# Strategia Implementacji i Rekomendacje Techniczne dla Dedykowanego Serwera MCP dla Shopify

### Wprowadzenie

#### Cel Raportu

Niniejszy dokument przedstawia szczegółową analizę badawczą oraz strategię implementacyjną dla dedykowanego, prywatnego serwera Model Context Protocol (MCP) przeznaczonego do integracji z pojedynczą instancją sklepu Shopify. Raport ten stanowi techniczny plan działania dla wewnętrznego zespołu inżynierskiego, mający na celu umożliwienie budowy, wdrożenia i operacyjnego utrzymania bezpiecznego i wydajnego rozwiązania. Analiza koncentruje się na dostarczeniu praktycznych rekomendacji, konkretnych przykładów kodu, oceny ryzyka oraz estymacji pracochłonności dla każdego z kluczowych obszarów technicznych.

#### Streszczenie dla Kadry Zarządzającej

Analiza wskazuje, że optymalnym podejściem do budowy własnego serwera MCP jest wykorzystanie biblioteki fastmcp w języku Python, ze względu na jej wysokopoziomową abstrakcję i znaczące przyspieszenie prac deweloperskich. Rekomendowaną architekturą hostingową jest platforma Cloudflare Workers w połączeniu z Durable Objects do zarządzania stanem sesji, co zapewnia niemal zerowe opóźnienia przy zimnym starcie, globalną dystrybucję i zintegrowane mechanizmy bezpieczeństwa. Kluczowym wyzwaniem jest bezpieczne zarządzanie tokenem dostępowym do Admin API Shopify, który – z uwagi na ograniczenia platformy – wymaga wdrożenia manualnej procedury rotacji i przechowywania w dedykowanym menedżerze sekretów (np. AWS Secrets Manager). Architektura powinna od samego początku uwzględniać proaktywne mechanizmy zgodności z RODO, w tym automatyczną redakcję danych osobowych w logach, oraz rozbudowaną telemetrię, która posłuży nie tylko do monitorowania operacyjnego, ale również do iteracyjnego ulepszania inteligencji agenta AI.

#### Główne Zasady Architektoniczne

Proponowana architektura opiera się na czterech fundamentalnych zasadach, które gwarantują stworzenie solidnego, skalowalnego i bezpiecznego rozwiązania:

1. **Bezpieczeństwo od Podstaw (Security by Design):** Każdy komponent, od uwierzytelniania po logowanie, jest projektowany z myślą o minimalizacji powierzchni ataku i ochronie wrażliwych danych, w szczególności tokena Admin API.
2. **Prostota i Utrzymywalność (Simplicity and Maintainability):** Wybór technologii i wzorców projektowych priorytetyzuje czytelność, łatwość wdrożenia i niskie koszty utrzymania w długim terminie, unikając nadmiernej złożoności.
3. **Efektywność Kosztowa w Skali Pojedynczego Sklepu (Cost-Efficiency at Single-Store Scale):** Rekomendowane rozwiązania, w szczególności model serverless, są zoptymalizowane pod kątem minimalizacji kosztów operacyjnych dla aplikacji o zmiennym obciążeniu, typowym dla pojedynczego sklepu e-commerce.
4. **Szybkość Rozwoju (Developer Velocity):** Wykorzystanie wyspecjalizowanych frameworków i sprawdzonych wzorców ma na celu maksymalne skrócenie czasu potrzebnego na dostarczenie działającego produktu (MVP) i jego dalszy rozwój.

## 1. FastMCP / Prototyp: Minimalistyczny Serwer Narzędziowy JSON-RPC

### 1.1. Streszczenie

W tej sekcji szczegółowo opisano proces tworzenia szybkiego prototypu serwera MCP. Wykorzystana zostanie biblioteka fastmcp dla języka Python, której podejście oparte na dekoratorach znacząco przyspiesza rozwój poprzez abstrahowanie złożoności protokołu JSON-RPC.2 Prototyp zdefiniuje podstawowe narzędzie, zaimplementuje prosty, statyczny token do uwierzytelniania oraz skonfiguruje elementarne logowanie w celu walidacji pętli komunikacyjnej. Wybór

fastmcp jest strategiczną decyzją, która wpływa na szybkość i złożoność implementacji; w przeciwieństwie do generycznych bibliotek JSON-RPC 2.0 dla Node.js 4,

fastmcp jest narzędziem wyspecjalizowanym, które automatyzuje zadania specyficzne dla MCP, takie jak generowanie schematów z adnotacji typów w Pythonie.7

### 1.2. Rekomendowany Minimalny Stack/Architektura

* **Język/Framework:** Python 3.10+ z fastmcp.
* **Zależności:** uv jako menedżer pakietów, rekomendowany ze względu na wyższą wydajność w porównaniu do pip.8
* **Uwierzytelnianie:** Statyczny token typu "Bearer", weryfikowany w prostej funkcji pośredniczącej (middleware) lub wewnątrz dekoratora narzędzia.
* **Logowanie:** Wbudowany w Pythona moduł logging, skonfigurowany do generowania logów w ustrukturyzowanym formacie JSON na standardowe wyjście (stdout).

### 1.3. Przykładowy Minimalny Snippet/Polecenia

Poniższy przykład ilustruje kompletną, działającą implementację serwera prototypowego.

* **Kod serwera (mcp\_server.py):**  
  Python  
  import os  
  import logging  
  from fastmcp import FastMCP, Context, HTTPException  
    
  # Prosta konfiguracja logowania do JSON  
  logging.basicConfig(level=logging.INFO, format='{"timestamp": "%(asctime)s", "level": "%(levelname)s", "message": "%(message)s"}')  
  logger = logging.getLogger(\_\_name\_\_)  
    
  # Pobranie statycznego tokena ze zmiennej środowiskowej  
  STATIC\_AUTH\_TOKEN = os.environ.get("MCP\_STATIC\_TOKEN")  
  if not STATIC\_AUTH\_TOKEN:  
   raise ValueError("Zmienna środowiskowa MCP\_STATIC\_TOKEN nie jest ustawiona.")  
    
  mcp = FastMCP(name="Shopify MCP Prototype")  
    
  # Prosty dekorator do weryfikacji statycznego tokena  
  def require\_auth(func):  
   async def wrapper(ctx: Context, \*args, \*\*kwargs):  
   auth\_header = ctx.http\_request.headers.get("authorization")  
   if not auth\_header or not auth\_header.startswith("Bearer "):  
   raise HTTPException(status\_code=401, detail="Brak nagłówka autoryzacyjnego.")  
    
   token = auth\_header.split(" ")[1]  
   # Porównanie w stałym czasie w celu uniknięcia ataków timing attacks  
   if not hmac.compare\_digest(token, STATIC\_AUTH\_TOKEN):  
   raise HTTPException(status\_code=403, detail="Nieprawidłowy token.")  
    
   return await func(ctx, \*args, \*\*kwargs)  
   return wrapper  
    
  @mcp.tool()  
  @require\_auth  
  async def greet(ctx: Context, name: str) -> str:  
   """Zwraca proste powitanie dla podanego imienia."""  
   logger.info(f"Wywołano narzędzie 'greet' z imieniem: {name}")  
   return f"Hello, {name}!"  
    
  if \_\_name\_\_ == "\_\_main\_\_":  
   logger.info("Uruchamianie serwera MCP na porcie 8000...")  
   # FastMCP domyślnie używa uvicorn do uruchomienia serwera HTTP  
   mcp.run(transport="http", port=8000)
* **Polecenia uruchomienia i testowania (curl):**  
  Bash  
  # Ustawienie zmiennej środowiskowej z tokenem  
  export MCP\_STATIC\_TOKEN="your-secret-prototype-token"  
    
  # Uruchomienie serwera  
  uv run python mcp\_server.py  
    
  # Wywołanie narzędzia za pomocą curl (w nowym terminalu)  
  curl -X POST http://localhost:8000/mcp \  
  -H "Content-Type: application/json" \  
  -H "Authorization: Bearer your-secret-prototype-token" \  
  -d '{  
   "jsonrpc": "2.0",  
   "method": "greet",  
   "params": {"name": "World"},  
   "id": 1  
  }'

### 1.4. Bezpieczeństwo i Secrets Handling

Na etapie prototypu, statyczny token autoryzacyjny musi być ładowany wyłącznie ze zmiennej środowiskowej (MCP\_STATIC\_TOKEN). Jest to absolutnie kluczowe, aby zapobiec przypadkowemu umieszczeniu sekretu w systemie kontroli wersji.9 Aplikacja powinna odczytywać nagłówek

Authorization: Bearer <token> z przychodzących żądań i dokonywać porównania wartości tokena w sposób odporny na ataki typu "timing attack", na przykład przy użyciu funkcji hmac.compare\_digest w Pythonie.

### 1.5. Estymacja Pracy dla MVP w Dniach Roboczych

* **Całkowity czas:** 1–2 dni robocze.
* **Podział prac:**
  + Konfiguracja środowiska (Python, uv, fastmcp): 0.5 dnia.
  + Implementacja minimalnego kodu serwera z jednym narzędziem i statycznym uwierzytelnianiem: 0.5 dnia.
  + Testowanie, debugowanie i implementacja podstawowego logowania: 0.5 dnia.

### 1.6. Główne Ryzyka i Mitigacje

* **Ryzyko:** Biblioteka fastmcp, mimo swojej mocy, może stanowić wyzwanie dla zespołu, który nie jest zaznajomiony ze specyfikacją MCP lub asynchronicznym programowaniem w Pythonie.
  + **Mitygacja:** Dokumentacja biblioteki jest obszerna i zawiera liczne przykłady.2 Faza prototypowania została zaprojektowana jako bezpieczna przestrzeń do nauki i eksperymentowania z nową technologią.
* **Ryzyko:** Ekosystem Node.js oferuje dojrzałe biblioteki JSON-RPC, ale brakuje mu bezpośredniego odpowiednika wysokopoziomowego frameworka, jakim jest fastmcp. Wybór Pythona może być sprzeczny z dotychczasowymi kompetencjami zespołu.
  + **Mitygacja:** Zysk produktywności wynikający z użycia fastmcp dla tego konkretnego protokołu jest na tyle znaczący, że uzasadnia wykorzystanie Pythona, nawet w zespole zorientowanym na Node.js. Prostota prototypu minimalizuje wymagany poziom zaawansowania w Pythonie.

## 2. Bezpieczne Uwierzytelnianie dla Pojedynczego Sklepu: Zarządzanie Tokenem Admin API

### 2.1. Streszczenie

Ta sekcja koncentruje się na kluczowym wyzwaniu, jakim jest bezpieczne zarządzanie tokenem dostępowym do Admin API aplikacji typu Custom App w Shopify. Jest to poświadczenie o długim okresie ważności, które, w przypadku aplikacji tworzonych w panelu administracyjnym, nie może być rotowane programistycznie.11 To ograniczenie platformy Shopify stanowi centralne wyzwanie dla bezpieczeństwa całej architektury. Wymusza ono wdrożenie strategii, która w normalnych warunkach mogłaby wydawać się nadmiarowa dla aplikacji obsługującej jeden sklep. Rekomenduje się użycie dedykowanego, chmurowego menedżera sekretów (np. AWS Secrets Manager) do przechowywania tokena, wstrzykiwanie go do środowiska wykonawczego w momencie startu aplikacji oraz ustanowienie sformalizowanej, manualnej procedury rotacji.

### 2.2. Rekomendowany Minimalny Stack/Architektura

* **Przechowywanie Sekretów:** AWS Secrets Manager, Google Cloud Secret Manager lub HashiCorp Vault.
* **Mechanizm Dostępu:** Rola IAM przypisana do funkcji serverless lub kontenera musi posiadać uprawnienia o minimalnym zakresie (least privilege), pozwalające na odczyt *wyłącznie* tego konkretnego sekretu, który zawiera token Shopify.
* **Dostęp w Czasie Wykonania:** Kod aplikacji wykorzystuje SDK odpowiedniego dostawcy chmury do pobrania sekretu podczas inicjalizacji (w przypadku kontenerów) lub podczas zimnego startu (dla funkcji serverless). Pobrany token jest następnie przechowywany w pamięci przez cały cykl życia procesu lub wywołania. Taki mechanizm pozwala na rotację sekretu bez konieczności ponownego wdrażania aplikacji.12

### 2.3. Przykładowy Minimalny Snippet/Polecenia

* **Pseudo-kod (Python z AWS SDK boto3):**  
  Python  
  import boto3  
  from botocore.exceptions import ClientError  
  import json  
  import os  
    
  def get\_shopify\_admin\_token():  
   secret\_name = os.environ.get("SHOPIFY\_ADMIN\_TOKEN\_SECRET\_ARN")  
   region\_name = "eu-central-1" # Przykładowy region  
    
   session = boto3.session.Session()  
   client = session.client(  
   service\_name='secretsmanager',  
   region\_name=region\_name  
   )  
    
   try:  
   get\_secret\_value\_response = client.get\_secret\_value(  
   SecretId=secret\_name  
   )  
   except ClientError as e:  
   # Logowanie błędu i obsługa sytuacji krytycznej  
   logging.error(f"Nie można pobrać sekretu Shopify: {e}")  
   raise e  
   else:  
   # Sekrety mogą być przechowywane jako string lub binary  
   if 'SecretString' in get\_secret\_value\_response:  
   secret = get\_secret\_value\_response  
   # Zakładając, że sekret jest JSON-em z kluczem 'token'  
   return json.loads(secret)['shopify\_admin\_api\_token']  
   else:  
   # Obsługa sekretu binarnego, jeśli konieczne  
   pass
* **Polityka IAM (JSON):**  
  JSON  
  {  
   "Version": "2012-10-17",  
   "Statement":  
  }

### 2.4. Bezpieczeństwo i Secrets Handling

* **Zasada Najmniejszych Uprawnień:** Rola IAM powiązana z serwerem MCP nie może posiadać żadnych innych uprawnień poza tymi, które są absolutnie niezbędne do pobrania sekretu i zapisu logów.
* **Szyfrowanie w Spoczynku i w Transporcie:** Wybrany menedżer sekretów musi szyfrować token w spoczynku (at rest). Wszystkie wywołania SDK w celu pobrania sekretu muszą odbywać się za pośrednictwem protokołu TLS.
* **Procedura Rotacji Tokena:** Ze względu na ograniczenia Shopify, procedura musi być manualna, ale ściśle udokumentowana.11
  1. **W Panelu Admina Shopify:** Odinstaluj, a następnie ponownie zainstaluj aplikację Custom App, aby wygenerować nowy token dostępowy do Admin API.
  2. **W Menedżerze Sekretów:** Utwórz nową wersję sekretu, wklejając nowo wygenerowany token.
  3. **Restart Aplikacji:** Jeśli aplikacja działa w kontenerze, wywołaj kroczący restart usługi, aby zmusić ją do pobrania nowej wersji sekretu. Funkcje serverless automatycznie pobiorą nowy token przy następnym zimnym starcie.

### 2.5. Estymacja Pracy dla MVP w Dniach Roboczych

* **Całkowity czas:** 2 dni robocze.
* **Podział prac:**
  + Konfiguracja sekretu w wybranym menedżerze i przypisanie uprawnień IAM: 1 dzień.
  + Integracja kodu SDK do pobierania sekretu w czasie wykonania, wraz z obsługą błędów: 1 dzień.

### 2.6. Główne Ryzyka i Mitigacje

* **Ryzyko:** Manualny proces rotacji tokena jest podatny na błędy ludzkie i może spowodować przerwę w działaniu usługi, jeśli nie zostanie wykonany poprawnie.
  + **Mitygacja:** Stworzenie szczegółowej, krok po kroku instrukcji (runbook) dla procedury rotacji. Automatyzacja elementów, które można zautomatyzować (np. skrypt restartujący aplikację). Przeprowadzanie okresowych testów procedury rotacji w środowisku deweloperskim.
* **Ryzyko:** Długi czas życia tokena czyni go celem o wysokiej wartości. Kompromitacja środowiska chmurowego może prowadzić do wycieku tokena.
  + **Mitygacja:** Rygorystyczne przestrzeganie zasady najmniejszych uprawnień dla ról IAM. Wdrożenie solidnego monitoringu bezpieczeństwa chmury (np. AWS GuardDuty, CloudTrail) w celu wykrywania anomalii w aktywności.

## 3. Hosting: Cloudflare Workers vs. Kontener Node.js (Fly.io)

### 3.1. Streszczenie

Ta sekcja przedstawia analizę porównawczą dwóch wiodących opcji hostingowych dla serwera MCP: Cloudflare Workers z Durable Objects do zarządzania stanem oraz standardowy kontener (Node.js lub Python) na platformie takiej jak Fly.io. Dla aplikacji obsługującej pojedynczy sklep, która wymaga interaktywności, Cloudflare Workers oferuje znacznie lepszą wydajność przy zimnym starcie oraz unikalny model dla stanowych sesji na brzegu sieci (edge) za pomocą Durable Objects.14 Chociaż stale działający kontener na Fly.io może osiągnąć niższe opóźnienia dla "ciepłych" żądań 15, stos technologiczny Cloudflare stanowi bardziej zintegrowane, natywnie serverlessowe rozwiązanie, które lepiej pasuje do zdarzeniowej natury agenta MCP, zwłaszcza pod kątem wsparcia dla Server-Sent Events (SSE) i zredukowanego narzutu operacyjnego. Wybór ten nie jest prostą decyzją "edge kontra kontener", lecz wyborem między wyspecjalizowanym, zintegrowanym narzędziem (Workers + DO) a rozwiązaniem ogólnego przeznaczenia, które trzeba odpowiednio skonfigurować (kontener z utrzymywaniem stanu w pamięci).

### 3.2. Rekomendowany Minimalny Stack/Architektura

* **Główna rekomendacja:** Cloudflare Workers z powiązaniem Durable Object do zarządzania stanem sesji (np. ID koszyka).
* **Alternatywa:** Kontener Node.js (lub Python) na Fly.io, skonfigurowany z min\_machines\_running = 1, aby wyeliminować zimne starty i zapewnić stałe, niskie opóźnienia.

### 3.3. Przykładowy Minimalny Snippet/Polecenia

* **Cloudflare wrangler.toml:**  
  Ini, TOML  
  name = "shopify-mcp-worker"  
  main = "src/index.ts"  
  compatibility\_date = "2024-01-01"  
    
  [[durable\_objects.bindings]]  
  name = "SESSION\_STATE"  
  class\_name = "SessionState"  
    
  [[migrations]]  
  tag = "v1"  
  new\_classes =
* **Klasa Durable Object (TypeScript):**  
  TypeScript  
  export class SessionState {  
   state: DurableObjectState;  
    
   constructor(state: DurableObjectState) {  
   this.state = state;  
   }  
    
   async fetch(request: Request) {  
   // Przykład: zarządzanie stanem sesji (np. ID koszyka Shopify)  
   const url = new URL(request.url);  
   const path = url.pathname;  
    
   if (path === "/cart/get") {  
   const cartId: string | undefined = await this.state.storage.get("cartId");  
   return new Response(JSON.stringify({ cartId }), {  
   headers: { "Content-Type": "application/json" },  
   });  
   } else if (path === "/cart/set" && request.method === "POST") {  
   const { cartId } = await request.json();  
   await this.state.storage.put("cartId", cartId);  
   return new Response("OK");  
   }  
    
   return new Response("Not found", { status: 404 });  
   }  
  }
* **Fly.io fly.toml:**  
  Ini, TOML  
  app = "shopify-mcp-container"  
  primary\_region = "ams"  
    
  [build]  
   builder = "paketobuildpacks/builder:base"  
    
  [http\_service]  
   internal\_port = 8080  
   force\_https = true  
   auto\_stop\_machines = false # Kluczowe dla eliminacji zimnych startów  
   auto\_start\_machines = true  
   min\_machines\_running = 1 # Gwarantuje, że co najmniej jedna maszyna jest zawsze aktywna

### 3.4. Bezpieczeństwo i Secrets Handling

* **Cloudflare Workers:** Sekrety (takie jak ARN tokena Shopify z menedżera sekretów) powinny być przechowywane jako zaszyfrowane sekrety przy użyciu Wrangler CLI (wrangler secret put). Worker ma do nich dostęp jako do zmiennych globalnych w środowisku wykonawczym.
* **Fly.io:** Sekrety powinny być zarządzane za pomocą fly secrets set. Są one wstrzykiwane jako zmienne środowiskowe do kontenera w czasie wykonania. Dla tokena Admin API Shopify nadal obowiązuje najlepsza praktyka z Sekcji 2 (pobieranie z menedżera sekretów).

### 3.5. Estymacja Pracy dla MVP w Dniach Roboczych

* **Cloudflare Workers:** 3 dni robocze (uwzględnia czas na zapoznanie się z modelem Durable Objects).
* **Kontener Fly.io:** 2 dni robocze (bardziej znany paradygmat dla większości deweloperów).

### 3.6. Główne Ryzyka i Mitigacje

* **Ryzyko (Cloudflare):** Model programowania Durable Objects (oparty na modelu aktora) może być nieznany i wprowadza dodatkową złożoność w porównaniu do tradycyjnego serwera stanowego. Funkcja hibernacji WebSocketów wymaga starannego zarządzania stanem, aby uniknąć utraty danych.16
  + **Mitygacja:** Rozpoczęcie pracy od oficjalnych tutoriali i prostych przypadków użycia. W ramach MVP należy unikać zaawansowanych funkcji hibernacji i skupić się na podstawowym, transakcyjnym przechowywaniu stanu sesji.
* **Ryzyko (Fly.io):** Błędna konfiguracja reguł skalowania (np. zezwolenie na skalowanie do zera) wprowadzi znaczące opóźnienia związane z zimnym startem (rzędu 1.5s), co jest nieakceptowalne dla interfejsu konwersacyjnego.15
  + **Mitygacja:** Jawne ustawienie min\_machines\_running = 1 w pliku fly.toml i monitorowanie metryk latencji p95/p99 w celu upewnienia się, że nie występują zimne starty.
* **Ryzyko (Oba):** Połączenia z bazą danych ze środowiska serverless/edge mogą wprowadzać dodatkowe opóźnienia.
  + **Mitygacja:** Proponowana architektura nie wymaga tradycyjnej bazy danych, ponieważ stan jest zarządzany przez Durable Objects lub w pamięci kontenera. Głównym źródłem opóźnień są wywołania API do Shopify, czego nie da się uniknąć.

### Porównanie Architektur Hostingowych (MVP dla Pojedynczego Sklepu)

| Cecha | Cloudflare Workers + Durable Objects | Kontener Fly.io (min\_machines\_running=1) | Rekomendacja i Uzasadnienie |
| --- | --- | --- | --- |
| **Opóźnienie zimnego startu** | Prawie zerowe (<10 ms) | Wyeliminowane (zawsze "ciepły") | **Cloudflare** wygrywa dla prawdziwego skalowania do zera i nieprzewidywalnego ruchu. |
| **Opóźnienie "ciepłego" żądania** | Niskie (średnio ∼180 ms globalnie) | Bardzo niskie (średnio ∼60 ms regionalnie) | **Fly.io** wygrywa, jeśli baza użytkowników znajduje się w jednym regionie. |
| **Sesje stanowe** | Natywne, transakcyjne, trwałe przechowywanie danych przez Durable Objects. | Stan w pamięci (tracony przy restarcie) lub wymaga zewnętrznej bazy/cache. | **Cloudflare** wygrywa dzięki solidnemu, wbudowanemu mechanizmowi zarządzania stanem. |
| **Model kosztowy (skala MVP)** | Płatność za żądanie/czas/przechowywanie. Hojny darmowy plan. Prawdopodobnie <5 USD/mies. | Stały miesięczny koszt za działającą maszynę wirtualną (np. ∼5−10 USD/mies. za małą instancję). | **Cloudflare** jest prawdopodobnie tańszy i skaluje się do zera kosztów, jeśli nie jest używany. |
| **Wsparcie dla SSE** | Natywne wsparcie przez TransformStream. Dobrze udokumentowane.18 | Wspierane, ale może podlegać problemom z buforowaniem przez proxy na niektórych platformach.20 | **Cloudflare** oferuje bardziej niezawodną, natywną implementację. |
| **Doświadczenie deweloperskie** | Wrangler CLI, orientacja na TypeScript/JS. Nieznany model aktora dla stanu. | Oparte na Dockerze, niezależne od języka. Znany paradygmat serwera stanowego. | **Fly.io** jest bardziej znajome, ale platforma Cloudflare jest bardziej zintegrowana dla tego przypadku użycia. |

## 4. Implementacja Podstawowych Narzędzi MCP

### 4.1. Streszczenie

Ta sekcja zawiera projekty implementacyjne dla trzech kluczowych narzędzi Shopify, które będą udostępniane przez serwer MCP. Narzędzie search\_shop\_catalog będzie korzystać z publicznego Storefront GraphQL API w celu zapewnienia szybkości i uniknięcia ujawniania wrażliwych danych. Narzędzie update\_cart również będzie używać Storefront API, wykorzystując jego mutację cartLinesAdd. W przeciwieństwie do nich, narzędzie get\_order\_status wymaga dostępu do wrażliwych danych klientów i zamówień, dlatego musi korzystać z Admin GraphQL API, zabezpieczonego tokenem zarządzanym zgodnie z wytycznymi z Sekcji 2. Każda implementacja zostanie przedstawiona w formie pseudo-kodu z wyraźnie zaznaczoną logiką obsługi błędów.

### 4.2. Rekomendowany Minimalny Stack/Architektura

* **Interakcja z API:** Lekka biblioteka klienta HTTP w wybranym języku (np. httpx dla Pythona, node-fetch dla Node.js).
* **Klient GraphQL:** Dla MVP nie jest potrzebna ciężka biblioteka kliencka; wystarczą proste żądania POST z ciałem JSON zawierającym zapytanie GraphQL.

### 4.3. Przykładowy Minimalny Snippet/Polecenia

* **Narzędzie: search\_shop\_catalog (Storefront API):**
  + **Pseudo-kod (Python):**  
    Python  
    import httpx  
      
    STOREFRONT\_API\_URL = f"https://{SHOPIFY\_DOMAIN}/api/2024-07/graphql.json"  
    STOREFRONT\_ACCESS\_TOKEN = os.environ.get("SHOPIFY\_STOREFRONT\_ACCESS\_TOKEN")  
      
    async def search\_shop\_catalog(query: str, first: int = 10) -> dict:  
     graphql\_query = """  
     query searchProducts($query: String, $first: Int) {  
     products(first: $first, query: $query) {  
     edges {  
     node {  
     id  
     title  
     handle  
     description  
     priceRange {  
     minVariantPrice { amount, currencyCode }  
     }  
     }  
     }  
     }  
     }  
     """  
     variables = {"query": f"title:\*{query}\*", "first": first}  
     headers = {"X-Shopify-Storefront-Access-Token": STOREFRONT\_ACCESS\_TOKEN}  
      
     async with httpx.AsyncClient() as client:  
     response = await client.post(STOREFRONT\_API\_URL, json={"query": graphql\_query, "variables": variables}, headers=headers)  
     response.raise\_for\_status()  
     data = response.json()  
     # Należy obsłużyć błędy zwracane przez GraphQL API  
     if "errors" in data:  
     raise Exception(f"Błąd GraphQL: {data['errors']}")  
     return data["data"]["products"]
  + **Uzasadnienie:** Użycie Storefront API jest właściwe dla publicznie dostępnych danych katalogowych.21 Zapytanie  
    products obsługuje parametr query do filtrowania, w tym po tytule.23
* **Narzędzie: update\_cart (Storefront API):**
  + **Pseudo-kod (Python):**  
    Python  
    async def update\_cart(cart\_id: str, merchandise\_id: str, quantity: int) -> dict:  
     graphql\_mutation = """  
     mutation addLinesToCart($cartId: ID!, $lines: [CartLineInput!]!) {  
     cartLinesAdd(cartId: $cartId, lines: $lines) {  
     cart {  
     id  
     lines(first: 50) {  
     edges { node { id quantity merchandise {... on ProductVariant { title } } } }  
     }  
     cost { totalAmount { amount } }  
     }  
     userErrors {  
     field  
     message  
     }  
     }  
     }  
     """  
     variables = {  
     "cartId": cart\_id,  
     "lines": [{"merchandiseId": merchandise\_id, "quantity": quantity}]  
     }  
     headers = {"X-Shopify-Storefront-Access-Token": STOREFRONT\_ACCESS\_TOKEN}  
      
     async with httpx.AsyncClient() as client:  
     response = await client.post(STOREFRONT\_API\_URL, json={"query": graphql\_mutation, "variables": variables}, headers=headers)  
     response.raise\_for\_status()  
     data = response.json()  
     if data["data"]["cartLinesAdd"]["userErrors"]:  
     raise Exception(f"Błąd użytkownika GraphQL: {data['data']['cartLinesAdd']['userErrors']}")  
     return data["data"]["cartLinesAdd"]["cart"]
  + **Uzasadnienie:** Mutacje do tworzenia i modyfikacji koszyka (cartCreate, cartLinesAdd) są częścią Storefront API, a nie Admin API, ponieważ operują w kontekście sesji klienta.25
* **Narzędzie: get\_order\_status (Admin API):**
  + **Pseudo-kod (Python):**  
    Python  
    ADMIN\_API\_URL = f"https://{SHOPIFY\_DOMAIN}/admin/api/2024-07/graphql.json"  
      
    async def get\_order\_status(order\_id: str) -> dict:  
     # Token pobierany bezpiecznie w czasie wykonania  
     admin\_token = get\_shopify\_admin\_token()   
      
     graphql\_query = """  
     query getOrder($id: ID!) {  
     order(id: $id) {  
     id  
     name  
     displayFinancialStatus  
     displayFulfillmentStatus  
     statusPageUrl  
     }  
     }  
     """  
     variables = {"id": order\_id}  
     headers = {"X-Shopify-Access-Token": admin\_token}  
      
     async with httpx.AsyncClient() as client:  
     response = await client.post(ADMIN\_API\_URL, json={"query": graphql\_query, "variables": variables}, headers=headers)  
     response.raise\_for\_status()  
     data = response.json()  
     if "errors" in data:  
     raise Exception(f"Błąd GraphQL: {data['errors']}")  
     return data["data"]["order"]
  + **Uzasadnienie:** Zapytanie order w Admin API jest właściwym sposobem na pobranie szczegółowych danych zamówienia, w tym statusów.26 Kluczowe pola to  
    displayFinancialStatus i displayFulfillmentStatus.26

### 4.4. Bezpieczeństwo i Secrets Handling

* **Segregacja Punktów Końcowych API:** Logika aplikacji musi wyraźnie rozróżniać wywołania do publicznego Storefront API i zabezpieczonego Admin API.
* **Użycie Tokenów:** Storefront Access Token jest tokenem publicznym i może być częścią konfiguracji. Admin Access Token jest wysoce wrażliwy i musi być pobierany z menedżera sekretów wyłącznie w czasie wykonania, zgodnie z opisem w Sekcji 2. Nigdy nie może być logowany ani w żaden inny sposób ujawniany.

### 4.5. Estymacja Pracy dla MVP w Dniach Roboczych

* **Całkowity czas:** 3–4 dni robocze.
* **Podział prac:**
  + search\_shop\_catalog: 1 dzień.
  + update\_cart (wraz z logiką tworzenia koszyka): 1.5 dnia.
  + get\_order\_status (wraz z obsługą bezpiecznego tokena): 1 dzień.

### 4.6. Główne Ryzyka i Mitigacje

* **Ryzyko:** Różne API Shopify mają odmienne schematy ograniczania liczby żądań (rate limiting). Admin API używa obliczonego kosztu zapytania, podczas gdy Storefront API jest generalnie nieograniczone, ale dławi tworzenie checkoutów.29 Niezoptymalizowane zapytania mogą prowadzić do throttlingu.
  + **Mitygacja:** Projektowanie zapytań do Admin API w sposób wydajny, żądając tylko niezbędnych pól. Implementacja mechanizmu ponawiania prób z wykładniczym czasem oczekiwania (exponential backoff) dla odpowiedzi 429 Too Many Requests. Logowanie obiektu extensions.cost z odpowiedzi GraphQL w celu monitorowania kosztów zapytań.
* **Ryzyko:** Nieprawidłowa obsługa błędów userErrors w GraphQL może prowadzić do cichych niepowodzeń, gdzie narzędzie pozornie działa poprawnie, ale operacja w Shopify się nie powiodła.
  + **Mitygacja:** Wszystkie procedury obsługi wywołań API muszą jawnie sprawdzać obecność tablicy userErrors w odpowiedzi GraphQL i zgłaszać wyjątek na poziomie aplikacji, jeśli nie jest ona pusta, logując komunikaty błędów w celu debugowania.

## 5. Telemetria: Lekka Struktura Obserwowalności

### 5.1. Streszczenie

Ta sekcja określa prostą, ale skuteczną strategię telemetrii do monitorowania stanu i wydajności agenta MCP. Fundamentem jest ustrukturyzowane logowanie w formacie JSON, co umożliwia łatwe parsowanie i analizę. Zdefiniowany zostanie minimalny schemat logów oraz kluczowe metryki do śledzenia, takie jak opóźnienie zapytań, użycie narzędzi i zapytania bez odpowiedzi. Telemetria ta nie służy jedynie celom operacyjnym; jest to podstawowa pętla sprzężenia zwrotnego niezbędna do iteracyjnego ulepszania inteligencji i skuteczności agenta AI.31

### 5.2. Rekomendowany Minimalny Stack/Architektura

* **Format Logów:** Ustrukturyzowany JSON zapisywany na stdout.
* **Usługa Logowania:** Lekka, przyjazna dla środowisk serverless usługa agregacji logów, taka jak Axiom, Logtail, lub wbudowane usługi wybranego dostawcy chmury (np. Cloudflare Logpush, AWS CloudWatch Logs).
* **Metryki:** Wyprowadzane z zapytań do logów w ramach usługi logowania. Dla MVP nie jest potrzebny osobny agent metryk.

### 5.3. Przykładowy Minimalny Snippet/Polecenia

* **Konfiguracja logging w Pythonie:**  
  Python  
  import logging  
  import json  
  from pythonjsonlogger import jsonlogger  
    
  logger = logging.getLogger()  
  logHandler = logging.StreamHandler()  
  formatter = jsonlogger.JsonFormatter('%(asctime)s %(name)s %(levelname)s %(message)s')  
  logHandler.setFormatter(formatter)  
  logger.addHandler(logHandler)  
  logger.setLevel(logging.INFO)  
    
  # Przykład użycia  
  logger.info("Uruchomiono serwer", extra={'service': 'mcp-server'})
* **Przykładowe Zapytanie do Logów (pseudo-SQL):**  
  SQL  
  -- Obliczenie średniego opóźnienia dla narzędzia search\_shop\_catalog  
  SELECT avg(duration\_ms)   
  FROM logs   
  WHERE json.tool\_name = 'search\_shop\_catalog' AND json.status = 'SUCCESS'  
    
  -- Zliczenie zapytań bez odpowiedzi w ciągu ostatniej godziny  
  SELECT count(\*)  
  FROM logs  
  WHERE json.response\_source = 'NO\_MATCH' AND timestamp > now() - 1h

### 5.4. Bezpieczeństwo i Secrets Handling

* **Redakcja PII:** Framework logowania musi zawierać filtr lub procesor do redakcji danych osobowych (PII) przed zapisaniem logów. Jest to kluczowy środek bezpieczeństwa i zgodności z RODO (szczegółowo omówiony w Sekcji 7).
* **Poświadczenia Usługi Logowania:** Klucz API do usługi przesyłania logów powinien być przechowywany jako sekret (np. w sekretach Cloudflare Worker lub Fly.io), a nie w kodzie źródłowym.

### 5.5. Estymacja Pracy dla MVP w Dniach Roboczych

* **Całkowity czas:** 2 dni robocze.
* **Podział prac:**
  + Konfiguracja loggera JSON: 0.5 dnia.
  + Instrumentacja kodu w celu emitowania logów w kluczowych punktach (początek/koniec żądania, wywołanie narzędzia, błędy): 1 dzień.
  + Konfiguracja usługi agregacji logów i stworzenie podstawowych pulpitów nawigacyjnych/alertów: 0.5 dnia.

### 5.6. Główne Ryzyka i Mitigacje

* **Ryzyko:** Nadmierne logowanie może stać się kosztowne i generować szum, utrudniając znalezienie ważnych informacji.
  + **Mitygacja:** Rozpoczęcie od minimalnego zestawu logów na poziomie INFO dla kluczowych zdarzeń. Używanie poziomu DEBUG dla szczegółowych informacji i włączanie go w produkcji tylko na krótki czas podczas rozwiązywania konkretnych problemów.
* **Ryzyko:** Niespójne schematy logów w różnych częściach aplikacji utrudniają tworzenie zapytań.
  + **Mitygacja:** Scentralizowanie konfiguracji loggera i używanie współdzielonej biblioteki lub funkcji pomocniczej do emitowania logów, co zapewni, że schemat będzie zawsze stosowany poprawnie.

### Minimalny Schemat Logów JSON

| Nazwa Pola | Typ | Opis | Przykład |
| --- | --- | --- | --- |
| timestamp | String (ISO 8601) | Znacznik czasu zdarzenia. | "2025-10-26T10:00:00.123Z" |
| level | String | Poziom logu (INFO, WARN, ERROR). | "INFO" |
| message | String | Czytelny dla człowieka komunikat logu. | "Tool call successful" |
| correlation\_id | String (UUID) | Unikalny identyfikator dla pojedynczego żądania użytkownika. | "a1b2c3d4-..." |
| session\_id | String | Identyfikator sesji konwersacyjnej użytkownika. | "sess\_xyz789" |
| source\_ip\_hash | String | Hash SHA256 adresu IP źródła (dla prywatności). | "a665a4..." |
| user\_query | String (zredagowany) | Zapytanie wejściowe użytkownika z usuniętymi PII. | "Jaki jest status mojego zamówienia?" |
| tool\_name | String | Nazwa wywołanego narzędzia MCP. | "get\_order\_status" |
| tool\_params | Object (zredagowany) | Parametry przekazane do narzędzia z usuniętymi PII. | {"order\_id": ""} |
| status | String | Wynik operacji (SUCCESS, FAILURE, FALLBACK). | "SUCCESS" |
| duration\_ms | Integer | Czas trwania operacji w milisekundach. | 150 |
| error\_message | String | Komunikat błędu, jeśli status to FAILURE. | "Order not found" |
| response\_source | String | Źródło ostatecznej odpowiedzi. | "TOOL\_GET\_ORDER\_STATUS" |

## 6. Rate Limiting i Ochrona przed Nadużyciami

### 6.1. Streszczenie

Ta sekcja porównuje dwa komplementarne podejścia do ograniczania liczby żądań: egzekwowanie na poziomie platformy przy użyciu funkcji bezpieczeństwa Cloudflare oraz kontrolę na poziomie aplikacji za pomocą oprogramowania pośredniczącego (middleware). Te dwie metody nie są konkurencyjne, lecz stanowią uzupełniające się warstwy ochrony. Cloudflare działa jako tarcza, zapewniając solidną pierwszą linię obrony przed atakami DDoS i brute-force, blokując złośliwy ruch na brzegu sieci, zanim dotrze on do aplikacji.33 Z kolei middleware na poziomie aplikacji oferuje bardziej granularną, kontekstową kontrolę, egzekwując logikę biznesową i zasady sprawiedliwego użytkowania, działając jak kontroler ruchu.35 W celu zapewnienia kompleksowej strategii rekomenduje się stosowanie obu tych podejść.

### 6.2. Rekomendowany Minimalny Stack/Architektura

* **Poziom Platformy (Rekomendowane):** Cloudflare WAF z regułami Rate Limiting.
* **Poziom Aplikacji (Opcjonalne, ale Zalecane):**
  + **Node.js:** Middleware express-rate-limit.
  + **Python (w kontekście FastAPI/Starlette):** Middleware takie jak slowapi.
* **Przechowywanie dla Limitera na Poziomie Aplikacji:** W przypadku kontenera z jedną instancją wystarczy magazyn w pamięci. Dla środowisk wieloinstancyjnych lub serverless wymagany jest zewnętrzny magazyn, taki jak Redis lub Cloudflare KV. W przypadku aplikacji na Workers, do śledzenia limitów na sesję lub IP można użyć Durable Objects.

### 6.3. Przykładowy Minimalny Snippet/Polecenia

* **Reguła Rate Limiting w Cloudflare (opis konfiguracji w panelu):**
  + **Nazwa reguły:** Limit\_MCP\_Endpoint\_Per\_IP
  + **Warunek (If incoming requests match...):** URI Path equals /mcp
  + **Charakterystyka (With the same characteristics):** IP Source Address
  + **Limit (When rate exceeds):** 100 żądań na 1 minutę.
  + **Akcja (Then take action):** Block na 10 minut.
  + **Uzasadnienie:** Konfiguracja reguł w Cloudflare opiera się na charakterystykach (jak IP), okresie, liczbie żądań na okres oraz akcji i czasie jej trwania.34
* **Middleware express-rate-limit (JavaScript):**  
  JavaScript  
  const rateLimit = require('express-rate-limit');  
    
  const mcpLimiter = rateLimit({  
   windowMs: 15 \* 60 \* 1000, // 15 minut  
   max: 100, // Limit 100 żądań na IP w ciągu 15 minut  
   standardHeaders: true, // Zwraca informacje o limicie w nagłówkach `RateLimit-\*`  
   legacyHeaders: false, // Wyłącza nagłówki `X-RateLimit-\*`  
   message: 'Too many requests from this IP, please try again after 15 minutes'  
  });  
    
  // Zastosowanie middleware do konkretnego endpointu  
  app.post('/mcp', mcpLimiter, (req, res) => {  
   // Logika obsługi żądania MCP  
  });  
  + **Uzasadnienie:** Biblioteka express-rate-limit jest konfigurowana za pomocą parametrów windowMs i max w celu kontroli częstotliwości żądań.35

### 6.4. Bezpieczeństwo i Secrets Handling

* **Zaufanie do Adresu IP:** Gdy aplikacja znajduje się za Cloudflare, rzeczywisty adres IP klienta jest zazwyczaj przekazywany w nagłówku CF-Connecting-IP. Limiter na poziomie aplikacji musi być skonfigurowany tak, aby ufał temu nagłówkowi pochodzącemu z proxy Cloudflare, co zapobiega fałszowaniu adresu IP.

### 6.5. Estymacja Pracy dla MVP w Dniach Roboczych

* **Całkowity czas:** 1 dzień roboczy.
* **Podział prac:**
  + Konfiguracja podstawowej reguły rate limiting w Cloudflare: 0.5 dnia.
  + Implementacja prostego, opartego na pamięci, limitera na poziomie aplikacji: 0.5 dnia.

### 6.6. Główne Ryzyka i Mitigacje

* **Ryzyko:** Zbyt agresywne ograniczanie liczby żądań może blokować legalnych użytkowników.
  + **Mitygacja:** Rozpoczęcie od łagodnych limitów i monitorowanie logów pod kątem błędów 429 Too Many Requests. Stopniowe zaostrzanie limitów w oparciu o obserwowane wzorce ruchu. Użycie akcji "Log" w Cloudflare do testowania reguł bez blokowania ruchu.
* **Ryzyko:** Ograniczanie oparte wyłącznie na adresie IP jest nieskuteczne przeciwko rozproszonym atakom i może niesprawiedliwie karać użytkowników korzystających ze współdzielonego NAT.
  + **Mitygacja:** Z tego powodu zalecane jest podejście wielowarstwowe. Użycie ograniczeń opartych na IP na brzegu sieci dla szerokiej ochrony, a na poziomie aplikacji wdrożenie bardziej granularnych ograniczeń opartych na sesji lub identyfikatorze użytkownika, aby sprawiedliwie obsługiwać uwierzytelniony ruch.

## 7. RODO i Prywatność od Podstaw

### 7.1. Streszczenie

Ta sekcja określa kluczowe kroki w celu zapewnienia zgodności serwera MCP z RODO, koncentrując się na zasadach ograniczenia przechowywania i minimalizacji danych. Zastosowanie proaktywnej anonimizacji przekształca zgodność z prawem z obciążenia operacyjnego w zautomatyzowaną, techniczną cechę systemu. Takie podejście architektoniczne znacząco upraszcza realizację Praw Podmiotów Danych (DSAR), redukuje ryzyko i obniża koszty operacyjne związane z przestrzeganiem przepisów.37 Zaproponowana zostanie konkretna polityka retencji logów, strategia redakcji Danych Osobowych (PII) w momencie ich pozyskiwania oraz procedura obsługi wniosków DSAR.

### 7.2. Rekomendowany Minimalny Stack/Architektura

* **Redakcja PII:** Biblioteka Python, taka jak presidio-analyzer lub szybsza datafog, do skanowania i redagowania/anonimizacji PII z komunikatów logów przed ich zapisaniem.37
* **Przepływ Pracy DSAR:** Udokumentowana procedura wewnętrzna, zarządzana za pomocą systemu ticketowego lub dedykowanego rejestru DSAR.
* **Przechowywanie Logów:** System zarządzania logami, który wspiera konfigurowalne polityki retencji i bezpieczne usuwanie danych.

### 7.3. Przykładowy Minimalny Snippet/Polecenia

* **Redakcja PII (Pseudo-kod w Pythonie):**  
  Python  
  from datafog import TextService  
    
  # Inicjalizacja serwisu do detekcji PII  
  pii\_service = TextService(engine="smart")  
    
  def redact\_pii\_from\_log\_dict(log\_data: dict) -> dict:  
   """Przechodzi przez słownik logu i redaguje wartości PII."""  
   redacted\_data = {}  
   for key, value in log\_data.items():  
   if isinstance(value, str):  
   # Użycie DataFog do detekcji i zastąpienia PII  
   # W praktyce można zaimplementować bardziej zaawansowaną logikę  
   # np. hashowanie zamiast prostego zastępowania  
   annotated = pii\_service.annotate\_text\_sync(value)  
   if annotated:  
   # Prosty przykład redakcji - zastąpienie wykrytych wartości  
   for pii\_type, pii\_values in annotated.items():  
   for pii\_val in pii\_values:  
   value = value.replace(pii\_val, f"")  
   redacted\_data[key] = value  
   elif isinstance(value, dict):  
   redacted\_data[key] = redact\_pii\_from\_log\_dict(value)  
   else:  
   redacted\_data[key] = value  
   return redacted\_data  
    
  # Przykład użycia w loggerze  
  log\_record = {  
   "message": "User query for order",  
   "user\_query": "Jaki jest status zamówienia dla jan.kowalski@example.com?",  
   "tool\_params": {"email": "jan.kowalski@example.com"}  
  }  
  redacted\_log = redact\_pii\_from\_log\_dict(log\_record)  
  # logger.info(redacted\_log)  
  + **Uzasadnienie:** Biblioteki takie jak datafog i presidio mogą wykrywać szeroki zakres encji PII.37 Redakcja może polegać na zastąpieniu stałym ciągiem znaków, hashem lub tagiem.39
* **Zarys Procedury DSAR:**
  1. **Odbiór Wniosku:** Wniosek może wpłynąć dowolnym kanałem (e-mail, czat).38
  2. **Weryfikacja Tożsamości Wnioskodawcy:** Kluczowy krok w celu uniknięcia naruszenia danych.38
  3. **Rejestracja Wniosku:** Zapisanie wniosku w dedykowanym rejestrze DSAR.
  4. **Przeszukanie Logów:** Wyszukanie logów na podstawie identyfikatora użytkownika (np. zhashowanego IP, ID sesji).
  5. **Zestawienie Danych:** Zebranie (już zredagowanych) danych z logów.
  6. **Przegląd pod kątem Danych Osób Trzecich:** Dodatkowa weryfikacja, czy odpowiedź nie ujawnia danych innych osób.42
  7. **Bezpieczne Dostarczenie Odpowiedzi:** Przekazanie danych wnioskodawcy w bezpieczny sposób.
  8. **Zarejestrowanie Zakończenia:** Odnotowanie zamknięcia wniosku w rejestrze DSAR.

### 7.4. Bezpieczeństwo i Secrets Handling

* **Minimalizacja Danych:** Schemat logów (Sekcja 5) powinien być zaprojektowany tak, aby zbierać tylko niezbędne informacje. Należy unikać logowania wrażliwych danych, takich jak pełne adresy IP czy numery kart kredytowych.43
* **Bezpieczne Usuwanie:** System zarządzania logami musi być skonfigurowany do automatycznego i trwałego usuwania logów po upływie okresu ich retencji, zgodnie z polityką.

### 7.5. Estymacja Pracy dla MVP w Dniach Roboczych

* **Całkowity czas:** 3–5 dni roboczych.
* **Podział prac:**
  + Zdefiniowanie i udokumentowanie polityki retencji i procedur DSAR: 1 dzień.
  + Implementacja filtra redakcji PII w potoku logowania (w tym testowanie pod kątem fałszywych pozytywów/negatywów): 2–3 dni.
  + Konfiguracja reguł retencji w usłudze zarządzania logami: 0.5 dnia.

### 7.6. Główne Ryzyka i Mitigacje

* **Ryzyko:** Narzędzia do wykrywania PII mogą generować fałszywe pozytywy (redagując dane niewrażliwe) lub fałszywe negatywy (pomijając PII).
  + **Mitygacja:** Dokładne przetestowanie biblioteki do redakcji PII na reprezentatywnych danych logów. Rozpoczęcie od konserwatywnej polityki, która redaguje szerszy zestaw potencjalnych PII. Regularny przegląd próbki zredagowanych logów w celu dostrojenia reguł wykrywania. Najlepszą mitygacją jest unikanie logowania PII w pierwszej kolejności, tam gdzie to możliwe.37
* **Ryzyko:** Brak odpowiedzi na wniosek DSAR w ustawowym terminie jednego miesiąca może prowadzić do kar regulacyjnych.
  + **Mitygacja:** Wdrożenie jasnej i dobrze udokumentowanej procedury DSAR. Proaktywna redakcja PII znacznie przyspiesza i upraszcza proces zestawiania danych, skracając czas potrzebny na realizację wniosku.

### Polityka Retencji Logów

| Typ Logu | Cel | Podstawa Prawna | Okres Retencji | Uzasadnienie |
| --- | --- | --- | --- | --- |
| **Logi zapytań i interakcji użytkownika** | Analiza wydajności agenta AI, debugowanie problemów użytkowników. | Uzasadniony interes. | 90 dni | Wystarczający okres do analizy ostatnich trendów i rozwiązywania problemów bez przechowywania danych w nieskończoność.44 |
| **Logi wykonania narzędzi i wywołań API** | Monitorowanie bezpieczeństwa, analiza wydajności. | Uzasadniony interes. | 30 dni | Krótszy okres ze względu na większą objętość tych logów i ich głównie operacyjny charakter.44 |
| **Logi błędów i systemowe** | Debugowanie, reagowanie na incydenty. | Uzasadniony interes. | 60 dni | Równowaga między potrzebą badania przeszłych incydentów a minimalizacją danych. |
| **Logi bezpieczeństwa i audytu (np. próby autoryzacji)** | Badanie incydentów bezpieczeństwa. | Obowiązek prawny / Uzasadniony interes. | 180 dni | Dłuższa retencja jest uzasadniona dla celów dochodzeń w sprawach bezpieczeństwa.45 |

## 8. Migracja/Fallback: Inteligentne Kierowanie Odpowiedzi

### 8.1. Streszczenie

Ta sekcja opisuje wzorzec inteligentnego kierowania (routingu) w celu generowania odpowiedzi, co zwiększa zarówno dokładność, jak i możliwości systemu. Główna ścieżka polega na odpytaniu bazy wiedzy Shopify (np. artykułów z centrum pomocy zaimportowanych do bazy wektorowej). Jeśli to wyszukiwanie zwróci wynik z wynikiem pewności (confidence score) poniżej zdefiniowanego progu, system automatycznie przejdzie do ścieżki zapasowej (fallback): wywołania niestandardowego narzędzia MCP (np. search\_shop\_catalog). Taki wzorzec to nie tylko strategia odporności na błędy, ale również strategia optymalizacji ekonomicznej. Wykorzystuje on tanią i szybką metodę (wyszukiwanie w bazie wiedzy) dla typowych pytań i eskaluje do droższej, bardziej zaawansowanej metody (wywołanie narzędzia) tylko wtedy, gdy jest to konieczne.46

### 8.2. Rekomendowany Minimalny Stack/Architektura

* **Baza Wiedzy:** Treści z natywnego Centrum Pomocy Shopify, zaimportowane do bazy danych wektorowej (np. Pinecone, lub Cloudflare Vectorize DB dla zintegrowanego stosu).
* **Ocena Pewności (Confidence Scoring):** Wykorzystanie prawdopodobieństw logarytmicznych (logprobs) z odpowiedzi LLM lub dedykowanej metryki oceny (np. ocena wierności - faithfulness score - w odniesieniu do pobranego kontekstu) do wygenerowania wyniku pewności.48
* **Logika:** Prosty blok if/else w głównej logice aplikacji, który sprawdza wynik pewności.
* **Mechanizm Fallback:** with\_fallbacks() z biblioteki LangChain stanowi czysty wzorzec do tego celu 50, ale można go również zaimplementować manualnie.

### 8.3. Przykładowy Minimalny Snippet/Polecenia

* **Pseudo-kod (Python):**  
  Python  
  CONFIDENCE\_THRESHOLD = 0.85 # Próg do dostrojenia  
    
  async def get\_answer(query: str) -> dict:  
   """  
   Pobiera odpowiedź, najpierw próbując z bazy wiedzy,  
   a następnie przechodząc do narzędzia MCP, jeśli pewność jest niska.  
   """  
   kb\_response, confidence\_score = await search\_knowledge\_base(query)  
    
   if confidence\_score >= CONFIDENCE\_THRESHOLD:  
   logger.info(f"Odpowiedź z bazy wiedzy z pewnością {confidence\_score:.2f}")  
   return {  
   "answer": kb\_response,  
   "source": "KNOWLEDGE\_BASE",  
   "confidence": confidence\_score  
   }  
   else:  
   logger.warning(  
   f"Niska pewność z bazy wiedzy ({confidence\_score:.2f}). Przejście do narzędzia MCP."  
   )  
   # Logika decydująca, które narzędzie wywołać na podstawie zapytania  
   tool\_response = await invoke\_mcp\_tool\_based\_on\_query(query)  
   return {  
   "answer": tool\_response,  
   "source": "CUSTOM\_TOOL",  
   "confidence": None # Pewność z narzędzia może być mierzona inaczej  
   }  
  + **Uzasadnienie:** Ten wzorzec fallbacku oparty na warunku (a nie tylko na błędzie API) jest zaawansowanym zastosowaniem strategii odporności.51 Użycie wyników pewności jako wyzwalacza dla alternatywnych ścieżek jest kluczową najlepszą praktyką.48

### 8.4. Bezpieczeństwo i Secrets Handling

Dostęp do bazy danych wektorowej oraz do LLM używanego do oceny pewności powinien być zabezpieczony kluczami API, zarządzanymi zgodnie ze strategią zarządzania sekretami opisaną w Sekcji 2.

### 8.5. Estymacja Pracy dla MVP w Dniach Roboczych

* **Całkowity czas:** 4–5 dni roboczych.
* **Podział prac:**
  + Konfiguracja podstawowej bazy wektorowej i potoku importu artykułów z bazy wiedzy: 2 dni.
  + Implementacja funkcji wyszukiwania w bazie wiedzy: 1 dzień.
  + Implementacja oceny pewności i logiki fallback: 1–2 dni.

### 8.6. Główne Ryzyka i Mitigacje

* **Ryzyko:** Zdefiniowanie odpowiedniego progu CONFIDENCE\_THRESHOLD jest trudne i może prowadzić albo do zbyt wielu niepotrzebnych fallbacków (jeśli jest za wysoki), albo do udzielania odpowiedzi niskiej jakości z bazy wiedzy (jeśli jest za niski).
  + **Mitygacja:** To nie jest wartość typu "ustaw i zapomnij". Należy zacząć od konserwatywnego progu (np. 0.85). Stworzyć "złoty zestaw" zapytań testowych i ocenić wydajność wyszukiwania w bazie wiedzy oraz logiki fallback na ich podstawie. Użyć telemetrii (metryka fallback\_triggered z Sekcji 5) do monitorowania wskaźnika fallbacków w produkcji i odpowiednio dostosowywać próg.
* **Ryzyko:** Logika fallback dodaje złożoność i opóźnienie do ścieżki odpowiedzi dla niektórych zapytań.
  + **Mitygacja:** Główna ścieżka (wyszukiwanie w bazie wiedzy) powinna być wysoce zoptymalizowana pod kątem szybkości. Całkowite opóźnienie pozostanie akceptowalne, ponieważ ścieżka fallback jest wybierana tylko dla mniejszości złożonych zapytań. Należy monitorować opóźnienie end-to-end dla obu ścieżek.

## 9. Priorytetyzowany Plan Wdrożenia MVP

### Streszczenie

Poniższy plan działania określa krytyczną ścieżkę do dostarczenia funkcjonalnego produktu minimalnego (Minimum Viable Product - MVP). Kroki są ułożone w sekwencji, która pozwala najpierw zbudować fundamentalne komponenty, zapewniając bezpieczny i testowalny system na każdym etapie.

### Priorytetyzowana Lista Kroków

1. **Krok 1: Bezpieczne Zarządzanie Tokenem i Narzędzie Admin API (3–4 dni)**
   * **Uzasadnienie:** Bezpieczeństwo jest najważniejsze. Pierwszym krokiem musi być ustanowienie bezpiecznego zarządzania tokenem Admin API Shopify przy użyciu menedżera sekretów. Zaimplementowanie narzędzia get\_order\_status zweryfikuje to bezpieczne połączenie end-to-end.
   * **Zadania:** Ukończenie wszystkich zadań z Sekcji 2. Implementacja narzędzia get\_order\_status z Sekcji 4.
2. **Krok 2: Prototyp Serwera MCP i Konfiguracja Hostingu (3–4 dni)**
   * **Uzasadnienie:** Po ustanowieniu bezpiecznego połączenia z backendem, następnym krokiem jest zbudowanie rdzenia serwera MCP i wdrożenie go w docelowym środowisku hostingowym (rekomendowane Cloudflare Workers).
   * **Zadania:** Ukończenie zadań z Sekcji 1 (prototyp FastMCP) i Sekcji 3 (konfiguracja hostingu). Integracja bezpiecznego zarządzania tokenem z Kroku 1.
3. **Krok 3: Implementacja Podstawowych Narzędzi Storefront (2–3 dni)**
   * **Uzasadnienie:** Rozbudowa głównych, skierowanych do użytkownika, możliwości agenta poprzez implementację publicznie dostępnych narzędzi do przeszukiwania katalogu i aktualizacji koszyka.
   * **Zadania:** Implementacja narzędzi search\_shop\_catalog i update\_cart z Sekcji 4.
4. **Krok 4: Podstawowa Telemetria i Rate Limiting (2–3 dni)**
   * **Uzasadnienie:** Przed jakimikolwiek testami z użytkownikami należy wdrożyć fundamentalną obserwowalność i ochronę. Zapewni to możliwość monitorowania zachowania i zapobiegania nadużyciom od samego początku.
   * **Zadania:** Implementacja schematu logowania ustrukturyzowanego z Sekcji 5 i konfiguracja ograniczania żądań na poziomie platformy z Sekcji 6.

### Całkowita Szacowana Pracochłonność MVP

Przybliżony całkowity czas potrzebny na realizację MVP wynosi **11–14 dni roboczych**. W rezultacie powstanie funkcjonalny, bezpieczny i obserwowalny serwer MCP, gotowy do integracji z klientem front-endowym i zdalnym modelem LLM.

#### Cytowane prace

1. A Beginner's Guide to Use FastMCP - Apidog, otwierano: września 23, 2025, <https://apidog.com/blog/fastmcp/>
2. How to Create an MCP Server in Python - FastMCP, otwierano: września 23, 2025, <https://gofastmcp.com/tutorials/create-mcp-server>
3. node-red-contrib-jsonrpc, otwierano: września 23, 2025, <https://flows.nodered.org/node/node-red-contrib-jsonrpc>
4. njs-jsonrpc-2.0 - Yarn Classic, otwierano: września 23, 2025, <https://classic.yarnpkg.com/en/package/njs-jsonrpc-2.0>
5. tedeh/jayson: Jayson is a simple but featureful JSON-RPC 2.0/1.0 client and server for node.js - GitHub, otwierano: września 23, 2025, <https://github.com/tedeh/jayson>
6. jlowin/fastmcp: The fast, Pythonic way to build MCP servers ... - GitHub, otwierano: września 23, 2025, <https://github.com/jlowin/fastmcp>
7. How to Build MCP Servers in Python: Complete FastMCP Tutorial for AI Developers, otwierano: września 23, 2025, <https://www.firecrawl.dev/blog/fastmcp-tutorial-building-mcp-servers-python>
8. MCP Shopify | MCP Servers - LobeHub, otwierano: września 23, 2025, <https://lobehub.com/mcp/antoineschaller-shopify-mcp-server>
9. Best Practices for API Key Safety | OpenAI Help Center, otwierano: września 23, 2025, <https://help.openai.com/en/articles/5112595-best-practices-for-api-key-safety>
10. Generate access tokens for custom apps in the Shopify admin, otwierano: września 23, 2025, <https://shopify.dev/docs/apps/build/authentication-authorization/access-tokens/generate-app-access-tokens-admin>
11. Secrets Management for AWS Powered Serverless Applications, otwierano: września 23, 2025, <https://www.serverless.com/blog/aws-secrets-management>
12. Custom apps - Shopify Help Center, otwierano: września 23, 2025, <https://help.shopify.com/en/manual/apps/app-types/custom-apps>
13. What are Durable Objects? · Cloudflare Durable Objects docs, otwierano: września 23, 2025, <https://developers.cloudflare.com/durable-objects/concepts/what-are-durable-objects/>
14. Monitoring latency: Cloudflare Workers vs Fly vs Koyeb vs Railway ..., otwierano: września 23, 2025, <https://www.openstatus.dev/blog/monitoring-latency-cf-workers-fly-koyeb-raylway-render>
15. Debugging WebSocket Hibernation with Cloudflare Durable Objects (and What I Learned), otwierano: września 23, 2025, <https://thomasgauvin.com/writing/how-cloudflare-durable-objects-websocket-hibernation-works/>
16. Monitoring latency: Cloudflare Workers vs Fly vs Koyeb vs Railway vs Render, otwierano: września 23, 2025, <https://news.ycombinator.com/item?id=39434271>
17. ️ Elegant Server-Sent Events (SSE) Streaming for Cloudflare Workers - GitHub, otwierano: września 23, 2025, <https://github.com/d521bb85/cloudflare-workers-sse>
18. Workers Server-Sent events (server push) - Cloudflare Community, otwierano: września 23, 2025, <https://community.cloudflare.com/t/workers-server-sent-events-server-push/184219>
19. Server Sent Events and Cloudflare - Render, otwierano: września 23, 2025, <https://community.render.com/t/server-sent-events-and-cloudflare/3238>
20. Shopify API access scopes, otwierano: września 23, 2025, <https://shopify.dev/docs/api/usage/access-scopes>
21. Introduction to Shopify's Storefront GraphQL - F22 Labs, otwierano: września 23, 2025, <https://www.f22labs.com/blogs/shopify-storefront-graphql/>
22. products - Storefront API - Shopify.dev, otwierano: września 23, 2025, <https://shopify.dev/docs/api/storefront/latest/queries/products>
23. Shopify's GraphQL API | Apply filter & Add metafields for product - Oscprofessionals, otwierano: września 23, 2025, <https://www.oscprofessionals.com/shopify-app/how-to-filter-products-using-shopifys-graphql-api-and-metafields/>
24. graphql and shopify Field 'cartCreate' doesn't exist on type 'Mutation ..., otwierano: września 23, 2025, <https://community.shopify.com/t/graphql-and-shopify-field-cartcreate-doesnt-exist-on-type-mutation/274970>
25. orders - GraphQL Admin - Shopify.dev, otwierano: września 23, 2025, <https://shopify.dev/docs/api/admin-graphql/latest/queries/orders>
26. Order - GraphQL Admin - Shopify.dev, otwierano: września 23, 2025, <https://shopify.dev/docs/api/admin-graphql/latest/objects/Order>
27. How to get all statuses of a specific order with GraphQL? - Shopify Community, otwierano: września 23, 2025, <https://community.shopify.com/c/graphql-basics-and/how-to-get-all-statuses-of-a-specific-order-with-graphql/td-p/1891904>
28. REST Admin API rate limits - Shopify.dev, otwierano: września 23, 2025, <https://shopify.dev/docs/api/admin-rest/usage/rate-limits>
29. Shopify API limits, otwierano: września 23, 2025, <https://shopify.dev/docs/api/usage/limits>
30. Agent Factory: Top 5 agent observability best practices for reliable AI | Microsoft Azure Blog, otwierano: września 23, 2025, <https://azure.microsoft.com/en-us/blog/agent-factory-top-5-agent-observability-best-practices-for-reliable-ai/>
31. Why observability is essential for AI agents - IBM, otwierano: września 23, 2025, <https://www.ibm.com/think/insights/ai-agent-observability>
32. What is rate limiting? | Rate limiting and bots - Cloudflare, otwierano: września 23, 2025, <https://www.cloudflare.com/learning/bots/what-is-rate-limiting/>
33. Rate limiting rules · Cloudflare Web Application Firewall (WAF) docs, otwierano: września 23, 2025, <https://developers.cloudflare.com/waf/rate-limiting-rules/>
34. Securing APIs: Express rate limit and slow down - MDN, otwierano: września 23, 2025, <https://developer.mozilla.org/en-US/blog/securing-apis-express-rate-limit-and-slow-down/>
35. Rate limiting middleware in ASP.NET Core - Microsoft Learn, otwierano: września 23, 2025, <https://learn.microsoft.com/en-us/aspnet/core/performance/rate-limit?view=aspnetcore-9.0>
36. Detecting PII with Python. Horror stories of dumping PII into… | by ..., otwierano: września 23, 2025, <https://mostafa.dev/detecting-pii-with-python-4c0da7b3b5bd>
37. Data Subject Access Requests (DSARs) - 5 Essential Steps - DPO Centre, otwierano: września 23, 2025, <https://www.dpocentre.com/data-subject-access-requests-5-essential-steps/>
38. sanityze - PyPI, otwierano: września 23, 2025, <https://pypi.org/project/sanityze/>
39. SOTA PII Redaction on Your Laptop - OpenPipe, otwierano: września 23, 2025, <https://openpipe.ai/blog/pii-redact>
40. What is Data Subject Access Request (DSAR)? - CookieYes, otwierano: września 23, 2025, <https://www.cookieyes.com/blog/dsar-data-subject-access-request/>
41. Subject access request Q and As for employers | ICO - Information Commissioner's Office, otwierano: września 23, 2025, <https://ico.org.uk/for-organisations/uk-gdpr-guidance-and-resources/employment/subject-access-request-q-and-as-for-employers/>
42. Collecting and using log data | NCSC-FI - Kyberturvallisuuskeskus, otwierano: września 23, 2025, <https://www.kyberturvallisuuskeskus.fi/en/ncsc-news/instructions-and-guides/collecting-and-using-log-data>
43. GDPR Log Management: A Practical Guide for Engineers - Last9, otwierano: września 23, 2025, <https://last9.io/blog/gdpr-log-management/>
44. What is log retention? Overview and best practices - LogicMonitor, otwierano: września 23, 2025, <https://www.logicmonitor.com/blog/what-is-log-retention>
45. How to design a reliable fallback system for LLM apps using an AI gateway - Portkey, otwierano: września 23, 2025, <https://portkey.ai/blog/how-to-design-a-reliable-fallback-system-for-llm-apps-using-an-ai-gateway>
46. LLM Router: Best strategies to route failed LLM requests - Vellum AI, otwierano: września 23, 2025, <https://www.vellum.ai/blog/what-to-do-when-an-llm-request-fails>
47. LLM Applications with Confidence Scoring: Know what you are evaluating | by Teckchuan, otwierano: września 23, 2025, <https://medium.com/@teckchuan/llm-applications-with-confidence-scoring-know-what-you-are-evaluating-cf1d58c0c899>
48. Confidence Scores in LLMs: Ensure 100% Accuracy in Large Language Models - Infrrd, otwierano: września 23, 2025, <https://www.infrrd.ai/blog/confidence-scores-in-llms>
49. How to add fallbacks to a runnable | 🦜️ LangChain, otwierano: września 23, 2025, <https://python.langchain.com/docs/how_to/fallbacks/>
50. How to add LLM Fallback to your LangChain Application - DigitalOcean, otwierano: września 23, 2025, <https://www.digitalocean.com/community/tutorials/langchain-llm-fallback-gradient-ai>
51. Designing Resilient LLM Architectures: Disaster Recovery Strategies | by Frank Goortani, otwierano: września 23, 2025, <https://medium.com/@FrankGoortani/designing-resilient-llm-architectures-disaster-recovery-strategies-6ad2e2f65942>
52. AI Agent Tools : Tutorial & Examples - Patronus AI, otwierano: września 23, 2025, <https://www.patronus.ai/ai-agent-development/ai-agent-tools>