sesji jest zapisywany jako plik cookie o nazwie **qc_domain** z datą wygaśnięcia ustawioną na kolejny dzień.

```

1   setTimeout(() =>
2 {
3     const expr = new Date();
4     expr.setDate(expr.getDate() + 1);
5
6     document.cookie = "qc_domain=" + atob(Array
7       .from(document.querySelectorAll('.a'))
8       .sort((a, b) => parseInt(a.dataset['c']) - parseInt(b.dataset['c']))
9       .map((n) => n.dataset['b'])
10      .join('')) + "; expires=" + expr.toUTCString() + "; path=/";
11
12     window.location.href = "/";
13 }, 10000)

```

Listing 7.9: Funkcja dekodująca klucz sesji

- Po zapisaniu klucza sesji użytkownik zostaje przekierowany na docelową stronę.

W mechanizmie "challenge" zarządzanie sesjami w bazie danych (tabela **qc_Session**) jest kluczowym elementem zapewnienia bezpieczeństwa i efektywnego zarządzania przestarzałymi sesjami. Sesje mają określoną datę ważności zapisaną w polu **expr**, która określa, kiedy sesja jest uważana za przestarzałą i powinna być usunięta. Istnieją dwa główne sposoby usuwania przestarzałych sesji:

1. Event scheduler w bazie danych

- Event ten jest odpowiedzialny za usuwanie wszystkich sesji, których data ważności (**expr**) przekroczyła aktualny czas.
- Jest on uruchamiany raz na 24 godziny.

```

1 CREATE EVENT qc_SessWatch
2 ON SCHEDULE EVERY 1 DAY
3 STARTS CURRENT_TIMESTAMP
4 DO
5 BEGIN
6   DELETE FROM `qc_Session` WHERE expr < CURRENT_TIMESTAMP;
7 END

```

Listing 7.10: Event usuwający przestarzałe sesje

2. Zarządzanie sesjami po stronie aplikacji

W przypadku braku możliwości użycia zdarzeń planowych w bazie danych, w mechanizmie "challenge" zastosowano alternatywną metodę zarządzania sesjami. Funkcja **tryRemoveOldSessions** jest wywoływana po stronie aplikacji, ale nie za każdym razem, gdy użytkownik wywołuje zapytanie. Funkcja ta przy każdym nowym tworzeniu sesji, losowo decyduje, czy ma usunąć przestarzałe sesje na podstawie pewnej szansy, tj. średnio raz na 100 zapytań. Nie jest to najlepsze ani deterministyczne rozwiązanie, jednakże specyfika projektu i ograniczenia nakładane przez zewnętrznych usługodawców wymagają pewnych obejść.

```
1  private function tryRemoveOldSessions ()  
2  {  
3      global $_config;  
4      global $_dbc;  
5  
6      if ($_config->get('sched'))  
7          return;  
8  
9      if (rand(0, 99) != 0)  
10         return;  
11  
12     $_dbc->query("DELETE FROM `qc_Session` WHERE expr < CURRENT_TIMESTAMP");  
13 }
```

Listing 7.11: Funkcja usuwająca przestarzałe sesje

Jest to elastyczne podejście, które pozwala na zarządzanie sesjami w różnych środowiskach. Można korzystać z **event scheduler** w bazie danych, jeśli jest dostępny, lub z alternatywnej metody opartej na losowym usuwaniu przestarzałych sesji, jeśli nie ma takiej możliwości. Dzięki temu zapewnia się sprawną obsługę sesji użytkowników i utrzymanie bazy danych w czystości, jednocześnie rozważając ograniczenia narzucone przez niektórych usługodawców hostingowych.

Dzięki temu mechanizmowi możliwe jest skuteczne uwierzytelnianie prawdziwych użytkowników i minimalizowanie zapytań generowanych przez boty, które nie działają jak przeglądarki internetowe. Jest to istotny element w zapewnieniu bezpieczeństwa i wydajności serwera.

7.8 Moduły serwera skrajnego

7.8.1 Moduł inicjalizujący świadczenie treści

Moduł **qc_CDN** w serwerze skrajnym systemu sCDN pełni rolę centralną, będąc głównym modułem, od którego rozpoczyna się cały proces świadczenia treści. Jego działanie jest nie tylko fundamentalne dla rozpoczęcia procesu dostarczania treści, ale także wpływa na funkcjonowanie innych modułów systemu. Na samym początku, qc_CDN zajmuje się weryfikacją sesji użytkownika, co jest kluczowe dla zapewnienia bezpieczeństwa. Informacje o domenie, pobierane w tym module, są wykorzystywane przez pozostałe komponenty systemu. W trakcie prac nad projektem, moduł qc_CDN przeszedł znaczne zmiany.

```
1 private function readDomainFromCookie ($path)
2 {
3     global $_dbc;
4     if (!isset($_COOKIE['qc_domain']))
5         qc_Error::render(400);
6
7     $sess = $_dbc->exec("SELECT * FROM qc_Session WHERE sess = ?",
8         $_COOKIE['qc_domain']);
9     if (count($sess) != 1)
10        qc_Error::render(401);
11
12     return (object) array
13     (
14         "domain" => $sess[0]['domain'],
15         "path" => $path
16     );
17 }
```

Listing 7.12: Weryfikacja i pobranie informacji o sesji

W obecnej wersji oprogramowania w pliku cookie przechowywany jest klucz sesji, przy pomocy którego z tabeli **qc_Session** pobierana jest nazwa domeny świadczonej użytkownikowi. Przed dodaniem systemu "antybotowego" nazwa domeny przechowywana była bezpośrednio w pliku cookie. Jednakże w początkowych fazach projektu, nazwa domeny przechowywana była w ścieżce zapytania, co generowało wiele problemów, chociażby skomplikowane operacje przepisywania linków czy też niszczenie logiki w skryptach js.

```
1 private function convertPathToDomainPath ($path)
2 {
3     $toks = parse_url("http://" . substr($path, 1));
4     if (is_bool($toks) || array_key_exists("host", $toks) == false)
5         exit(header('HTTP/1.1 400 Bad Request'));
6
7     if (isset($toks['path']) == false)
8         $toks['path'] = '/';
9
10    return (object) array
11    (
12        "domain" => $toks['host'],
13        "path" => $toks['path']
14    );
15}
```

Listing 7.13: Pierwsza wersja przekazywania informacji o domenie

W pierwotnej wersji serwera skrajnego systemu sCDN, nazwa domeny była identyfikowana na podstawie pierwszego segmentu ścieżki zapytania. Mimo skomplikowanej konstrukcji konwertera, który miał za zadanie przekształcać adresy źródłowe na adresy serwera skrajnego, czasami napotykano na trudności z poprawnym dodaniem nazwy domeny do ścieżki. W związku z tym, zaimplementowano dodatkowy mechanizm pozyskiwania nazwy domeny. W języku PHP istnieje predefiniowana globalna tablica asocjacyjna `$_SERVER` zawierająca informacje o serwerze, jednym z kluczy tej tablicy jest `$_SERVER['HTTP_REFERER']`, który przechowuje adres URL strony, z której pochodzi żądanie. Dzięki temu, możliwe jest uzyskanie adresu domeny nawet w sytuacjach, gdy aktualny adres URL nie zawiera takich danych. Po określeniu nazwy domeny, adres ten może być następnie dołączony do obecnego adresu URL, co umożliwia przekierowanie użytkownika na odpowiedni adres, zapewniając poprawność i spójność przetwarzania żądań. Rozwiązanie, polegające na wykorzystaniu **HTTP_REFERER** do określenia nazwy domeny, jest skuteczne w przypadku prostych, statycznych stron internetowych, które nie wykorzystują intensywnie JavaScript. Jednakże, w przypadku bardziej złożonych witryn internetowych, które intensywnie korzystają z dynamicznych skryptów JavaScript, to podejście może nie być wystarczające. Takie strony często zawierają linki i skrypty, których adresy nie mogą być łatwo przekonwertowane przez konwertery serwera skrajnego. W rezultacie, nawet zabezpieczenie za pomocą **HTTP_REFERER** nie jest w stanie poradzić sobie z takimi wyzwaniami.

```
1 if (isset($_SERVER['HTTP_REFERER']) == false)
2     qc_forbidden();
3
4 $toks = parse_url($_SERVER['HTTP_REFERER']);
5 if (isset($toks['host']) == false || $toks['host'] != $_SERVER['SERVER_NAME'])
6     qc_forbidden();
7
8 $toks = parse_url("http://" . $toks['path']);
9 $domain = $toks['host'];
10 $toks = parse_url($_SERVER['REQUEST_URI']);
11 $red = $_SERVER['REQUEST_SCHEME'] . "://" . $_SERVER['SERVER_NAME'] . "/" .
12     ↳ $domain . $toks['path'];
12 exit(header("Location: $red"));
```

Listing 7.14: Wykorzystanie HTTP_REFERER

Problem pojawia się w przypadku skryptów JavaScript, które bazują na bieżącej ścieżce URL do prawidłowego działania. Mechanizm, który polega na przekierowaniu użytkownika lub manipulacji ścieżką, może prowadzić do nieprawidłowego działania tych skryptów. Dlatego też, w kontekście bardziej zaawansowanych stron internetowych, wymagane są bardziej złożone i zaawansowane metody obsługi żądań, aby zapewnić sprawne i poprawne działanie serwera skrajnego.

Decyzja o przejściu na przechowywanie informacji o domenie w plikach cookies okazała się być słusznym krokiem w rozwoju serwera skrajnego. To rozwiązanie poprawiło kompatybilność i funkcjonalność serwera, szczególnie w kontekście złożonych stron internetowych. Poprzez przechowywanie informacji o domenie w pliku cookie, serwer skrajny zyskał możliwość bardziej efektywnej i niezawodnej obsługi żądań, niezależnie od złożoności strony i użytych w niej skryptów JavaScript. Dzięki temu pojedściu, znaczna większość testowanych stron, które wcześniej napotykały problemy w działaniu, zaczęła funkcjonować prawidłowo. Rozszerzenie tego mechanizmu stanowi kolejny ważny krok w poprawie działania serwera skrajnego. Wprowadzenie sesji dostarcza dodatkowej warstwy ochrony, co jest kluczowe w kontekście bezpieczeństwa i skuteczności świadczonych usług. Sesje umożliwiają serwerowi skrajnemu nie tylko przechowywanie informacji o domenie, ale również weryfikację, czy żądania pochodzą od prawidłowo działających przeglądarek, a nie od automatycznych botów. Jest to szczególnie istotne w dzisiejszym środowisku internetowym, gdzie boty często generują sztuczny ruch, co może prowadzić do niepotrzebnego obciążenia serwera i potencjalnie do jego nadużycia.

7.8.2 Moduł zarządzający lokalną zawartością na serwerze skrajnym

Moduł zarządzania lokalną zawartością (**qc_Cache**), pełni istotną rolę w zwiększeniu wydajności procesu dostarczania treści przez serwer skrajny. Jego działanie skupia się na zapisie i odczytanie **lokalnych kopii** treści pobranych z serwera źródłowego. Dzięki temu, raz pobrana zawartość może być ponownie wykorzystywana bez konieczności ponownego łączenia się z serwerem źródłowym. Zawartość zapisana lokalnie na serwerze skrajnym obejmuje różnorodne rodzaje treści, począwszy od plików multimedialnych, poprzez skrypty czy arkusze stylów, które są często żądane przez przeglądarki użytkowników.

W systemie sCDN, użytkownik ma możliwość kontroli nad procesem cache'owaniem poprzez definiowanie reguł, które są przypisane do konkretnej domeny i pozwalają na określenie, czy dana zawartość - na podstawie ścieżki dostępu lub typu MIME - ma być przechowywana w pamięci lokalnej. Dzięki wykorzystaniu wyrażeń regularnych, użytkownicy mogą tworzyć bardziej skomplikowane zasady, co daje im większą elastyczność i precyzję w zarządzaniu treściami serwowanymi przez ich domeny.

Integracja tego modułu z bazą danych pozwala na zarządzanie i ewidencjonowanie zasobów, łącząc każdy plik z odpowiednią domeną. W rekordzie każdego pojedynczego rekordu, przechowywanego w tabeli **qc_Content**, gromadzone są istotne informacje, które są kluczowe dla efektywnego zarządzania i dostarczania treści. Podstawowym elementem jest nazwa pliku, generowana jako **hash MD5** oryginalnej ścieżki dostępu. W połączeniu z nazwą domeny gwarantuje to unikalność w obrębie serwera i pozwala na precyzyjne powiązanie pliku z konkretną domeną. Kluczowym aspektem przechowywanych danych jest rodzaj treści, określany poprzez **typ MIME**. Ta informacja jest niezbędna do prawidłowego poinformowania przeglądarki o rodzaju świadczonej treści. Jest to szczególnie ważne, ponieważ bez odpowiedniego nagłówka typu MIME, wiele rodzajów treści, jak skrypty czy zdjęcia, mogą nie działać prawidłowo lub mogą wystąpić problemy z ich wyświetlaniem. Różne przeglądarki mogą różnie interpretować treści bez określonego typu MIME, co może prowadzić do niekonsistentnego zachowania. W celu zapewnienia deterministycznego i standardowego działania, moduł qc_Cache zawsze określa typ zwracanego kontentu w nagłówku odpowiedzi. Ponadto, w rekordach plików przechowywane są także ścieżka dostępu i rozmiar pliku. Ta pierwsza, ma charakter głównie informacyjny, zaś rozmiar pliku jest ważny w procesie obliczenia dostępnego miejsca na pliki, który jest zależny od wybranego planu.

```
1 /**
2  * Sprawdzenie czy w ewidencji z zachowanych plików
3  * znajduje się ta treść + zapamiętanie informacji
4  * o niej na później.
5 */
6
7 public function hasContent ()
8 {
9     if ($_SERVER["REQUEST_METHOD"] != "GET")
10        return false;
11
12    global $_dbc;
13    $r = $_dbc
14    ->exec("SELECT * FROM qc_Content WHERE domain = ? AND name = ?", $this->domain,
15           hex2bin($this->md5path));
16
17    if (count($r) == 0)
18        return false;
19
20    $this->dbInfo = (object) $r[0];
21    $this->dbInfo->cached = true;
22
23    return true;
}
```

Listing 7.15: Integracja modułu z bazą danych

W ramach systemu sCDN dla każdej domeny na serwerze skrajnym tworzony jest dedykowany katalog, którego nazwa jest generowana jako hash MD5 nazwy domeny. To podejście zapewnia unikalność i organizację przechowywanych danych, co jest kluczowe dla efektywnego zarządzania zawartością. Lokalizacja, gdzie przechowywane są katalogi domen, jest zdefiniowana w pliku konfiguracyjnym serwera, w polu oznaczonym jako '**cache**'. Podobnie jak w przypadku katalogów domen, nazwy plików z zawartością domeny są również przechowywane jako hashe nazw plików. Taki sposób przechowywania nazw plików gwarantuje ich unikalność oraz ułatwia efektywne odnajdywanie treści. Każdy plik zapisany na serwerze jest stale powiązany z ewidencją w bazie danych. Jest to fundamentalne dla zapewnienia spójności i integralności danych. Taka hierarchia i metoda organizacji danych zapobiega sytuacjom, w których plik istnieje na serwerze bez odpowiedniego wpisu w bazie danych lub odwrotnie.

```
1 public function save ($content)
2 {
3     global $__dbc;
4     global $__config;
5     $this->checkCachingRules();
6     if ($this->cachingEnabled == false)
7         return;
8
9     /* sprawdzenie quota na dysku */
10    $sumKb = $__dbc->exec("SELECT * FROM qc_Domains WHERE domain = ?",
11                           $this->domain)[0]['size'];
12    $maxKb = ($this->free ? 10 : 100) * 1000000;
13
14    if ($sumKb + strlen($content) > $maxKb)
15        return;
16
17    $__dbc->exec("UPDATE qc_Domains SET size = ? WHERE domain = ?",
18                  $sumKb + strlen($content), $this->domain);
19
20    $__dbc->exec("INSERT INTO qc_Content VALUES (?, ?, ?, ?, ?, ?)",
21                  hex2bin($this->md5path), $this->domain, $this->path, $this->mime,
22                  strlen($content));
23
24    $__dbc->disable();
25    $dir = $this->cacheDir . $this->md5domain;
26    if (file_exists($dir) == false)
27        mkdir($dir, 0777, true);
28 }
```

Listing 7.16: Zapisanie pliku na serwerze skrajnym

Zarządzanie przestrzenią dyskową wykorzystywaną przez poszczególne domeny w systemie sCDN to kwestia wymagająca szczególnej uwagi. W idealnym przypadku, w znormalizowanej bazie danych, obliczenie wykorzystanego miejsca przez daną domenę wymagałoby przeprowadzenia operacji sumowania rozmiarów wszystkich plików powiązanych z tą domeną. Jednakże, taka operacja, zwłaszcza w przypadku częstych zmian i dużej liczby plików, może być czasochłonna i mało wydajna. W odpowiedzi na to wyzwanie, zostało zastosowane zdenormalizowanie bazy danych w zakresie przechowywania informacji o wykorzystanym miejscu. W tabeli **qc_Domains** wprowadzono kolumnę **size**, która przechowuje łączny rozmiar wszystkich plików należących

do danej domeny. To podejście znacznie upraszcza i przyspiesza proces obliczania wykorzystanego miejsca, ponieważ nie wymaga przeprowadzania złożonych operacji agregacji za każdym razem, gdy potrzebne są takie informacje. W momencie dodawania nowego pliku do domeny, wartość **size** w tabeli **qc_Domains** jest automatycznie zwiększana o rozmiar tego pliku. Analogicznie, przy usuwaniu pliku, wartość ta jest odpowiednio zmniejszana. Dzięki temu, informacje o wykorzystanej przestrzeni dyskowej są na bieżąco aktualizowane i odzwierciedlają rzeczywisty stan wykorzystania zasobów przez domenę.

7.8.3 Moduł przekazywania żądań do serwera źródłowego

Tunelowanie jest procesem, w którym żądania klienta są przekazywane do serwera źródłowego. Choć z pozoru prosty, niesie ze sobą szereg komplikacji technicznych, które muszą być starannie zarządzane. Jednym z głównych wyzwań jest przekazywanie informacji zawartych w plikach cookies, nagłówkach oraz parametrach i ciałach żądań. W tunelowaniu serwer skrajny pełni rolę **pośrednika** między użytkownikiem a serwerem źródłowym, co wymaga dokładnego przekazywania wszystkich niezbędnych informacji, aby zapewnić **spójność** i **ciągłość żądania**. Pliki cookies są kluczowe dla zachowania stanu sesji użytkownika, a nagłówki HTTP przenoszą ważne informacje o typie zapytania, typie treści czy innych metadanych, które są niezbędne do prawidłowego przetworzenia żądania przez serwer źródłowy. Dodatkowo, przekazywanie ciała żądania, które może zawierać dane formularzy lub inne informacje przesyłane przez użytkownika, musi odbywać się w sposób przejrzysty i bezpieczny. Wszystkie te elementy muszą być skrupulatnie zarządzane przez serwer skrajny, aby zapewnić, że żądanie jest prawidłowo przetwarzane przez serwer źródłowy, a odpowiedź odpowiednio dostarczana użytkownikowi. Te komplikacje wymagają solidnej logiki przetwarzania i wysokiej dokładności w implementacji mechanizmów tunelowania.

W procesie tunelowania pierwszym krokiem jest odtworzenie ścieżki dostępu do zasobu znajdującego się na serwerze źródłowym. To działanie jest kluczowe dla zapewnienia, że żądanie zostanie poprawnie przekazane i przetworzone. Po odtworzeniu ścieżki, aktywowany jest moduł Top Level Rewrites (**TLR**) - zostanie on opisany w dalszej części pracy.

```
1 private function fetchExternalContent ()  
2 {  
3     global $__config;  
4     global $cookies;  
5  
6     $url = $this->servInfo->origin . $this->servInfo->path;  
7  
8     $tlr = $__config->get("tlr");  
9     foreach ($tlr as $key)  
10        $url = str_replace("-$key-", $key, $url);  
11  
12    if ($_SERVER["REQUEST_METHOD"] == "GET")  
13    {  
14        $params = [];  
15        foreach ($_GET as $key => $val)  
16            $params[] = "$key=$val";  
17        if (count($params) > 0)  
18            $url .= "?" . implode("&", $params);  
19    }  
20    ...
```

Listing 7.17: Metoda tunelująca treści cz. 1

W procesie tunelowania wykorzystuje się bibliotekę cURL w PHP, aby pobrać dane z serwera źródłowego. Proces ten rozpoczyna się od inicjalizacji sesji cURL, gdzie tworzony jest uchwyt (*handle*) **\$ch** z adresem URL serwera źródłowego. Następnie, ustawia się opcję cURL **CURLOPT_CUSTOMREQUEST**, aby dopasować metodę żądania HTTP do metody wywołującej na serwerze skrajnym, co zapewnia spójność między wysyłanym żądaniem a oczekiwana metodą na serwerze źródłowym. Jeśli metoda żądania nie jest metodą GET, ustawia się dane żądania POST za pomocą **CURLOPT_POSTFIELDS**. Następnie, konfiguruje się cURL, aby zwracał wyniki operacji bezpośrednio do zmiennej, zamiast je wypisywać. Dodatkowo, ustawia się identyfikator agenta użytkownika **CURLOPT_USERAGENT** na podstawie nagłówka żądania, aby serwer źródłowy mógł zidentyfikować typ klienta. Oprócz tego, ustawia się limit czasu żądania **CURLOPT_TIMEOUT_MS** na określoną wartość, aby uniknąć nadmiernie długiego oczekiwania na odpowiedź. Ważnym ustaleniem jest również **CURLOPT_COOKIESESSION**, które zarządza sesjami cookies, oraz **CURLOPT_FOLLOWLOCATION**, umożliwiające automatyczne podążanie za przekierowaniami, co jest kluczowe w przypadku, gdy serwer źródłowy odpowiada przekierowaniem na inne URL.

```
1 $ch = curl_init($url);
2
3 curl_setopt($ch, CURLOPT_CUSTOMREQUEST, $_SERVER["REQUEST_METHOD"]);
4
5 if ($_SERVER["REQUEST_METHOD"] != "GET")
6     curl_setopt($ch, CURLOPT_POSTFIELDS, $_POST);
7
8 curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
9 curl_setopt($ch, CURLOPT_USERAGENT, $_SERVER['HTTP_USER_AGENT']);
10 curl_setopt($ch, CURLOPT_TIMEOUT_MS, 5000);
11 curl_setopt($ch, CURLOPT_COOKIESESSION, true);
12 curl_setopt($ch, CURLOPT_FOLLOWLOCATION, true);
```

Listing 7.18: Metoda tunelująca treści cz. 2

Istotnym elementem procesu tunelowania jest zarządzanie plikami cookies, co realizowane jest za pomocą funkcji **copyOverCookies**. Ta funkcja ma za zadanie odczytać pliki cookies z odpowiedzi serwera źródłowego i przekazać je użytkownikowi. Działanie funkcji rozpoczyna się od wykrycia linii nagłówka **Set-Cookie** w odpowiedzi serwera źródłowego. Gdy taki nagłówek zostanie znaleziony, funkcja odczytuje nazwę i wartość cookie, a następnie sprawdza, czy cookie o tej samej nazwie już nie został ustawiony. Jeśli nie, ustawia je w przeglądarce użytkownika, z określonymi parametrami, takimi jak czas wygaśnięcia, ścieżka, domena, oraz atrybuty bezpieczeństwa **secure** i **httponly**. Po ustaleniu funkcji copyOverCookies jako **CURLOPT_HEADERFUNCTION**, następuje proces zbierania i przesyłania istniejących plików cookies użytkownika. Kod iteruje przez każdy plik cookie przechowywany w przeglądarce użytkownika, tworząc listę ciągów tekstowych w formacie klucz=wartość. Ta lista jest następnie łączona i przesyłana jako nagłówek Cookie w żądaniu cURL do serwera źródłowego. Dzięki temu mechanizmowi, serwer skrajny zapewnia, że wszystkie istotne informacje związane z sesją użytkownika, takie jak stan logowania, preferencje czy inne dane sesji przechowywane w cookies, są prawidłowo przekazywane między użytkownikiem a serwerem źródłowym, co jest kluczowe dla spójności i funkcjonalności serwisów internetowych.

```
1 function copyOverCookies ($ch, $headerLine)
2 {
3     global $cookie_cache;
4     if (preg_match('/^Set-Cookie:\s*([^;]*);?mi', $headerLine, $cookie) == 1)
5     {
6         $toks = explode("=", $cookie[1]);
7         if (array_key_exists($toks[0], $cookie_cache) == false)
8         {
9             $cookie_cache[$toks[0]] = $toks[1];
10
11             setcookie($toks[0], $toks[1], array('expires' => time() + 3600, 'path' =>
12             '/'), 'domain' => $_SERVER['HTTP_HOST'], 'secure' => true, 'httponly' =>
13             false, 'samesite' => 'None'));
14         }
15     }
16
17     return strlen($headerLine);
18 }
19 curl_setopt($ch, CURLOPTHEADERFUNCTION, "copyOverCookies");
20
21 $cookies = array();
22 foreach ($_COOKIE as $key => $val)
23     $cookies[] = "$key=$val";
24
25 curl_setopt($ch, CURLOPTCOOKIE, implode(";", $cookies));
```

Listing 7.19: Metoda tunelująca treści cz. 3

Po ustawieniu wszystkich parametrów, proces tunelowania w systemie sCDN kontynuowany jest przez wykonanie funkcji **curl_exec**, która realizuje żądanie HTTP za pomocą skonfigurowanej sesji cURL. Wynik tego żądania jest przechowywany w polu **\$this->externalContent**. Następnie, przeprowadzane są sprawdzenia, aby upewnić się, że żądanie zostało wykonane poprawnie i odpowiedź jest prawidłowa. Jeśli wystąpi błąd podczas wykonywania żądania cURL, wywoływana jest funkcja obsługi błędów, która w tym przypadku generuje błąd **502**. Dodatkowo, sprawdzany jest status odpowiedzi HTTP, aby stwierdzić, czy nastąpiło przekierowanie (kody **301** lub **302**). W przypadku wykrycia przekierowania, adres URL, na który ma nastąpić przekierowanie, jest pobierany, a następnie realizowane jest przekierowanie przez funkcję **tunelRedirect**. Po zakończeniu tego procesu, sesja cURL jest zamknięta.

Kluczowe jest także wczytanie typu MIME (typu zawartości) z odpowiedzi, co jest realizowane poprzez funkcję `curl_getinfo($ch, CURLINFO_CONTENT_TYPE)`. Ta informacja jest używana do ustawienia nagłówka **Content-Type** w odpowiedzi serwera skrajnego, co jest istotne dla poprawnego zinterpretowania treści przez przeglądarkę użytkownika. Po wykonaniu wszystkich tych działań, sesja cURL jest bezpiecznie zamknięta przez wywołanie `curl_close($ch)`.

```
1 private function tunelRedirect ($url)
2 {
3     $toks = parse_url($url);
4     $red = $_SERVER['REQUEST_SCHEME'] . "://" . $_SERVER['HTTP_HOST'] . "/" .
5           ↵ $this->servInfo->domain . $toks["path"];
6     header("Location: $red");
7     exit;
}
```

Listing 7.20: Metoda tunelująca treści cz. 4

Podczas tunelowania w systemie sCDN, przekierowania stanowią istotną komplikację. Początkowo, serwer skrajny musiał śledzić przekierowania, aby prawidłowo obsłużyć proces dostarczania treści. Oznaczało to konieczność podążania za łańcuchem przekierowań wysyłanych przez serwer źródłowy, co było skomplikowane i czasochłonne. Jednakże, w celu usprawnienia i optymalizacji tego procesu, zastosowano inne podejście: modyfikowanie adresu w nagłówku przekierowania na adres serwera skrajnego. Dzięki tej metodzie, zamiast podążać za każdym przekierowaniem, serwer skrajny dynamicznie zmienia adres URL w nagłówku przekierowania, kierując użytkownika bezpośrednio do odpowiedniego zasobu. Ta technika znacznie upraszcza proces obsługi przekierowań, redukując obciążenie serwera skrajnego i przyspieszając czas dostępu do treści dla użytkowników. Poprzez bezpośrednie przekierowanie użytkownika na serwer skrajny, złożoność zarządzania przekierowaniami jest minimalizowana, a cały proces staje się bardziej wydajny i niezawodny.

Ten proces nie tylko zapewnia efektywne pobieranie i przekazywanie treści, ale także gwarantuje, że wszystkie kluczowe aspekty żądania, takie jak błędy, przekierowania i typy zawartości, są odpowiednio obsługiwane.

7.8.4 Moduł modyfikacji świadczonej treści

Jak zostało wspomniane, treści obsługiwane przez programowy CDN muszą przechodzić proces modyfikacji, aby zapewnić ich prawidłowe wyświetlanie i funkcjonowanie. Podstawowym zadaniem tego modułu jest przepisywanie linków (odnośników) na adresy serwera skrajnego. Dzięki temu, użytkownik, który łączy się z serwisem, otrzymuje treści dostosowane pod serwer skrajny. Poza przepisywaniem linków, moduł ten zajmuje się również wstawianiem specjalnego skryptu, który umożliwia komunikację z sandboxem. W planie płatnym, do świadczonych treści doklejany jest skrypt dodający reklamy popup. Idąc za nowoczesnymi trendami w branży CDN, proces modyfikacji treści został rozszerzony o dodatkowe narzędzia optymalizacyjne. Pojedynczą metodę modyfikującą treści nazywamy **konwerterem**. Obejmują one kompresory zdjęć, które redukują rozmiar plików graficznych bez znacznego wpływu na ich jakość, oraz minifikatory plików CSS i JavaScript, które minimalizują ich rozmiar poprzez usunięcie zbędnych znaków, takich jak białe znaki czy komentarze. Konwerter przyjmuje oryginalną treść i przetwarza ją w charakterystyczny dla swojej specyfiki sposób i zwraca. Te narzędzia przyczyniają się do dalszego przyspieszenia ładowania się stron, poprawy wydajności. Podobnie jak w przypadku zasad zapisywania lokalnej zawartości na serwerach skrajnych, w tym module, istnieje możliwość szczegółowej parametryzacji reguł aplikowania poszczególnych konwerterów. Poza standardowymi polami (selektor, priorytet), pojedyncza reguła zawiera dodatkową informację, czy treść ma zostać zmodyfikowana przez konkretny konwerter.

Warto zaznaczyć, że ze względu na charakterystykę budowy stron internetowych oraz działanie sieci sCDN pewne konwertery w zależności od rodzaju treści będą wywoływane prawie zawsze. Aby uprościć proces wstępnej konfiguracji i nie powieść podstawowych zasad, system sCDN posiada listę predefiniowanych reguł twardo zapisanych w kodzie (*hard-coded*), które są automatycznie aplikowane do treści. Mimo to, użytkownicy mają możliwość nadpisania tych zasad za pomocą własnych konfiguracji.

```
1 private $rewriteRules =
2 [
3     "URL_REPL",
4     "HT_SANDBOX",
5     "IMG_MIN",
6     "JS_MIN",
7     "CSS_MIN"
8 ];
9
10 private $defaultRules =
11 [
12     "/text\\html*/" => ["HT_SANDBOX", "URL_REPL"],
13     "/text\\css*/" => ["URL_REPL"],
14     "/application\\javascript*/" => ["URL_REPL"]
15 ];
```

Listing 7.21: Tablica konwerterów oraz lista predefiniowanych reguł

W klasie modułu **qc_Conv**, znajduje się tablica, w której każdy konwerter jest przypisany do unikalnego indeksu. Te indeksy są używane do kojarzenia konkretnych konwerterów z odpowiednimi regułami w bazie danych. Warto zaznaczyć, iż w tablicy tej nie jest ujęty konwerter **HT_ADS**, gdyż jego konfiguracja jest niedostępna dla zwykłych użytkowników systemu.

W tablicy asocjacyjnej **\$defaultRules** przechowywane są predefiniowane reguły, jako klucz przetrzymywane jest wyrażenie regularne, służące do wyłapywania konkretnych typów MIME treści. Jako wartości przechowywane są listy kolejno aplikowanych konwerterów. Reguły te mają niższy priorytet, od tych definiowanych przez użytkowników.

Konwerter **URL_REPL** w systemie sCDN pełni prostą, lecz niezwykle istotną rolę. Jego głównym zadaniem jest przepisanie adresów URL serwera źródłowego na adresy serwera skrajnego. To działanie jest kluczowe dla funkcjonowania stron internetowych na serwerach skrajnych, ponieważ zapewnia, że wszelkie odwołania do zasobów (takich jak obrazy, skrypty, arkusze stylów) kierują użytkownika do odpowiednich treści na serwerze skrajnym. Ponadto, konwerter uwzględnia reguły TLR (*Top Level Rewrites*), które zostały wcześniej opisane. Są to reguły służące do przepisywania ścieżek na najwyższym poziomie.

W początkowych etapach rozwoju projektu, działanie konwertera URL_REPL było znacznie bardziej złożone niż obecnie. Jednym z głównych wyzwań była konieczność zawarcia nazwy domeny w adresie serwera skrajnego, co wymagało bardziej skomplikowanej logiki przetwarzania. Konwerter musiał analizować kod HTML, wykrywać różne tagi i atrybuty zawierające linki, a następnie w tych miejscach dokonywać modyfikacji adresu URL, aby wskazywały one na serwer skrajny. Takie podejście miało jednak swoje ograniczenia. Przede wszystkim skoncentrowane było wyłącznie na dokumentach HTML, co oznaczało, że skrypty JavaScript i arkusze stylów CSS były pomijane w procesie przepisywania URL-i. To prowadziło do problemów w działaniu niektórych stron internetowych, szczególnie tych, które intensywnie wykorzystywały treści i zasoby załadowane za pomocą JavaScript i CSS. W związku z tym, obecne rozwiązanie oparte wyłącznie na prostym przepisywaniu ciągów znaków w kodzie HTML okazałoby się niewystarczające dla pełnej i efektywnej funkcjonalności serwera skrajnego.

```
1 private function HT_REPL ()
2 {
3     error_reporting(~E_WARNING);
4     $dom = new DOMDocument('1.0', 'utf-8');
5     @$dom->loadHTML(mb_convert_encoding($this->content, 'HTML-ENTITIES', 'UTF-8'));
6     $dom->encoding = 'utf-8';
7     $xpath = new DOMXPath($dom);
8     $tags = array('a' => 'href', 'area' => 'href', 'audio' => 'src', ...);
9
10    $cdn = $_SERVER['REQUEST_SCHEME'] . "://" . $_SERVER['HTTP_HOST'] . "/" .
11        → $this->servInfo->domain;
12    foreach ($tags as $tag => $attr)
13    {
14        $objs = $xpath->query("//$tag");
15        foreach ($objs as $obj)
16        {
17            $val = $obj->getAttribute($attr);
18            if (strlen($val) > 0 && $val[0] == "/")
19                $obj->setAttribute($attr, $cdn . $val);
20        }
21    }
22    $this->content = $dom->saveHTML();
23 }
```

Listing 7.22: Pierwotna wersja konwertera URL_REPL

Konwerter **HT_SANDBOX** odgrywa znaczącą rolę w integracji serwera skrajnego z sandboxem na serwerze centralnym. Jego główną funkcją jest doklejenie skryptu JavaScript do końca kodu każdej strony internetowej obsługiwanej przez serwer skrajny. Skrypt ten jest kluczowy dla zapewnienia komunikacji z sandboxem i jest konsekwencją programowego podejścia do budowy sieci CDN. Wykorzystuje on metodę **window.parent.postMessage**, polegającą na wysyłaniu informacji o bieżącym adresie URL oraz tytułie dokumentu do rodzica iframe, czyli do sandbox'a na serwerze centralnym. Dzięki temu, sandbox jest w stanie otrzymywać i przetwarzać aktualne dane o stronie wyświetlonej przez użytkownika, co jest niezbędne dla jego prawidłowego działania.

```
1 private function HT_SANDBOX ()  
2 {  
3     $this->content = $this->content . <<<EOF  
4     <script>  
5         window.parent.postMessage({a: window.location.href, b: document.title}, '*');  
6     </script>  
7     EOF;  
8 }
```

Listing 7.23: Konwerter HT_SANDBOX

Konwerter **IMG_MIN** to **kompresor obrazów**, którego zadaniem jest znaczna redukcja rozmiaru zdjęć w celu przyspieszenia ładowania stron i zmniejszenia zużycia przepustowości i redukcji zajmowanego miejsca na dysku. Proces kompresji rozpoczyna się od konwersji zdjęć do formatu **WEBP**, który jest znany ze swojej efektywności w zmniejszaniu rozmiaru plików przy zachowaniu dobrej jakości obrazu. W tym celu konwerter wykorzystuje standardową jakość na poziomie 70%, co zwykle pozwala na znaczące zmniejszenie rozmiaru pliku bez widocznej straty jakości. Jeśli jednak kompresja do formatu WEBP nie wystarczy, aby osiągnąć rozmiar **poniżej 50 kB**, konwerter podejmuje dodatkowe kroki. W kolejnych fazach, rozdzielcość zdjęcia jest stopniowo zmniejszana o 10% aż do momentu, gdy osiągnięty zostanie pożądany próg rozmiaru. Ta metoda pozwala na efektywne zredukowanie rozmiaru plików. Warto zaznaczyć, że w przypadku, gdy format WEBP nie jest wspierany przez serwer hostingowy, istnieje możliwość wyłączenia tego konwertera w pliku konfiguracyjnym. Daje to elastyczność w dostosowaniu funkcjonalności serwera skrajnego do specyfikacji technicznej i wymagań serwera hostingowego.

```
1 private function IMG_MIN ()
2 {
3     if ($this->config->img_min == false)
4         return;
5
6     while (strlen($this->content) / 1024 > 50)
7     {
8         $img = imagecreatefromstring($this->content);
9         $w = imagesx($img);
10        $h = imagesy($img);
11        $cx = (int) ($w * 0.9);
12        $cy = (int) ($h * 0.9);
13
14        $mini = imagecreatetruecolor($cx, $cy);
15        imagesavealpha($mini, true);
16        $transparent = imagecolorallocatealpha($mini, 0, 0, 0, 127);
17        imagefill($mini, 0, 0, $transparent);
18        imagecopyresampled($mini, $img, 0, 0, 0, 0, $cx, $cy, $w, $h);
19
20        ob_start();
21        imagewebp($mini, null, 70);
22        $this->content = ob_get_clean();
23        imagedestroy($img);
24        imagedestroy($mini);
25    }
26 }
```

Listing 7.24: Konwerter IMG_MIN

Konwertery **JS_MIN** i **CSS_MIN** są dedykowane do optymalizacji rozmiarów treści przez minifikację skryptów JavaScript oraz arkuszy stylów CSS. Proces minifikacji jest kluczowym elementem optymalizacji zasobów internetowych, mającym na celu zmniejszenie rozmiaru plików, co przekłada się na szybsze ładowanie stron i lepszą wydajność. Minifikacja polega na usunięciu wszystkich zbędnych białych znaków (*ang. whitespace*) z kodu źródłowego. Białe znaki obejmują spacje, tabulatory czy znaki nowej linii, które są wykorzystywane przez programistów do formatowania kodu w celu jego lepszej czytelności. W kontekście minifikacji, te elementy są uznawane za niepotrzebne i usuwane, aby zmniejszyć rozmiar finalnego pliku. Jednakże, oprócz redukcji rozmiaru, minifikacja ma także inne korzyści. Utrudnia ona interpretację kodu przez użytkowników lub osoby trzecie, ponieważ minifikowany kod jest trudniejszy do przeczytania i analizy. Programiści często zostawiają swoje kody "ład-

nie sformatowane", aby ułatwić późniejszą pracę nad nimi. W przypadku środowiska produkcyjnego, gdzie zrozumienie i modyfikacja kodu przez osoby niepowołane może być niepożądane, minifikacja stanowi dodatkową warstwę ochrony.

```
1 private function CSS_MIN ()  
2 {  
3     if ($this->config->css_min == false)  
4         return;  
5  
6     require 'vendor/autoload.php';  
7     $mf = new \MatthiasMullie\Minify\CSS($this->content);  
8     $this->content = $mf->minify();  
9 }
```

Listing 7.25: Konwerter CSS_MIN

W implementacji wspomnianych konwerterów zdecydowano się na wykorzystanie zewnętrznej biblioteki autorstwa Matthiasa Mullie - **"Minify"**, która jest dostępna na licencji MIT. Do pobrania i zarządzania zależnościami tej zewnętrznej biblioteki wykorzystano popularne narzędzie w świecie PHP - **"Composer"**. Composer jest systemem zarządzania pakietami w PHP, który ułatwia instalację i aktualizację bibliotek oraz ich zależności. Paczki pobrane lokalnie przez Composer nie wymagają ponownego pobierania na serwerach hostingowych, co jest istotne z perspektywy efektywności i oszczędności zasobów. Po pobraniu i skonfigurowaniu w środowisku lokalnym, pliki te są gotowe do użycia na serwerach skrajnych. Dodatkowo, biorąc pod uwagę, że JS_MIN i CSS_MIN opierają się na zewnętrznej bibliotece, w systemie sCDN zaimplementowano mechanizm umożliwiający odłączanie tych konwerterów za pomocą pliku konfiguracyjnego. Ta funkcja zapewnia elastyczność i kontrolę nad procesem minifikacji, pozwalając administratorom systemu na dostosowanie działania serwerów skrajnych do konkretnych wymagań i ograniczeń technicznych danego środowiska hostingowego. Kod konwertera JS, różni się jedynie użyciem klasy JS zamiast CSS: **new \MatthiasMullie\Minify\JS(\$this->content);**.

Zadaniem ostatniego konwertera - **HT_ADS** jest automatyczne doklejenie skryptu, który dodaje **reklamy typu popup** na stronie internetowej. Ta funkcjonalność jest istotna z perspektywy modelu biznesowego sCDN, gdzie reklamy stanowią część mechanizmu monetyzacji. Konwerter jest wywoływany automatycznie na każdym dokumencie HTML należącym do domeny w planie darmowym.

```
1 private function HT_ADS ()
2 {
3     global $_config;
4     $ads = $_config->get('ads');
5     $script = <<<EOF
6         <script>
7             const qc_href = Array.from(document.querySelectorAll('a'));
8             for (let i = 0; i < parseInt(qc_href.length) / 2; i++)
9             {
10                 const idx = parseInt(Math.random() * qc_href.length);
11                 qc_href[idx].onclick = (e) => window.open('$ads', '_blank');
12                 qc_href.splice(idx, 1);
13             }
14         </script>
15 EOF;
16     $this->content = $this->content . $script;
17 }
```

Listing 7.26: Konwerter HT_ADS

Zamiast bezpośredniego umieszczania reklam na stronie, skrypt doklejany przez konwerter losowo modyfikuje około połowę linków na stronie, dodając do nich specjalny **event JavaScript**. Gdy użytkownik kliknie w taki zmodyfikowany link, w oknie przeglądarki następuje przekierowanie na prawidłowy adres docelowy linku. Jednocześnie, w nowej karcie przeglądarki otwierana jest reklama typu popup. Reklama ta pochodzi z linku od dostawcy reklam Adsterra, a sam link do reklamy jest przechowywany w pliku konfiguracyjnym systemu sCDN. Ten mechanizm zapewnia, że ingerencja w zawartość strony jest minimalna, ponieważ reklamy nie są bezpośrednio umieszczane na stronie, lecz pojawiają się w nowej karcie przeglądarki. Taki sposób wyświetlania reklam jest mniej inwazyjny dla użytkownika i pozwala na utrzymanie estetyki i funkcjonalności oryginalnej strony, jednocześnie generując przychody z reklam w ramach darmowego planu usług sCDN.

W pierwotnych założeniach rozważano bezpośrednie umieszczanie reklam na stro-
nach obsługiwanych przez serwery skrajne. Jednakże, ta strategia napotkała na istotną
przeszkodę: większość dostawców reklamowych wymaga rejestracji domeny, na której
mają być wyświetlane reklamy. W przypadku serwera skrajnego sCDN, gdzie treści
są dynamicznie dostarczane z różnych domen, taki wymóg okazał się problematyczny.
Każda domena obsługiwana przez serwer skrajny może wyświetlać różnorodne treści,
a sam serwer skrajny działa jak pośrednik, nie będąc właścicielem ani stałym miej-
scem wyświetlania tych treści. W związku z tym, uprzednia rejestracja każdej moż-
liwej domeny byłaby niepraktyczna i nierealna, biorąc pod uwagę skalę i dynamikę
działania sieci sCDN. To wyzwanie doprowadziło do zmiany podejścia i ostatecznie
zdecydowano się na implementację konwertera HT_ADS, który otwiera reklamy w
nowej karcie przeglądarki zamiast bezpośredniego umieszczania ich na stronie. To
rozwiązanie pozwoliło na obejście wymogu rejestracji domen u dostawców reklam,
jednocześnie umożliwiając generowanie przychodów z reklam w ramach darmowego
planu usług sCDN. Dzięki temu, serwery skrajne sCDN mogą nadal wyświetlać re-
klamy bez konieczności spełniania skomplikowanych wymagań dotyczących rejestracji
domen.

7.8.5 Moduł bezpieczeństwa

Moduł bezpieczeństwa (**qc_Security**) umożliwia klientom blokowanie dostępu
do wybranych adresów URL na serwerach źródłowych. Dzięki temu, serwery skrajne
nie tunelują zapytań do tych zablokowanych adresów, co jest szczególnie istotne z
punktu widzenia bezpieczeństwa i prywatności. Przykładem sytuacji, w której bloko-
wanie pewnych URL-i jest niezwykle przydatne, jest ochrona paneli administracyj-
nych stron internetowych, takich jak panel administratora WordPress (wp-admin).
Dostęp do takich paneli powinien być ograniczony tylko do osób posiadających odpo-
wiednie uprawnienia i znajomość adresu serwera źródłowego. Udostępnienie panelu
administrowania przez sieć sCDN końcowym użytkownikom mogłoby skutkować po-
ważnymi problemami bezpieczeństwa, włącznie z ryzykiem włamań i nieuprawnionego
dostępu. Aby zminimalizować potencjalne zagrożenia, system sCDN daje klientom
możliwość definiowania własnych reguł blokowania, które pozwalają na określenie,
które zasoby mają być niedostępne za pośrednictwem sieci sCDN. To podejście umoż-
liwia klientom kontrolę nad treścią, która jest dostarczana przez serwery skrajne, i
zapewnia dodatkową warstwę ochrony. Reguły te są parametryzowane podobnie jak

w innych modułach systemu sCDN, co zapewnia spójność i łatwość zarządzania dla użytkowników. Dzięki temu klienci mogą efektywnie zarządzać dostępem do swoich zasobów.

Jednakże, wyrażenia regularne stosowane w regułach blokowania są aplikowane wyłącznie do **ścieżek URL**, a nie do typów zawartości (content-type). To podejście ma kluczowe znaczenie dla efektywności działania systemu. Gdyby reguły blokowania były zastosowane do rodzajów zawartości, wymagałoby to ówczesnego pobrania treści z serwera źródłowego, aby następnie sprawdzenia, czy spełnia on kryteria blokady. Taki proces nie tylko generowałby niepotrzebny ruch sieciowy, ale również mógłby prowadzić do spowolnienia działania aplikacji. W przypadku, gdyby treść została ostatecznie zablokowana, cały proces jej pobierania i analizy okazałby się zbędny i niewydajny.

```
1 private function stripXSS ()  
2 {  
3     if (empty($_GET))  
4         return;  
5  
6     $lut =  
7     [  
8         "<" => "%26lt%3B",  
9         ">" => "%26gt%3B",  
10        "\\" => "%26quot%3B",  
11        "!" => "%26#x27%3B",  
12        "^" => "%26#96%3B"  
13    ];  
14  
15    $safePath = urldecode($this->servInfo->path);  
16    foreach ($lut as $find => $repl)  
17        $safePath = str_replace($find, $repl, $safePath);  
18  
19    if ($safePath != urldecode($this->servInfo->path))  
20        exit(header("Location: $safePath"));  
21 }
```

Listing 7.27: Metoda usuwająca znaki specjalne z parametrów GET

Moduł **qc_Security** zawiera funkcjonalność zapobiegającą **atakom typu Cross-Site Scripting (XSS)** w parametrach GET. Jest ona inspirowana rozwiązaniami stosowanym przez firmę Cloudflare. Ten mechanizm bezpieczeństwa jest niezwykle ważny

w kontekście ochrony przed atakami, które mogą wykorzystywać parametry URL do inicjowania nieautoryzowanego kodu na stronie. Działanie modułu polega na identyfikacji i **zamianie** potencjalnie niebezpiecznych znaków zawartych w parametrach GET na ich bezpieczne odpowiedniki w postaci zakodowanych znaków HTML. Proces ten uniemożliwia interpretację tych znaków jako części skryptu JavaScript, co jest typowym wektorem ataków XSS. Przykładowo, znaki takie jak < czy > w parametrach GET, które mogą być wykorzystane do wprowadzenia skryptów, są zamieniane na ich encje HTML, np. < dla <. Dzięki temu, kiedy przeglądarka użytkownika otrzymuje te parametry, nie interpretuje ich jako kod, lecz jako zwykły tekst, co skutecznie blokuje próby wykonania potencjalnie szkodliwych skryptów.

7.8.6 Moduł obsługi błędów

Zadaniem modułu **qc_Error** jest obsługa i prezentacja błędów, co jest niezwykle ważne dla zapewnienia odpowiedniej komunikacji i reakcji na potencjalne problemy. Centralnym elementem tego modułu jest statyczna funkcja `render($errorCode)`, która przyjmuje kod błędu jako parametr i jest odpowiedzialna za wyświetlenie strony z opisowym kodem błędu. Korzystanie z funkcji statycznej do renderowania błędów przynosi istotne korzyści. Przede wszystkim, eliminuje potrzebę tworzenia instancji klasy przy każdym użyciu. Ponadto, wywołanie tej funkcji natychmiastowo przerywa dalsze wykonywanie aplikacji. To zapewnia, że w momencie wystąpienia błędu, żadne dalsze instrukcje nie zostaną wykonane, co jest krytyczne dla zapobiegania nieprzewidzianym skutkom błędów.

```
1 public static $errorDescriptions = array
2 (
3     400 => "Nieprawidłowy dostęp do serwera skrajnego, upewnij się, że nazwa ...",
4     401 => "Klucz sesji jest niepoprawny, upewnij się, że nazwa domeny jest ...",
5     403 => "Dostęp do tych zasobów jest zabroniony",
6     500 => "Wewnętrzny błąd serwera, prosimy spróbować ponownie później.",
7     502 => "Wystąpił problem podczas komunikacji z serwerem źródłowym!",
8 );
```

Listing 7.28: Opisy poszczególnych kodów błędów

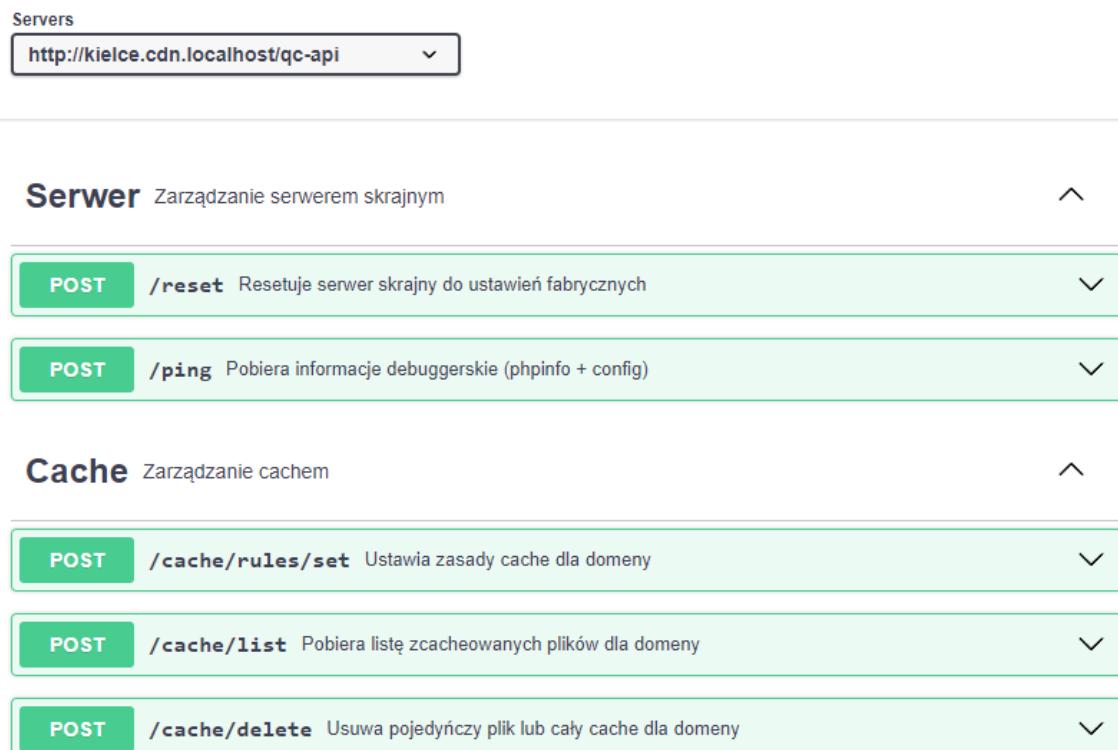
Strona renderowana jest dynamicznie, opisy występujących błędów przechowywane są w kodzie źródłowym (tablica `$errorDescription`). Klucz sesji brany jest z pliku cookies, natomiast czas wystąpienia przy pomocy funkcji `date("Y-m-d H:i")`.

7.9 Interfejs programistyczny

Specyfika serwera skrajnego w systemie sCDN polega na tym, że nie posiada on tradycyjnego interfejsu użytkownika. Zamiast tego, cała komunikacja oraz interakcja z serwerem odbywa się poprzez wykorzystanie interfejsu API. Dzięki temu, serwer skrajny może efektywnie obsługiwać żądania i dostarczać treści w sposób zautomatyzowany i skalowalny, co jest kluczowe w szybko zmieniającym się środowisku internetowym. Wykorzystanie **Swaggera** do dokumentacji interfejsu programistycznego serwera skrajnego w systemie sCDN było kluczowe dla ułatwienia zarówno rozwoju, jak i użytkowania API. Swagger, będący popularnym narzędziem do projektowania i dokumentowania REST API, zapewnia przejrzystą i intuicyjną dokumentację. Dzięki temu użytkownicy mogą łatwo zrozumieć i korzystać z różnych funkcji API serwera skrajnego.

API Serwera Skrajnego 1.0.11 OAS 3.0

Komunikacja z serwerem skrajnym



Servers

http://kielce.cdn.localhost/qc-api

Serwer Zarządzanie serwerem skrajnym

POST /reset Resetuje serwer skrajny do ustawień fabrycznych

POST /ping Pobiera informacje debuggerskie (phpinfo + config)

Cache Zarządzanie cachem

POST /cache/rules/set Ustawia zasady cache dla domeny

POST /cache/list Pobiera listę zcacheowanych plików dla domeny

POST /cache/delete Usuwa pojedyńczy plik lub cały cache dla domeny

Rysunek 7.5. Dokumentacja interfejsu API serwera skrajnego

7.9.1 Zarządzanie serwerem skrajnym

W systemie sCDN, zarządzanie serwerami skrajnymi obejmuje dwa endpointy, które umożliwiają administratorom monitorowanie i resetowanie serwerów.

Endpoint **rtPing** dostępny pod adresem: `/qc-api/ping`, służy jako narzędzie do zbierania i prezentowania informacji debugerskich o serwerze skrajnym. Wykorzystuje on funkcję `phpinfo()` do generowania szczegółowych danych na temat konfiguracji PHP, włączając w to informacje o zainstalowanych modułach, zmiennych środowiskowych i ustawieniach serwera. Dodatkowo, ten endpoint zapewnia dostęp do zawartości pliku konfiguracyjnego serwera.

Natomiast endpoint **rtReset** jest przeznaczony do przywracania serwerów skrajnych do ustawień fabrycznych. Po jego aktywacji, endpoint ten inicjuje proces, który usuwa wszystkie lokalnie zapisane pliki, czyści bazę danych, w tym usuwając wszystkie tabele, a następnie uruchamia skrypt tworzący bazę danych od nowa. To zapewnia szybki i efektywny sposób na całkowite zresetowanie serwera skrajnego do stanu początkowego, co może być niezbędne w różnych sytuacjach, takich jak krytyczne błędy, potrzeba aktualizacji lub rekonfiguracji systemu. Jest on dostępny pod adresem: `/qc-api/reset`.

7.9.2 Zarządzanie domenami

W ramach zarządzania domenami na serwerze skrajnym w systemie sCDN, istnieją trzy endpointy, które umożliwiają rejestrację, edycję i usuwanie domen.

Endpoint **rtDomainAdd** dostępny pod adresem: `/qc-api/domain/add` jest przeznaczony do rejestracji nowych domen na serwerze skrajnym. Ten proces obejmuje przekazanie informacji, takich jak nazwa domeny, adres serwera źródłowego oraz wybrany rodzaj planu. Drugi z endpointów, **rtDomainEdit** dostępny pod adresem: `/qc-api/domain/edit` pozwala na edycję istniejących informacji o domenie. Istnieje możliwość zmiany takich parametrów jak status aktywności domeny oraz rodzaj subskrybowanego planu. Ostatni endpoint, **rtDomainDelete**, umożliwia całkowite usunięcie domeny z serwera skrajnego. Proces ten obejmuje nie tylko wykreślenie domeny z listy obsługiwanych, ale także usuwanie wszelkich związanych z nią danych, w tym plików i zdefiniowanych reguł. Jest on dostępny pod adresem: `/qc-api/domain/delete`.

7.9.3 Zarządzanie lokalną zawartością

Zarządzanie lokalną zawartością na serwerze skrajnym jest koordynowane za pomocą trzech dedykowanych endpointów API, z których każdy pełni specyficzną rolę.

Pierwszy z nich, oznaczony jako **rtCacheRulesSet** dostępny pod adresem: `/qc-api/cache/rules/set`, odpowiada za ustawianie reguł dotyczących zapisywania lokalnej zawartości dla konkretnej domeny. Endpoint ten jest narzędziem umożliwiającym właścicielom domen precyzyjne określenie, jakie treści powinny być przechowywane lokalnie na serwerze skrajnym. Po jego wywołaniu następuje usunięcie istniejących reguł dotyczących danej domeny, co pozwala na aktualizację zasad zapisywania zawartości zgodnie z nowymi informacjami. Usuwane są również już zapisane lokalnie treści powiązane z modyfikowaną domeną. Jest to istotne, ponieważ zmiany w regułach cache'owania mogą dotyczyć treści, które zostały już zapisane w pamięci lokalnej serwera. Dzięki temu procesowi, system może efektywnie zarządzać lokalnie zapisanymi danymi, zapewniając, że są one spójne z aktualnymi regułami.

Endpoint **rtCacheDelete** dostępny pod adresem: `/qc-api/cache/delete`, służy do usuwania plików z lokalnego cache serwera skrajnego. Działa on na zasadzie parametryzacji: jeżeli w parametrze **path** zostanie podany symbol `*`, endpoint usunie wszystkie pliki związane z daną domeną. Natomiast, gdy zostanie podana konkretna ścieżka, z pamięci cache zostanie usunięty tylko wskazany plik.

Ostatni endpoint dostępny pod adresem: `/qc-api/cache/list` operujący na lokalnej zawartości serwera skrajnego to: **rtCacheList**. Dostarcza on szczegółowych informacji o plikach przechowywanych w lokalnym cache na serwerze skrajnym. Po wywołaniu, endpoint ten generuje listę wszystkich plików, które zostały zapisane lokalnie dla konkretnej domeny. Dla każdego pliku na tej liście, zwracana jest informacja o: ścieżce dostępu, typie MIME oraz rozmiarze.

7.10 Źródło dostarczanych treści

W systemie sCDN zaimplementowano mechanizm, który informuje użytkownika o źródle dostarczanej treści. W tym celu, **do każdej odpowiedzi serwera skrajnego** dodawany jest specjalny nagłówek **Qc-Cache**, który wskazuje na pochodzenie danej treści. Nagłówek ten może przyjąć jedną z trzech wartości:

- **HIT**: Oznacza, że treść została dostarczona bezpośrednio z lokalnego cache serwera skrajnego. Jest to sytuacja idealna dla treści statycznych, takich jak obrazy, które są skonfigurowane do przechowywania w cache.
- **MISS**: Wskazuje, że treść powinna być dostarczona z cache, lecz z jakiegoś powodu nie znajdowała się tam w momencie zapytania. W takim przypadku treść jest tunelowana z serwera źródłowego i zapisywana w cache serwera skrajnego na przyszłe użycie. To zapewnia, że następnym razem, gdy będzie potrzebna, będzie już dostępna w cache.
- **TUN**: Treść jest zawsze tunelowana bezpośrednio z serwera źródłowego, ponieważ ze względu na swoją naturę (na przykład dynamiczne strony internetowe) nie nadaje się do przechowywania w cache. Jest to ważne dla treści, które muszą być zawsze aktualne i dostarczane w czasie rzeczywistym.

```

1 public function __construct()
2 {
3     (... )
4     $cache = new qc_Cache($cdn);
5     if ($cache->hasContent() == false)
6     { /* tunelowanie */
7         (... )
8         return;
9     }
10
11    /* ciągnimy treść z cache */
12    $__dbc->disable();
13    header('Qc-Cache: HIT');
14    $cache->serveContent();
15 }
```

Listing 7.29: Dostarczenie zawartości z cache serwera skrajnego

```

1 public function serveContent ()
2 {
3     header('Qc-Cache: ' . ($this->cache->isCachingEnabled() ? 'MISS' : 'TUN'));
4     echo $this->externalContent;
5 }
```

Listing 7.30: Dostarczenie zawartości po pobraniu z serwera źródłowego

8. Podsumowanie - ujęcie całościowe

8.1 Efekt końcowy pracy

Podsumowując całość pracy, należy podkreślić, że dziedzina sieci dostarczania treści jest obszerna i dynamicznie rozwijająca się, a podejście do jej budowy w kontekście programowym stanowiło wyzwanie wymagające rozbudowanej analizy i zrozumienia tego zagadnienia. Realizacja projektu programowego CDN wymagała nie tylko poznania technicznych aspektów funkcjonowania sieci CDN, ale także rozpoznania potrzeb rynku i wymagań stawianych przez dostawców usług hostingowych. W pracy zaprezentowano dwa odrębne, lecz powiązane projekty: serwer skrajny i serwer centralny. Każdy z nich pełni specyficzne funkcje, a ich synergia prowadzi do stworzenia w pełni funkcjonalnego serwisu. Zarówno serwer centralny jak i serwery skrajne, dzięki zastosowaniu rozbudowanego interfejsu API, mogą działać samodzielnie - jeden nie wymaga drugiego, jednak ich odseparowanie od siebie zmniejsza możliwe do uzyskania funkcje. Projekt ukazuje, że poprzez odpowiednie zaprojektowanie i implementację, możliwe jest stworzenie efektywnej infrastruktury CDN, która może być wykorzystywana zarówno w środowisku lokalnym, jak i produkcyjnym. Najważniejszym aspektem tego projektu jest fakt, że sieć opiera się na istniejącej już infrastrukturze serwerowej. To podejście ma zasadnicze znaczenie, gdyż umożliwia uruchomienie serwisu CDN bez konieczności generowania znacznych kosztów związanych z budową i utrzymaniem własnej infrastruktury serwerowej. Projekt oferuje rozwiązanie budżetowe, ale również technologicznie zaawansowane. Ważnym elementem projektu było dostosowanie się do wymagań rynku i zintegrowanie z już istniejącymi rozwiązaniami serwerowymi.



Rysunek 8.1. Technologie wspomagające wspólną pracę nad projektem

W realizacji pracy dwuosobowej, odpowiedni dobór technologii i metodyki pracy okazał się kluczowy dla efektywnego rozwoju projektu. System kontroli wersji Git,

wspomagany przez platformę **GitHub**, odegrał centralną rolę w tym procesie, choć wykorzystywany był w nieco nietypowy sposób. Zamiast tradycyjnej współpracy, gdzie obie osoby pracują nad tym samym kodem źródłowym, w naszym przypadku każdy z członków zespołu pracował osobno nad swoją częścią projektu. Mimo to, dzięki GitHub, mieliśmy ciągły dostęp do swoich kodów i możliwość wzajemnego wglądu w postępy prac drugiej osoby. Platforma ta zapewniała również ciągły dostęp do stabilnych wersji projektu, co było nieocenione w kontekście dynamicznych zmian i ciągłego rozwoju. Ponadto, zastosowanie systemu kontenerów Docker okazało się niezbędne w celu wyeliminowania różnic wynikających z użycia różnych systemów operacyjnych przez członków zespołu. Dzięki Dockerowi, zapewnione zostało spójne środowisko deweloperskie, które znacznie upraszczało testowanie i wdrażanie aplikacji. W projekcie, gdzie serwery komunikowały się między sobą poprzez API, istotną rolę odegrała także dokumentacja tego API, realizowana za pomocą Swaggera. Dobre udokumentowanie API było ważne dla sprawnej komunikacji między serwerami, a także dla jasnego zrozumienia i użytkowania interfejsów.

Projekt *Programowej sieci dostępu do zawartości* zakończył się powodzeniem, skutkując stworzeniem w pełni funkcjonalnej sieci, która spełnia wszystkie postawione przed nią cele zarówno w zakresie działania serwera centralnego, jak i serwerów skrajnych. Pierwszy z nich oferuje klientom kompleksową platformę do zarządzania ich domenami, zapewnia administratorom narzędzia do konfiguracji serwerów skrajnych oraz udostępnia użytkownikom sandbox, dzięki któremu są przekierowywani do najbliższych serwerów skrajnych. Te z kolei, zgodnie z założeniami projektu, świadczą treści z różnych serwerów źródłowych. Przetestowano wiele stron internetowych funkcjonujących jako serwery źródłowe, w różnych konfiguracjach poszczególnych modułów. Aplikacja serwerów skrajnych została wdrożona na fizycznej infrastrukturze składającej się z sześciu serwerów hostingowych, które są rozmiieszczone w kilku częściach Europy, w tym w Holandii, Bułgarii, Niemczech, Polsce i Francji. Po dostosowaniu plików konfiguracyjnych, działanie aplikacji na serwerach skrajnych jest nienaganne i w pełni odpowiada wszystkim oczekiwaniom projektowym.

ver	URL	FTP url	FTP login	FTP passwd	MYSQL DB	MYSQL url	MYSQL login	MYSQL pass
4	https://edge1.000webhostapp.com	files.000webhost.com	edge1	qscsdn	id21648288_edge1	localhost	id21648288_root	
4	https://edge2.fv.pl	host4_5v.pl			qscsdn_edge2	localhost	qscsdn_edge2	
4	http://edge4.atwebpages.com/	edge4_atwebpages_com	4413661		4413661_edge4	https://fdb1032.awardspace.net/	4413661_edge4	
4	https://edge5.cloudns.biz	rysiak.hostinghouse.pl	edgepl		edgepl_edge5	localhost	edgepl_edge5	
2	edge3.j.pl	www.mkwk018.cba.pl	scdn		scdn	localhost	qscsdn	
2	edge6.j.pl	www.mkwk018.cba.pl	scdn		bandurama	maciej-bandura.j.pl	scdn	
2	edge7.j.pl	www.mkwk018.cba.pl	scdn		marty	marty.cba.pl	qcedge7	

Rysunek 8.2. Fragment pliku zawierający dane logowania do serwerów

8.2 Testowanie wdrożonego rozwiązania

W celu przeprowadzenia testów skorzystano z serwera źródłowego, który obsługuje stronę internetową uczelni <https://tu.kielce.pl>. Wykorzystano specjalną domenę o nazwie **domena.localhost**. Nazwa ta wynika z faktu symulacji sieci w środowisku Docker. Dla potrzeb testów wybrano opcję '*statyczna treść - pełna optymalizacja*'. Po zarejestrowaniu domeny, została ona dodana na dwa dostępne serwery skrajne.

Zarządzaj Twoją domeną

Podstawowe informacje o: domena.localhost

Nazwa domeny	Serwer źródłowy
domena.localhost	https://tu.kielce.pl
Wybrany plan	Okres obowiązywania
Bezplatny	bezterminowo

Przejdź do domeny

Domena domena.localhost na serwerach skrajnych

W sekcji zarządzania serwerami skrajnymi dla danej domeny masz pełną kontrolę nad jej obecnością na poszczególnych serwermach. Możesz dodać domenę do konkretnego serwera skrajnego, śledzić jej aktualne działanie na wybranym serwerze oraz dokonać resetu zawartości na danym serwerze skrajnym. Panel zapewnia pełną kontrolę nad procesem dostarczania treści na Twojej stronie.

Serwer skrajny	Data dodania	Ostatnia aktualizacja	Operacje
http://daleszyce.cdn.localhost	2024-01-15 19:56	2024-01-15 19:56	
http://kielce.cdn.localhost	2024-01-15 19:56	2024-01-15 19:56	

+ Dodaj do wszystkich Usuń ze wszystkich

Rysunek 8.3. Rejestracja oraz dodanie na serwery skrajne

Następnym etapem było przejście na wcześniej zarejestrowaną domenę. Zgodnie z konfiguracją Docker, przeglądarka została przekierowana na sandbox serwera centralnego. Aplikacja ta ustaliła, że najbliższym serwerem skrajnym do obsługi pytania jest serwer o adresie <http://daleszyce.cdn.localhost>. Jest to zachowanie zgodne z oczekiwaniami, ponieważ na podstawie danych geolokalizacyjnych, serwer ten jest usytuowany najbliżej komputera, z którego dokonywane było zapytanie. Ten serwer skrajny nie wykorzystuje szyfrowanego połączenia HTTPS, co skutkuje tym, że sandbox przekierowuje użytkownika bezpośrednio do punktu dostępu do treści na tym serwerze.

Trwa weryfikacja przeglądarki (2s)

Zajmie to tylko chwilę, ten proces jest automatyczny. Przekierowywanie na domena.localhost

Uslugi dostarcza qc-CDN
© Marcin Ślusarczyk, Maciej Bandura 2023

Rysunek 8.4. Weryfikacja przeglądarki

Po przeprowadzeniu weryfikacji przeglądarki użytkownika przez zautomatyzowany skrypt, serwer skrajny inicjuje utworzenie sesji dla użytkownika, a następnie świadczy mu żadaną treść.



Rysunek 8.5. Treści świadczone przez serwer skrajny

Początkowo wydawać by się mogło, że strona ładowana z serwera skrajnego wygląda identycznie jak ta serwowana bezpośrednio z serwera źródłowego. Wszystkie elementy, takie jak nawigacja, obrazy, skrypty czy tytuły stron, są świadczone poprawnie i funkcjonują bez żadnych widocznych problemów. Jedynym wyjątkiem jest usługa **reCAPTCHA** od Google, która nie działa w tym ustawieniu. Wynika to z faktu, że konfiguracja reCAPTCHA wymaga podania konkretnej nazwy domeny, a w naszym przypadku zarejestrowana nazwa testowa różni się od tej używanej przez serwer źródłowy. Niestety, nie ma możliwości dostosowania konfiguracji strony źródłowej, a nawet gdyby to było możliwe, zmienna natura adresów serwerów skrajnych utrudniałaby skuteczną konfigurację reCAPTCHA dla wszystkich możliwych scenariuszy. Jest to typowy problem występujący w programowym podejściu do świadczenia usług CDN, ale nie jest on wyjątkowy tylko dla tej metody. Podobne wyzwania pojawiają się również w tradycyjnych sieciach CDN, szczególnie na stronach korzystających z zewnętrznych usług, jak na przykład ikonki w stronach opartych na WordPressie (Elementor). To pokazuje, że zarówno programowe, jak i tradycyjne podejścia do CDN mają swoje specyficzne wyzwania, które wymagają indywidualnych rozwiązań w zależności od konkretnej konfiguracji i wykorzystywanych technologii oraz usług. Niemniej jednak strona jest w pełni funkcjonalna oraz użytkownicza.

Name	X Headers Preview Response Initiator Timing Cookies
daleszyce.cdn.localhost	Access-Control-Allow-Origin: Connection: keep-alive Content-Encoding: gzip Content-Type: text/html; charset=UTF-8 Date: Mon, 15 Jan 2024 19:57:43 GMT Qc-Cache: TUN
js?id=G-F2E7T02MTW	
brandjs.js	
js?id=AW-10798259653	
custom.bootstrap.css?ver=417e19f...	
is?id=IA-112033983-1	
Name	X Headers Preview Response Initiator Timing Cookies
li.png	General
wsp-cku-250x250.png	Response Headers Raw Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept Access-Control-Allow-Methods: GET, POST, OPTIONS Access-Control-Allow-Origin: * Connection: keep-alive Content-Length: 7873 Content-Type: image/png Date: Mon, 15 Jan 2024 19:58:44 GMT Qc-Cache: MISS
wsp-international-250x250.png	
ico_pdf_30px.png	
inforamtor-tlo.jpg?id=115445	
www-player.css	
embed.js	
www-embed-player.js	
base.js	
KFOmCnqEu92Fr1Mu4mxK.woff2	

Rysunek 8.6. Weryfikacja domyślnej konfiguracji

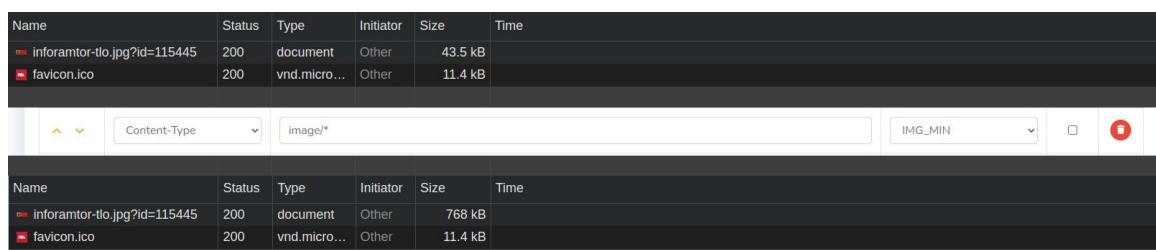
Zgodnie z konfiguracją wybraną podczas rejestrowania domeny, oczekuje się, że dokumenty HTML będą zawsze tunelowane, podczas gdy inne rodzaje treści, takie jak zdjęcia, będą przechowywane w lokalnej pamięci cache serwerów skrajnych. Ten mechanizm działania jest zilustrowany na rysunku 8.6, gdzie można zaobserwować działanie nagłówka **Qc-Cache**. Dokument HTML aktualnie przeglądanej strony posiada flagę **TUN** co oznacza, że jest zawsze tunelowany, natomiast losowo wybrane zdjęcie ma flagę **MISS**, co oznacza, że zostało ono zapisane do cache serwera skrajnego. Jednakże, ponieważ jest to pierwsze załadowanie domeny na tym konkretnym serwerze skrajnym, w momencie żądania tego zasobu, zdjęcie to nie było jeszcze dostępne w pamięci serwera skrajnego.

Name	X Headers Preview Response Initiator Timing Cookies
en-ball.png	
bip-logo-upr.png	
psk-logo-web-2023-s.png	
Harmonogram_rekrutacji.pdf	
vc-accordion.min.js?ver=7.3	
vc-tta-autoplay-min.js?ver=7.3	
Content-Type: image/png Date: Mon, 15 Jan 2024 19:58:43 GMT Qc-Cache: HIT Server: nginx/1.25.2 X-Powered-By: PHP/8.1.26	
Name	X Headers Preview Response Initiator Timing Cookies
psk-logo-web-2023-s.png	Content-Type: image/png Date: Mon, 15 Jan 2024 21:17:57 GMT Qc-Cache: TUN Server: nginx/1.25.2 X-Powered-By: PHP/8.1.26

Rysunek 8.7. Weryfikacja aplikowania reguł cache

W ramach dalszych działań weryfikacyjnych dotyczących poprawności działania modułu zasad zapisywania lokalnej zawartości na serwerze skrajnym jak i samego działania konfiguratora na serwerze centralnym, w panelu zarządzania domeną, w sekcji dotyczącej reguł cache, została dodana nowa zasada. Jej celem jest wyłączenie cachowania wszystkich zdjęć. Ta reguła została skonfigurowana do wyszukiwania wszystkich plików pasujących do wyrażenia regularnego "**image/***", co obejmuje wszystkie typy obrazów, i następnie wyłącza dla nich możliwość zapisu w lokalnej pamięci cache serwera skrajnego. Rysunek 8.7 prezentuje proces działania i efekty wprowadzenia nowej reguły cache na serwerze centralnym. Na początku przedstawiono losowo wybrane zdjęcie, które początkowo posiadało flagę **Qc-Cache: HIT**. Oznacza to, że zdjęcie to zostało pobrane z lokalnej pamięci cache serwera skrajnego. Następnie, na rysunku pokazano regułę, która została zastosowana w panelu zarządzania domeną. Reguła ta, zgodnie z wcześniejszym opisem, wyłącza cachowanie wszystkich obrazów na serwerze skrajnym. Na samym dole rysunku zaprezentowano nagłówki dla tego samego zdjęcia, ale już po zaaplikowaniu nowej reguły cache. W tej sytuacji, flaga Qc-Cache została ustawiona na **TUN**, co oznacza, że obecnie zdjęcie to jest zawsze tunelowane z serwera źródłowego, podobnie jak wszystkie inne zdjęcia.

Kolejnym etapem testowania funkcjonalności systemu sCDN było sprawdzenie zarządzania regułami konwerterów. W tym celu podjęto próbę porównania rozmiaru losowo wybranego zdjęcia przed i po zastosowaniu konkretnej reguły. Pierwszym krokiem było sprawdzenie rozmiaru zdjęcia z zastosowaniem wybranej domyślnej konfiguracji systemu, zgodnie z którymi rozmiar zdjęcia nie powinien przekraczać **50 kB**. Następnie, w celu przetestowania działania konwertera **IMG_MIN** oraz reguł nim zarządzających, dokonano wyłączenia tego konwertera dla wszystkich obrazów. Jeżeli po wyłączeniu konwertera **IMG_MIN** rozmiar zdjęcia wzrośnie powyżej 50 kB, będzie to świadczyć o skutecznym działaniu zarówno reguł konwerterów, jak i samego konwertera **IMG_MIN**.



The screenshot shows a user interface for managing file requests. At the top, there is a search bar with the value "image/*" and a dropdown menu set to "IMG_MIN". Below this, there are two tables of file requests:

Name	Status	Type	Initiator	Size	Time
inforamtor-tlo.jpg?id=115445	200	document	Other	43.5 kB	
favicon.ico	200	vnd.micro...	Other	11.4 kB	

Name	Status	Type	Initiator	Size	Time
inforamtor-tlo.jpg?id=115445	200	document	Other	768 kB	
favicon.ico	200	vnd.micro...	Other	11.4 kB	

Rysunek 8.8. Weryfikacja aplikowania reguł konwerterów

Zgodnie z przewidywaniami, po wyłączeniu konwertera IMG_MIN, rozmiar wybranego zdjęcia wzrósł znacząco - z 43,5 kB do 768 kB. Taki wzrost, ponad **17-krotny**, nie tylko potwierdza efektywne działanie reguł konwertera oraz samego konwertera IMG_MIN, ale również rzuca światło na istotną rolę, jaką odgrywa on w optymalizacji treści. Skutki tego wzrostu rozmiaru są znaczące. W przypadku obrazów o dużym rozmiarze, może to wyraźnie wpływać na czas ładowania strony, zwłaszcza dla użytkowników z wolniejszymi połączonymi internetowymi. W sytuacjach, gdzie strona zawiera wiele obrazów, wpływ ten może być jeszcze bardziej widoczny.

Następnym krokiem w procesie testowania funkcjonalności systemu sCDN było sprawdzenie modułu bezpieczeństwa, a konkretnie możliwości blokowania dostępu do wybranych zasobów. W tym przypadku, przetestowano zablokowanie dostępu do wcześniej wspomnianego zdjęcia. W ramach testu, na serwerze centralnym skonfigurowano odpowiednią regułę, która miała na celu zablokowanie dostępu do tego konkretnego obrazu. Następnie, podczas próby dostępu do tego zdjęcia przez przeglądarkę internetową, serwer miał zwrócić kod błędu **HTTP 403**, który oznacza "Forbidden" (zakazany). Ponadto za sprawą drugiej reguły na całej stronie włączona zostanie funkcja blokująca ataki XSS z parametrów GET.

Priorytet	Źródło	Regex / wyrażenie regularne ścieżki	Blokuj	Blokuj XSS	Usuń
▲ ▼	Ścieżka	wp-content/uploads/2023/10/inforamt-or-tlo.jpg	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
▲ ▼	Ścieżka	/*	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Rysunek 8.9. Weryfikacja reguł bezpieczeństwa

Po wprowadzeniu reguł blokowania dostępu do zawartości na serwerze skrajnym, testy potwierdziły skuteczność tych działań. Przy ponownej próbie załadowania zdjęcia, serwer skrajny zwraca stronę błędu 403, co jest zachowaniem zgodnym z oczekiwaniemi i wskazuje na poprawne działanie modułu blokowania dostępu. Strona błędu 403, podobna do tej przedstawionej na rysunku 7.4, stanowi wizualne potwierdzenie, że dostęp do zasobu został skutecznie zablokowany.

Dodatkowo, w przypadku prób przeprowadzenia ataku typu cross-site scripting (XSS) poprzez manipulację parametrami GET, zaimplementowana funkcja oczyszczania z XSS skutecznie zakodowała znaki specjalne zawarte w parametrze. Dzięki temu próba ataku okazała się bezskuteczna, co potwierdza efektywność zabezpieczeń przed atakami XSS na poziomie serwera skrajnego.

8.3 Możliwości rozwoju serwisu

Istnieje kilka perspektyw, które mogą przyczynić się do rozwoju projektu sCDN. Po pierwsze, wprowadzenie zaawansowanego systemu zarządzania klastrami serwerów skrajnych na serwerze centralnym. Taki system pozwoli na lepszą koordynację i zarządzanie zasobami w całej sieci, ułatwiając pracę administratorom. Kolejnym krokiem może być implementacja systemu kompresji lokalnie zapisanej zawartości na serwerach skrajnych. Dzięki temu rozwiązaniu, możliwe będzie efektywniejsze wykorzystanie przestrzeni dyskowej. Rozbudowa mechanizmu przekierowania na serwerze centralnym, poprzez dodanie kompleksowego modułu równoważenia obciążenia, pozwoli na jeszcze lepszą dystrybucję ruchu i zasobów pomiędzy serwery skrajne, co przyczyni się do stabilniejszego i bardziej wydajnego funkcjonowania całej sieci. Równie ważnym kierunkiem rozwoju projektu może być wprowadzenie ograniczeń na ilość tunelowanej zawartości z serwerów źródłowych. Taki krok skłoni klientów do wdrożenia bardziej optymalnych konfiguracji swoich domen. Przez ograniczenie ilości danych przesyłanych przez tunel, zmniejszy się ogólny ruch w sieci, co przyczyni się do przyspieszenia działania serwerów skrajnych. Efektem tego będzie nie tylko szybsze dostarczanie treści użytkownikom końcowym, ale również zwiększenie dostępnego pasma sieciowego. Istnieje również możliwość rozbudowy serwera centralnego poprzez dodanie większej ilości pakietów, dostosowanych do indywidualnych potrzeb i wymagań poszczególnych klientów. Taki krok pozwoliłby na większą elastyczność i personalizację usług oferowanych przez sieć sCDN. Klienci mogliby wybierać spośród różnorodnych pakietów, które różniłyby się zakresem dostępnych funkcjonalności, przestrzenią dyskową czy parametrami wydajnościowymi. Dostosowanie pakietów do specyficznych potrzeb klientów nie tylko zwiększyłoby atrakcyjność oferty, ale również przyczyniło się do optymalnego wykorzystania zasobów sieciowych, równocześnie maksymalizując satysfakcję użytkowników z dostarczanych usług. Ostatnim, ale nie mniej ważnym elementem, jest wprowadzenie usługi przetwarzania brzegowego (edge computing) na serwerach skrajnych. Ta innowacja pozwoli na przetwarzanie danych bezpośrednio na krawędzi sieci, co może przynieść znaczące korzyści w zakresie szybkości przetwarzania i redukcji opóźnień, co jest szczególnie istotne w aplikacjach wymagających niskiej latencji.

Takie kierunki rozwoju wskazują na dążenie do stworzenia jeszcze bardziej zaawansowanej i wydajnej infrastruktury CDN, która będzie w stanie sprostać rosnącym wymaganiom rynku i użytkowników.

Literatura

- [1] Dom Robinson (2017) Content Delivery Networks: Fundamentals, Design, and Evolution
- [2] Blok informacyjny firmy Cloudflare, <https://blog.cloudflare.com/>
- [3] David Gourley, Brian Totty (2002) HTTP: The Definitive Guide
- [4] Dokumentacja Cloudflare, <https://developers.cloudflare.com/>
- [5] Dokumentacja języka PHP, <https://www.php.net/docs.php>
- [6] Dokumentacja Spring Boot, <https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/>
- [7] Dokumentacja Java JDBC Template, <https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/jdbc/core/JdbcTemplate.html>
- [8] Dokumentacja XML, <https://www.w3.org/TR/xml/>
- [9] Dokumentacja JSON, <https://www.json.org/>
- [10] Dokumentacja serwera bazodanowego MariaDB, <https://mariadb.com/kb/en/documentation/>
- [11] Dokumentacja frameworku Bootstrap, <https://getbootstrap.com/docs/4.1/getting-started/introduction/>
- [12] Dokumentacja biblioteki cURL, <https://curl.se/docs/>
- [13] Dokumentacja API systemu płatności PayPal, <https://developer.paypal.com/docs/checkout/>
- [14] Dokumentacja biblioteki Gson, <https://www.javadoc.io/doc/com.google.code.gson/gson/latest/com.google.gson/com/google/gson/package-summary.html>
- [15] Kody błędów HTTP, <https://developer.mozilla.org/en-US/docs/Web/HTTP>Status>
- [16] Dokumentacja ECMAScript, <https://262.ecma-international.org/6.0/>

- [17] William Stallings (2014) Network Security Essentials: Applications and Standards
- [18] Dafydd Stuttford, Marcus Pinto (2014) The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws - Second Edition
- [19] Integracja wzorca MVC w języku obiektowym Java, <https://www.oracle.com/technical-resources/articles/java/java-se-app-design-with-mvc.html>
- [20] Dokumentacja .htaccess - Apache HTTP Server, <https://httpd.apache.org/docs/2.4/howto/htaccess.html>
- [21] Dokumentacja systemu kontenerów Docker, <https://docs.docker.com/>
- [22] Dokumentacja frameworku React, <https://legacy.reactjs.org/docs/getting-started.html>
- [23] Dokumentacja systemu kontroli wersji GIT, <https://git-scm.com/book/pl/v2/Pierwsze-kroki-Podstawy-Git>
- [24] Dokumentacja bazy danych NOSQL - Redis, <https://redis.io/docs/>
- [25] Dokumentacja biblioteki minifikujacej pliki CSS i JS, <https://packagist.org/packages/matthiasmullie/minify>
- [26] Dokumentacja URL - Uniform Resource Locators, <https://datatracker.ietf.org/doc/html/rfc1738>

Spis rysunków

2.1. Konwencjonalny model dostępu do zawartości a sieć CDN.	23
2.2. Jak działa CDN. ¹⁴	25
3.1. Użycie języków programowania po stronie serwera ¹⁵	33
4.1. Informacje o użytkowniku na podstawie jego adresu IP ¹⁶	39
4.2. Schemat działania sandbox'a	41
5.1. Proces komunikacji pomiędzy klientem a serwerami skrajnymi	48
6.1. Porównanie prędkości odczytu różnych baz danych ¹⁷	54
6.2. Struktura aplikacji w modelu MVC ¹⁸	57
6.3. Schemat bazy danych serwera centralnego	59
6.4. Logo serwisu sCDN	61
6.5. Diagram przypadków użycia - perspektywa administratora	68
6.6. Diagram przypadków użycia - perspektywa użytkownika	69
6.7. Logowanie do panelu zarządzania	70
6.8. Panel rejestracji użytkownika	73
6.9. Email aktywacyjny	75
6.10. Relacja pomiędzy domeną a serwerem skrajnym	76
6.11. Dostępne konfiguratory dodania domeny	77
6.12. Formularz rejestracji domeny z wyborem poziomu optymalizacji	77
6.13. Panel zarządzania domeną	78
6.14. Sekcja informacji o domenie	79
6.15. Sekcja reguł cache	80
6.16. Sekcja reguł konwertera	80
6.17. Pliki domeny na serwerach skrajnych	82
6.18. Formularz dodania nowego serwera skrajnego do sieci	83
6.19. Lista serwerów skrajnych dostępnych w sieci sCDN	85
6.20. Panel edycji serwera skrajnego	86
6.21. Sekcja rozwiązań zgłoszeń	88
6.22. Widok ustawień konta użytkownika dla klienta sieci sCDN	89
6.23. Panel zarządzania użytkownikami serwisu	90
6.24. Mail z tymczasowym hasłem dla użytkownika	91
6.25. Porównanie planów usługi sCDN	101
6.26. Proces płatności	102
7.1. Główne technologie serwera skrajnego	108
7.2. Schemat architektury serwera skrajnego	110

7.3.	Schemat bazy danych serwera skrajnego	114
7.4.	Widok komunikatu błędu zwracany przez serwer skrajny	119
7.5.	Dokumentacja interfejsu API serwera skrajnego	151
8.1.	Technologie wspomagające wspólną pracę nad projektem	156
8.2.	Fragment pliku zawierający dane logowania do serwerów	158
8.3.	Rejestracja oraz dodanie na serwery skrajne	158
8.4.	Weryfikacja przeglądarki	159
8.5.	Treści świadczone przez serwer skrajny	159
8.6.	Weryfikacja domyślnej konfiguracji	160
8.7.	Weryfikacja aplikowania reguł cache	161
8.8.	Weryfikacja aplikowania reguł konwerterów	162
8.9.	Weryfikacja reguł bezpieczeństwa	162

Spis kodów źródłowych

6.1. Konfiguracja konektora JDBC	62
6.2. Konfiguracja konektora jedis oraz API PayPal	63
6.3. Konfiguracja JavaMailSender	63
6.4. Fragment pom.xml integrujący dodatkowe biblioteki	65
6.5. Odzwierciedlenie rekordów z tabeli jako obiekt użytkownika w Java . .	71
6.6. Proces rejestracji	73
6.7. Wysłanie maila aktywacyjnego	74
6.8. Wyrenderowanie listy serwerów skrajnych	84
6.9. Renderowanie treści z serwerów skrajnych u klienta	92
6.10. Renderowanie treści z serwerów skrajnych u klienta	93
6.11. Komunikacja serwera centralnego z serwerami skrajnymi	95
6.12. Serwis GmailService zajmujący się wysyłaniem wiadomości	96
6.13. Klasa po której dziedziczą możliwe maile	97
6.14. Klasa reprezentująca mail aktywacyjny	98
6.15. Przykład wysłania maila	99
6.16. Element płatności z paczki paypal	102
6.17. Metoda wywoływana po dokonaniu płatności	103
6.18. Endpoint wywoływany po dokonaniu płatności	105
6.19. Obiekt reprezentujący transakcję z systemu PayPal	105
7.1. Plik konfiguracyjny serwera skrajnego	117
7.2. Plik konfiguracyjny serwera apache - .htaccess	119
7.3. Konstruktor klasy qc_dbc	120
7.4. Metoda wykonująca zapytania do bazy danych	121
7.5. Przykładowe wywołanie funkcji exec	122
7.6. Przykładowe dane zwracane przez funkcję exec	122
7.7. Metoda tworząca połączenie z bazą danych	123
7.8. Dynamiczne modyfikowanie DOM	125
7.9. Funkcja dekodująca klucz sesji	126
7.10. Event usuwający przestarzałe sesje	126
7.11. Funkcja usuwająca przestarzałe sesje	127
7.12. Weryfikacja i pobranie informacji o sesji	128
7.13. Pierwsza wersja przekazywania informacji o domenie	129
7.14. Wykorzystanie HTTP_REFERER	130
7.15. Integracja modułu z bazą danych	132

7.16. Zapisanie pliku na serwerze skrajnym	133
7.17. Metoda tunelująca treści cz. 1	136
7.18. Metoda tunelująca treści cz. 2	137
7.19. Metoda tunelująca treści cz. 3	138
7.20. Metoda tunelująca treści cz. 4	139
7.21. Tablica konwerterów oraz lista predefiniowanych reguł	141
7.22. Pierwotna wersja konwertera URL_REPL	142
7.23. Konwerter HT_SANDBOX	143
7.24. Konwerter IMG_MIN	144
7.25. Konwerter CSS_MIN	145
7.26. Konwerter HT_ADS	146
7.27. Metoda usuwająca znaki specjalne z parametrów GET	148
7.28. Opisy poszczególnych kodów błędów	149
7.29. Dostarczenie zawartości z cache serwera skrajnego	154
7.30. Dostarczenie zawartości po pobraniu z serwera źródłowego	155