

WORDPRESS Z WŁASNĄ SZTUCZNĄ INTELIGENCJĄ

Wyszukiwarka, dokumenty i baza wiedzy



PROJEKT TWORZY CZŁOWIEK.
AI WSPIERA PROCES.

05

10 ROZDZIAŁÓW / JEDEN SYSTEM DEMONSTRACYJNY

Spis treści

Wprowadzenie i mapa projektu	3
01 Własna baza wiedzy zamiast obietnicy własnego modelu	4
Co oznacza „własna” sztuczna inteligencja · Przepływ pytania i granice zaufania · Kiedy baza wiedzy jest dobrym wyborem	
02 Źródła, prywatność i kontrakt dokumentu	7
Biblioteka publiczna i źródła wewnętrzne · Metadane, które przeżyją zmianę pliku · Pełny snapshot oraz odpowiedzialny import	
03 Wydobywanie tekstu i przygotowanie fragmentów	10
TXT: najprostsza ścieżka do poprawnego tekstu · DOCX i PDF: parser nie jest OCR · Fragmenty i overlap bez utraty tożsamości	
04 Embeddingi i indeks w praktyce laboratoryjnej	13
Wektor nie jest streszczeniem ani dowodem · SQLite jako mała, czytelna baza odniesienia · Pierwszy import i pytanie w CLI	
05 Wyszukiwanie z uprawnieniami przed rankingiem	16
ACL musi ograniczać dane przed rankingiem · Grupy użytkownika pochodzą z serwera · Ranking i wynik podobieństwa	
06 Odpowiedź, cytowania i odmowa	19
Odpowiedź z kontekstu, nie z pamięci aplikacji · Cytowania weryfikowane wobec odzyskanych ID · Odmowa i prompt injection	
07 Prywatny gateway WordPress	22
Endpoint ask i zgoda na odczyt · Stały upstream i sekret tylko na serwerze · Walidacja odpowiedzi i brak cache współdzielonego	
08 Interfejs pytań oraz integracja biblioteki	25
Formularz, który pokazuje wynik jako tekst · Eksport biblioteki i brak automatycznego pobierania · Uruchomienie całości na stagingu	
09 Aktualizacje, usuwanie i eksploatacja	28
Nowa treść unieważnia wcześniejsze fragmenty · Wycofanie, purge i retencja · Koszty oraz obserwacja bez logowania treści	
10 Testy odbioru i świadoma rozbudowa	31
Opcjonalny adapter rzeczywistego modelu · Testy, które sprawdzają właściwe granice · Projekt przyjęty oraz kolejny bezpieczny etap	
Źródła techniczne i dalsza lektura	34

Kliknij rozdział. Nazwa tomu w stopce prowadzi do spisu. Pełne pliki techniczne i instrukcje odtworzenia są w EXAMPLES.

Odpowiedź zaczyna się od właściwego dokumentu

Zbudujesz mały, odtwarzalny system pytań do dokumentów. Najpierw ustalisz pochodzenie i dostęp, później oceniasz użyteczność odpowiedzi.

Wyobraź sobie redaktora WP DEMO LAB, który szuka procedury publikacji materiału. W zwykłej wyszukiwarce otrzymuje listę stron. W systemie opartym na dokumentach chce zobaczyć odpowiedni fragment, wersję i wskazanie miejsca, z którego pochodzi informacja. Takie rozwiązanie może skrócić drogę do źródła, ale potrzebuje znacznie więcej niż pola pytania i efektownego tekstu.

Tom rozwija publiczną bibliotekę materiałów z poprzedniego etapu. WordPress odpowiada za użytkownika, uprawnienia oraz interfejs. Lokalna usługa przechowuje indeks, wybiera dozwolone fragmenty i zwraca odpowiedź. Prywatne dokumenty demonstracyjne pozostają poza katalogiem uploads. Zwykły użytkownik nie może uzyskać dodatkowej grupy przez zmianę żądania w przeglądarce.

Kompletny przykład działa bez kluczy zewnętrznych. Jego reprezentacja tekstu to deterministyczny MOCK NONSEMANTIC, a odpowiedź jest wyciągiem znalezionych fragmentów. Nie demonstruje rozumienia języka, treningu ani jakości prawdziwej AI. Dzięki temu możesz sprawdzić przepływ, aktualizacje, cytowania i granice uprawnień bez kosztów dostawcy. Późniejsza integracja modelu jest odrębną decyzją projektową i testową.

Czytaj kolejne rozdziały z katalogiem EXAMPLES. Zawiera usługę Python, wtyczkę WordPress, klienta JavaScript, manifest, fikcyjne dokumenty oraz testy. Zapisz wynik każdego kroku. Python i składnię JavaScript sprawdzono tutaj rzeczywiście; aktywacja wtyczki, PHP oraz test użytkowników WordPress wymagają własnego stagingu. W odbiorze oddzielisz działający mechanizm od obietnicy, której ten pakiet jeszcze nie potwierdza.

Pracuj na kopii demonstracyjnej. Zachowaj oryginalny manifest, aby w każdej chwili odtworzyć import i porównać wynik z opisanym scenariuszem.

Co oznacza „własna” sztuczna inteligencja

Twoje dokumenty i reguły dostępu mogą być własne, nawet gdy model językowy pochodzi od zewnętrznego dostawcy.

Określenie „WordPress z własną sztuczną inteligencją” opisuje tutaj rozwiązanie pracujące na kontrolowanej bazie wiedzy. Nie oznacza wytrenowania modelu od początku ani zamiany każdego wpisu w dane treningowe. Najpierw wybierasz źródła pasujące do pytania, a dopiero później ewentualny model tworzy odpowiedź z otrzymanego kontekstu. To podejście nazywamy RAG.

Własność bazy oznacza odpowiedzialność za dokumenty: kto zatwierdził ich treść, kiedy zmieniła się wersja i kto może je przeczytać. Własność aplikacji oznacza kontrolę nad interfejsem, obsługą błędów i sposobem przekazywania danych. Wybór rzeczywistego modelu dodaje nowe zobowiązania: warunki przetwarzania, retencję, koszty i ocenę odpowiedzi.

W laboratorium celowo zaczynamy od rozwiązania offline. MOCK NONSEMANTIC koduje wystąpienia słów w stałym wektorze, a generator kopiuje fragmenty. Słowo „AI” w nazwie tomu nie zmienia tej właściwości. Taki mock jest przydatny do sprawdzenia przewodów systemu: czy pytanie dociera, czy prywatny tekst zostaje odfiltrowany, czy cytowanie wskazuje aktualny fragment.

Zapisz jeden mierzalny przypadek użycia. Przykład: redaktor znajduje zatwierdzoną procedurę publikacji i sprawdza wersję dokumentu w mniej niż kilku przejściach. Nie zaczynaj od ambicji odpowiedzi na każde pytanie. Im węższy zakres, tym łatwiej przygotować poprawne źródła i rozpoznać sensowną odmowę.

PROMPT DO CODEXA / Codex: granice pierwszej wersji

Opisz wykonalny zakres małej bazy wiedzy WP DEMO LAB.

Cel: Opisz wykonalny zakres małej bazy wiedzy WP DEMO LAB.

Dane: Trzy publiczne dokumenty, jeden internal, WordPress jako właściciel uprawnień; brak zewnętrznych kluczy.

Wynik: Tabela celu, wejścia, wyniku i odpowiedzialnego właściciela; osobno ograniczenia mocka.

KONTROLA: Nie nazywaj wektorów semantycznymi. Sprawdź, czy wynik pozwala wskazać źródło i odmówić przy braku materiału.

Przepływ pytania i granice zaufania

Najważniejsze rozdzielenie przebiega między przeglądarką, WordPressem oraz lokalną usługą indeksu.

Przeglądarka przesyła tylko pytanie. Cookie potwierdza sesję użytkownika w WordPress, a nagłówek X-WP-Nonce wspiera ochronę żądania przed CSRF w standardowym mechanizmie REST. Użytkownik nadal musi mieć capability `read_wdl_knowledge`. Sam nonce nie daje prawa czytania dokumentów i nie jest sekretem zastępującym hasło.

WordPress wylicza dozwolone grupy. Grupa `public` jest częścią naszego uprawnionego dostępu do wyszukiwarki. Grupa `internal` dochodzi wyłącznie po świadomym nadaniu dodatkowej capability. Gateway ignoruje pomysły przeglądarki dotyczące grup, a nieznane pola żądania odrzuca. Dzięki temu ta sama logika obowiązuje również użytkownika, który ręcznie modyfikuje request.

Usługa Python przyjmuje komunikację na loopback i wymaga wspólnego sekretu Bearer. To relacja serwer-serwer. Sekret nie trafia do skryptu, formularza ani HTML. Stały adres upstream usuwa potrzebę przyjmowania dowolnego URL od użytkownika. Wybrany URI jest elementem konfiguracji lokalnego profilu, nie częścią pytania.

Indeks ogranicza kandydatów do aktywnych dokumentów i dozwolonych grup jeszcze w SQL. Dopiero później liczy ranking. Wybrane fragmenty otrzymują lokatory i identyfikatory, które służą do kontroli cytowania. Odpowiedź wraca przez WordPress, dzięki czemu przeglądarka nigdy nie musi kontaktować się z prywatną usługą bezpośrednio.

Droga pytania

01 Przeglądarka

question, cookie i nonce; bez grup i sekretu

02 WordPress

capability oraz grupy wyliczone na serwerze

03 Usługa lokalna

Bearer, walidacja, SQL z ACL i aktywnością

04 Odpowiedź

fragmenty oraz cytowania; private, no-store

Ten przepływ pomaga wskazać właściciela każdego błędu. Uprawnienie konta, zgodność indeksu i wiarygodność treści sprawdzasz jako odrębne warunki odbioru.

Kiedy baza wiedzy jest dobrym wyborem

Źródło musi dawać się odnaleźć i sprawdzić. Brak odpowiedzi jest częścią poprawnego zachowania aplikacji.

Dobrze sprawdzają się procedury, instrukcje redakcyjne, opisy produktów i zatwierdzone odpowiedzi na powtarzające się pytania. Każdy dokument powinien mieć stabilną tożsamość oraz osobę odpowiedzialną za aktualność. Jeśli cała wiedza istnieje wyłącznie w rozmowach, najpierw potrzebujesz procesu dokumentowania, a nie bardziej skomplikowanego modelu.

Rozróżnij wyszukiwanie i interpretację. Pytanie „gdzie opisano przegląd makiety?” może prowadzić do fragmentu ze Studio Fala. Pytanie „czy na pewno ten klient zatwierdził publikację?” wymaga aktualnego dowodu; przykładowa procedura tego nie potwierdza. System nie powinien wypełniać brakujących informacji płynną odpowiedzią.

Ustal zakres ryzyka. Instrukcja prowadzenia szkolenia jest innym zadaniem niż ocena dokumentu prawnego czy decyzja dotycząca płatności. W małym projekcie odpowiedź ma pomóc człowiekowi odnaleźć dokument. Jeżeli użytkownik zaczyna traktować ją jako ostateczną decyzję, interfejs oraz polityka odbioru muszą jasno wyznaczyć granicę.

Przygotuj zestaw kontrolny zanim zaindeksujesz setki plików. Zapisz pytanie, oczekiwany dokument, poprawny fragment i przypadek odmowy. Dodaj pytanie o informację istniejącą wyłącznie w źródle internal, skierowane przez użytkownika public-only. Ten przykład pozwala oceniać dostęp oddzielnie od trafności. Nie używaj samego podobieństwa cosine jako miary prawdy: wysoki wynik oznacza podobieństwo reprezentacji, a nie wiarygodność dokumentu.

Pytanie	Poprawny cel
Jak publikować materiał?	Wskazanie procedury oraz wersji
Czy brief jest zaakceptowany?	Odmowa bez aktualnego dowodu
Co mówi internal?	Odmowa dla public-only

Próbka pytań powinna zawierać także brakujące informacje. Dzięki temu sprawdzisz, czy aplikacja potrafi powstrzymać się przed dopowiadaniem niepotwierdzonych faktów.

Biblioteka publiczna i źródła wewnętrzne

Publiczna biblioteka z tomu 04 oraz prywatny indeks z tomu 05 mają różne granice przechowywania.

CPT `wdl_material` porządkuje materiały przeznaczone do publicznego udostępnienia. Tytuł i opis pozostają standardowymi polami WordPress, załącznik wskazuje `_wdl_file_id`, a numer wersji przechowuje `_wdl_doc_version`. Opublikowany rekord może stać się źródłem publicznego indeksu. Stan draft nie jest jednak osłoną dla pliku, który już znajduje się w uploads.

Bezpośredni URL publicznego załącznika działa niezależnie od widoczności rekordu opisującego materiał. Jeżeli umieścisz poufny PDF w uploads, dodanie ACL w wyszukiwarce nie cofnie tej ekspozycji. Ochrona odpowiedzi i ochrona oryginału to dwa odrębne obowiązki. W tym projekcie prywatne źródła mają osobny katalog poza webroot oraz osobny manifest.

Folder `fixtures/private` zawiera wyłącznie sztuczne dane, więc można go dostarczyć jako materiał szkoleniowy. Nie traktuj nazwy folderu jako mechanizmu ochrony na hostingu. Cały katalog `EXAMPLES` instalujesz selektywnie: wtyczkę do WordPress, usługę oraz dokumenty poza publicznym katalogiem serwera. Uprawnienia plików są częścią odbioru.

Zacznij od klasyfikacji dokumentów. Osoba odpowiedzialna określa grupę, pochodzenie, wersję i retencję. Publiczny opis może być widoczny szeroko, podczas gdy wewnętrzna instrukcja pozostaje ograniczona. Jeśli klasyfikacja jest niejasna, nie importuj źródła do publicznego indeksu. Nie pozwalaj modelowi ani automatycznemu wykrywaniu słów zastąpić świadomej decyzji właściciela danych.

PROMPT DO CODEXA / Codex: mapa danych i ekspozycji

Rozpisz przechowywanie publicznych i wewnętrznych dokumentów.

Cel: Rozpisz przechowywanie publicznych i wewnętrznych dokumentów.

Dane: `wdl_material`, publiczne uploads, osobny katalog `private` poza webroot oraz grupy `public/internal`.

Wynik: Tabela pliku, miejsca, właściciela, grupy i sposobu odwołania dostępu.

KONTROLA: Zaznacz, że draft nie chroni URL uploads. Sprawdź, czy żadna prywatna ścieżka nie staje się publicznym linkiem.

Metadane, które przeżyją zmianę pliku

Dokument potrzebuje stabilnego `doc_id`. Numer wersji i hash opisują aktualny stan, a nie nową tożsamość.

Manifest jest lokalnym JSON-em kontrolowanym przez operatora. Każdy rekord zawiera `doc_id`, `version`, `group`, `active`, `title` i `source_url`. Tekst możesz przekazać bezpośrednio w `text` albo wskazać lokalny `source_file`. Te warianty prowadzą do tej samej reprezentacji po ekstrakcji. Importer nie pobiera dokumentu przez adres źródła; `source_url` jest metadaniem do późniejszej identyfikacji.

Stabilny identyfikator, na przykład `wp-material-101`, nie zmienia się po poprawieniu instrukcji. Numer wersji wzrasta zgodnie z procesem redakcyjnym. Hash wyliczany z wydobytego tekstu wychwytuje zmianę nawet wtedy, gdy autor zapomniał podnieść wersję. W rezultacie poprzednie fragmenty zostają zastąpione, a stare identyfikatory cytowań tracą ważność.

Lokator wskazuje miejsce w tekście po ekstrakcji. W naszym pakiecie są to numery słów, nie numery strony PDF ani akapitu w edytorze Worda. Ta decyzja jest skromna, lecz uczciwa: parser nie zachowuje pełnego układu dokumentu. Kiedy użytkownik potrzebuje precyzyjnych stron, musisz dodać parser zachowujący mapę stron i przetestować jej zgodność.

Import sprawdza typy, dodatnią wersję, dozwoloną grupę oraz ograniczone rozmiary. W wydaniu edukacyjnym URL jest pusty albo należy do `example.test`. Lokalna ścieżka nie może wyjść ponad katalog manifestu, również po rozwiązaniu ścieżki. Te walidacje oddzielają kontrakt danych od przypadkowych cech pierwszych pięciu `fixture`.

Pole	Rola
<code>doc_id</code>	Stała tożsamość źródła
<code>version</code> + <code>hash</code>	Stan treści i unieważnienie fragmentów
<code>group</code> + <code>active</code>	Dostęp oraz wycofanie
<code>locator</code>	Miejsce w wydobytym tekście

Pełny snapshot oraz odpowiedzialny import

Importer przyjmuje kompletny stan bazy. Pominięty dokument zostaje zdezaktywowany, dlatego manifest musi być świadomie złożony.

Publiczny eksport WordPress dostarcza opublikowane rekordy wdl_material bez hasła. Wewnętrzne materiały pochodzą z osobnego manifestu poza uploads. Przed importem operator łączy oba zbiory w jeden pełny snapshot. Jeśli uruchomi tylko listę publiczną, obecne dokumenty internal nie będą dalej aktywne. To zamierzony kontrakt pełnej synchronizacji, który trzeba zrozumieć przed pierwszą aktualizacją.

Ustal etapy pracy: przygotowanie eksportu, dołączenie zatwierdzonych źródeł prywatnych, walidacja, ekstrakcja i transakcja. Awaria parsera nie powinna pozostawić połowy nowych dokumentów w bazie. W dostarczonej usłudze load_manifest przygotowuje cały zbiór przed zapisami. import_docs zapisuje zmiany w jednej transakcji; niepowodzenie przy wersji powoduje rollback.

Nie wprowadzaj źródła wyłącznie dlatego, że plik istnieje. Sprawdź, czy jest aktualny, czy zawiera dane osobowe, czy jego licencja pozwala na użycie oraz czy uprawnienia odpowiadają docelowym odbiorcom. W projekcie demonstracyjnym wszystko jest fikcyjne, ale ćwiczenie warto wykonywać z taką samą dyscypliną jak rzeczywistą synchronizację.

Zapisuj wynik importu bez treści poufnych. Wystarczy liczba dokumentów, identyfikator partii, model reprezentacji i informacja o sukcesie lub błędzie. Pełne pytania, fragmenty oraz sekrety nie są potrzebne do zwykłego dziennika technicznego. W praktycznym odbiorze sprawdzisz, że stare fragmenty znikają po zmianie hash, a dokument pominięty w kolejnym snapshot nie uczestniczy w żadnym rankingu.

Od eksportu do aktywnego indeksu

01 Eksport publiczny

publish wdl_material bez hasła; opis i wersja

02 Manifest prywatny

lokalne źródła oraz świadoma klasyfikacja

03 Walidacja i parser

pełna partia gotowa przed transakcją

04 Transakcja SQLite

nowy stan aktywny; pominięte źródła wycofane

TXT: najprostsza ścieżka do poprawnego tekstu

Plik tekstowy pozwala najpierw sprawdzić przepływ bez komplikacji układu i rozpoznawania znaków.

Pierwsze źródła laboratorium są plikami TXT w UTF-8. Pozwalają obejrzeć dokładnie tę treść, którą otrzymuje indeks. Nie potrzebujesz edytora dokumentów, zewnętrznego SDK ani OCR. Rozmiar źródła jest ograniczony, pusty aktywny dokument jest błędem, a nie powodem do cichego dodania bezużytecznego rekordu.

Ekstrakcja jest odrębnym etapem od fragmentacji. Nie usuwaj od razu wszystkiego, co wygląda jak dekoracja, ponieważ numery wersji, nazwy procedur i oznaczenia sekcji bywają istotne. Z drugiej strony nagłówki powtarzane na każdej stronie może zdominować wyszukiwanie. W większym projekcie normalizacja wymaga testów na reprezentatywnych dokumentach, a nie jednej globalnej reguły.

Przejrzyj trzy publiczne źródła: Atelier Północ, Studio Fala i Pracownia Punkt. Każde ma inne zadanie, dzięki czemu pytanie o brief różni się od pytania o publikację czy dostępność. Odrębny dokument Latarnia zawiera sztuczny wskaźnik LAMPION-47. Jest celowo dostępny tylko dla grupy internal i służy sprawdzeniu granicy uprawnień.

Zanim przejdziesz do PDF, zaindeksuj fixture i odpytaj CLI. Zwróć uwagę na język mocka: dopasowuje słowa, nie ich znaczenie. Odmiana wyrazu albo synonim mogą dać słabszy wynik. To nie błąd opisany jako „model się nie nauczył”; modelu semantycznego tutaj nie ma. Dobre wyniki na małej próbce nie uzasadniają obietnicy obsługi dowolnego języka ani formatu.

PROMPT DO CODEXA / Codex: audyt tekstu wejściowego

Przygotuj raport jakości pięciu fixture przed indeksowaniem.

Cel: Przygotuj raport jakości pięciu fixture przed indeksowaniem.

Dane: Lokalne TXT w UTF-8, doc_id, grupa, wersja oraz limit rozmiaru; teksty są fikcyjne.

Wynik: Lista błędów kodowania, pustych źródeł, duplikatów tożsamości i niejasnych lokatorów.

KONTROLA: Nie poprawiaj automatycznie klasyfikacji ACL. Wykaż, czy indeksowany tekst odpowiada rzeczywistemu plikowi.

DOCX i PDF: parser nie jest OCR

Format pliku wpływa na utratę informacji. Odbiór ekstrakcji powinien poprzedzać ocenę wyszukiwarki.

DOCX jest archiwum zawierającym XML. Minimalny parser czyta word/document.xml, ogranicza rozmiar rozpakowanego dokumentu i zbiera tekst akapitów. Nie zachowuje tabel jako tabel, komentarzy, nagłówków stron, przypisów ani układu wizualnego. Jeśli dokument opiera się na relacji między kolumnami, prosty tekst może zmienić sens. Taki materiał wymaga przeglądu po ekstrakcji.

PDF w tym pakiecie korzysta z opcjonalnego lokalnego programu pdftotext. Wywołanie ma stałe argumenty, ograniczony czas i nie używa powłoki do wykonania fragmentu tekstu. Parser czyta warstwę tekstową. Skan bez takiej warstwy nie otrzymuje automatycznie treści; pusty wynik jest błędem. OCR oraz ocena jakości są poza zakresem.

Test PDF generuje niewielki plik tymczasowy z tekstem i sprawdza jego odczyt. Nie potwierdza obsługi każdego fontu, zaszyfrowanego dokumentu ani wielokolumnowego raportu. Test DOCX sprawdza jeden prosty akapit. Dzięki temu wynik testowy pozostaje adekwatny do mechanizmu, który naprawdę wykonano.

Dla każdego nowego formatu porównaj oryginał i wydobyty tekst. Zapisz fragment kontrolny obejmujący listę, datę, nagłówek oraz ewentualną tabelę. Jeżeli zdanie traci warunek albo kolejność, wstrzymaj indeksowanie. Lepiej pokazać błąd importu niż pewnym tonem cytować źródło, którego parser nie zrozumiał strukturalnie. Mapa stron jest rozszerzeniem przyszłym; bieżący lokator opisuje położenie w tekście po ekstrakcji.

Format	Wykonane	Poza zakresem
TXT	Odczyt UTF-8 z limitem	Naprawa zepsutego kodowania
DOCX	Tekst akapitów XML	Pełny układ i znaczenie tabel
PDF	pdftotext, warstwa tekstowa	OCR skanów i mapa stron

Fragmenty i overlap bez utraty tożsamości

Okno tekstu powinno być wystarczające do oceny odpowiedzi, a jego granice muszą dawać się odtworzyć.

Duży dokument zwykle nie mieści się w sensownym kontekście pytania. Fragmentacja tworzy mniejsze jednostki wyszukiwania. W przykładzie okno ma sto słów, a overlap dwadzieścia. Kolejny fragment zaczyna się przed końcem poprzedniego, aby zmniejszyć ryzyko przecięcia ważnego zdania lub warunku. Są to parametry ćwiczenia, nie uniwersalna recepta.

Większy fragment dostarcza więcej kontekstu, lecz może rozmyć dopasowanie i zwiększyć koszt przyszłego modelu. Zbyt mały fragment często oddziela polecenie od zastrzeżenia. Overlap powoduje powtórzenia, więc dłuższy indeks i podobne wyniki są naturalnym skutkiem. Oceniaj parametry na konkretnych pytaniach oraz źródłach, nie według samej liczby fragmentów.

Chunk_id zawiera tożsamość dokumentu, wersję, skrót hash i numer okna. Lokator podaje zakres słów w tekście po ekstrakcji. Identyfikator nie jest obietnicą wiecznego adresu: po zmianie źródła stary fragment powinien stracić ważność. Cytowanie zawsze odnosi się do fragmentu z bieżącego odzyskanego kontekstu.

Dostarczony algorytm nie zna struktury nagłówków. W dalszej rozbudowie możesz zastosować cięcie według sekcji i zdań, zachowując wspólny kontrakt identyfikatorów. Najpierw dodaj przypadek dokumentu, w którym ważny warunek leży na granicy okna. Porównaj zachowanie kilku ustawień, zapisując trafność i ilość powtórnego tekstu. Nie myl optymalizacji jakości z dowodem ochrony prywatności: ACL musi działać niezależnie od długości fragmentu.

Tożsamość fragmentu

01 Tekst po ekstrakcji

sprawdzony UTF-8 i lokalne źródło

02 Okna 100/20

długość sto słów; overlap dwadzieścia

03 Chunk_id i lokator

doc_id, wersja, hash i zakres słów

04 Indeks

wektor plus model i wymiary każdego fragmentu

Wektor nie jest streszczeniem ani dowodem

Embedding to reprezentacja przeznaczona do porównywania. Jego wynik zależy od konkretnego modelu i wymiarów.

Rzeczywisty embedding zamienia tekst na wektor liczb tak, aby użyteczne podobieństwa były możliwe do wyszukania. Model, jego wersja i liczba wymiarów należą do kontraktu indeksu. Nie wolno porównywać wektora pytania z indeksem utworzonym przez inny model tylko dlatego, że obie reprezentacje wyglądają jak listy liczb.

Nasze laboratorium używa mocka. Funkcja embed rozpoznaje słowa, oblicza ich hash i zlicza wystąpienia w 256 koszykach, po czym normalizuje długość. Nazwa MOCK-NONSEMANTIC-hashwords-v1 wyraźnie identyfikuje tę metodę. Stabilny hash daje odtwarzalność między uruchomieniami, lecz nie daje rozumienia synonimów, odmiany czy negacji.

Kolizje hash mogą zwiększać wynik między niezwiązanymi tekstami. Dlatego mock dodaje prosty warunek wspólnego słowa, zanim zaakceptuje kandydata. Nadal jest to dopasowanie leksykalne. Odpowiedź o procedurze może wyglądać przydatnie, ponieważ źródło zawiera to samo słowo, a nie dlatego, że aplikacja zrozumiała pytanie.

Zapisuj model i dimensions przy każdym fragmencie. Wyszukiwarka filtruje te wartości przed oceną. Gdy zechcesz użyć dostawcy semantycznego, przebudujesz cały indeks oraz zmienisz testy jakości. Nie przepinaj jedynie funkcji zapytania: dwa przestrzennie niezgodne zbiory wektorów nie stają się kompatybilne przez wspólny typ danych. Rzeczywisty adapter HTTP jest opcjonalny; jego protokół sprawdzono lokalnie bez wywołań dostawcy. Jakość wymaga osobnej oceny.

PROMPT DO CODEXA / Codex: zgodność przestrzeni wektorów

Sprawdź kontrakt modelu i wymiarów bez uruchamiania zewnętrznego API.

Cel: Sprawdź kontrakt modelu i wymiarów bez uruchamiania zewnętrznego API.

Dane: MOCK-NONSEMANTIC-hashwords-v1, dimensions=256, rekord chunk z modelem oraz normalizacja.

Wynik: Lista warunków reindeksacji i testy błędnego wymiaru lub modelu.

KONTROLA: Nie nazywaj hash semantyką. Wykaż, że zmiana dostawcy wymaga odtworzenia indeksu oraz osobnego pomiaru jakości.

SQLite jako mała, czytelna baza odniesienia

Prosty indeks pozwala zobaczyć każdą regułę. Jego łatwość kontroli jest ważniejsza niż obietnica wielkiej skali.

Tabela docs przechowuje tożsamość, wersję, grupę, tytuł, adres źródła, hash i aktywność. Tabela chunks zawiera tekst, lokator, wektor, model i wymiary. Klucz obcy wiąże fragment z dokumentem. Nie powielamy logiki uprawnień w odpowiedzi generatora: dostęp wynika z grupy dokumentu już podczas pobrania kandydatów.

Wektory są zapisane jako JSON, a ranking wykonuje Python po pobraniu dozwolonych wierszy. To świadomie mały profil. Pełny skan będzie kosztowny przy dużej liczbie fragmentów; przykład nie jest pomiarem wydajności ani konkurencją dla specjalistycznego silnika wektorowego. Przy późniejszej migracji zachowaj wymóg filtrowania ACL przed przekazaniem kontekstu.

Aktualizacja jest transakcyjna. Najpierw przygotowujesz dokumenty, potem zastępujesz fragmenty i metadane jako jedną jednostkę. Jeśli numer wersji cofa się, operacja nie kończy się częściowym stanem. Zmiana treści generuje nowy hash i nowe identyfikatory nawet przy tym samym numerze. Zapis poprzedniego źródła nie powinien nadal uczestniczyć w odpowiedzi.

Używaj SQL z parametrami. Jedynym dynamicznym elementem zapytania ACL jest liczba znaczników zapytania, wyznaczona na podstawie zwalidowanej listy grup. Nazwa grupy nie staje się kawałkiem składni SQL. Ta reguła działa również w małym pliku SQLite; rozmiar projektu nie zmniejsza konsekwencji błędów dostępu ani interpolowania wejścia.

Tabela	Minimalna odpowiedzialność
docs	ACL, aktywność, stan źródła i hash
chunks	Treść, lokator i zgodna reprezentacja
Transakcja	Spójne zastąpienie lub rollback
Ranking Python	Tylko kandydaci dopuszczeni przez SQL

Pierwszy import i pytanie w CLI

CLI jest narzędziem operatora. Nie przenosi automatycznie uprawnień użytkownika przeglądarki.

Uruchom polecenie index z manifestem fixture oraz lokalną ścieżką bazy poza publicznym katalogiem serwera. Otrzymasz liczbę dokumentów i nazwę modelu mocka. Pięć źródeł obejmuje trzy publiczne materiały, procedurę internal oraz testową instrukcję ataku. Nie ma tutaj pobierania treści z internetu ani wywołania modelu dostawcy.

Następnie użyj query z pytaniem „publikacja” i grupą public. Odpowiedź pokaże znalezione fragmenty, cytowania i tryb MOCK. Zapytanie o LAMPION ograniczone do public powinno zakończyć się odmową. Jako operator możesz świadomie podać public,internal, ponieważ CLI zakłada dostęp do lokalnej bazy. Nie używaj tego interfejsu jako autoryzacji publicznego formularza.

Otwórz SQLite i sprawdź relację między dokumentem oraz fragmentem. Po małej zmianie tekstu wykonaj ponowny import. Hash i chunk_id powinny się zmienić. To ważniejszy dowód spójności niż fakt, że ekran pokazuje jakąkolwiek odpowiedź. Dodaj również próbę z brakującym źródłem w snapshot i sprawdź jego nieaktywność.

Komendy z README można powtórzyć bez WordPress. Są pierwszą warstwą kontroli, która izoluje import i indeks od integracji PHP. W kolejnym etapie ten sam mechanizm wywoła lokalny serwer HTTP, a WordPress przekaże grupy wyliczone z zalogowanego konta. Wynik CLI nie jest zaliczonym testem cookie, nonce ani konfiguracji cache na hostingu.

KOD W PRAKTYCE / BASH / Start lokalnego indeksu — komendy shell

```
python3 services/knowledge/rag.py index \
  --db /tmp/wdl-knowledge.sqlite3 \
  --manifest fixtures/manifest.json
python3 services/knowledge/rag.py query \
  --db /tmp/wdl-knowledge.sqlite3 \
  --question 'publikacja' --groups public
```

PLIK: README.md

Komendy uruchomione offline; baza demonstracyjna poza webroot.

ACL musi ograniczać dane przed rankingiem

Filtrowanie gotowej odpowiedzi jest zbyt późne. Prywatne źródło nie powinno dotrzeć nawet do kontekstu generatora.

Wyobraź sobie usługę, która najpierw wybiera najbardziej podobne dokumenty ze wszystkich grup, a dopiero później usuwa niedozwolone cytowania. Generator zdążył już przeczytać treść prywatną. Może ją parafrazować bez przypisu albo włączyć do odpowiedzi w inny sposób. Ochrona listy linków nie chroni informacji, która wcześniej weszła do kontekstu.

Nasza usługa pobiera fragmenty przez JOIN tabel chunks i docs. Warunek sprawdza `active=1`, dozwoloną grupę, aktualną wersję oraz zgodność modelu i wymiarów. Dopiero dozwolone wiersze podlegają `cosine`. To kolejność, którą należy utrzymać także po zmianie magazynu wektorowego. Model nie otrzymuje kompetencji przyznawania sobie szerszych praw.

W testach prywatne wektory zostają celowo uszkodzone. Zapytanie `public` nadal musi działać. Gdyby ranking najpierw czytał wszystkie wektory, nastąpiłby błąd parsowania. Ta próba daje konkretny dowód, że ACL ogranicza kandydatów przed ich oceną; dodatkowo kontrolujemy brak wskaźnika prywatnego w zbudowanym kontekście.

Usługa ufa grupom przekazanych przez uprawniony gateway z sekretem. Dlatego `loopback` i `Bearer` są częścią tej samej granicy. Każdy proces posiadający ten sekret może deklarować `internal`. Przy większej architekturze potrzebujesz dodatkowej tożsamości usług, segmentacji i kontroli operatorów. Nie wystawiaj lokalnego interfejsu do przeglądarki ani nie publikuj wspólnego sekretu w konfiguracji klienta.

ACL przed kontekstem

01 Grupy z WordPress

`public`; `internal` tylko z `explicit capability`

02 SQL

`active`, grupa, wersja, model i `dimensions`

03 Ranking

`cosine` wyłącznie dozwolonych fragmentów

04 Kontekst

lista odzyskanych ID bez prywatnego przecieku

WP DEMO LAB • TOM 05

Grupy użytkownika pochodzą z serwera

Pola formularza mogą zmieniać pytanie, lecz nie mogą rozszerzać dostępu do dokumentów.

Endpoint WordPress akceptuje tylko obiekt JSON z question. Dopisanie `allowed_groups` albo `groups` do browser request kończy się błędem. Nie stosujemy rozwiązania, w którym frontend przesyła `internal`, a backend jedynie sprawdza, czy tekst grupy jest dobrze zapisany. Poprawna nazwa grupy nie oznacza uprawnienia do jej użycia.

Podstawowa capability `read_wdl_knowledge` otwiera wyszukiwarkę dla zalogowanego użytkownika. `Internal` wymaga dodatkowego `read_wdl_internal_knowledge`, nadanego świadomie. Wtyczka nie nadaje tego prawa automatycznie administratorowi podczas aktywacji. Sprawdza też jawne `allcaps`, aby uprzywilejowanie superadministratora nie zastąpiło wymagania wyraźnego przypisania grupy w tym profilu.

Uprawnienia do publikowania materiałów i do czytania bazy wiedzy są odrębne. Redaktor biblioteki nie musi mieć dostępu do każdej wewnętrznej instrukcji. Użytkownik wyszukiwarki nie potrzebuje edycji treści, instalowania wtyczek ani administracji serwerem. Rozdzielenie tych praw ułatwia ograniczenie konta do jego rzeczywistego zadania.

Przygotuj dwa konta testowe z minimalnymi uprawnieniami. Jedno ma wyłącznie dostęp podstawowy, drugie również `explicit internal`. Wyślij identyczne pytanie o Latarnia i porównaj źródła. Następnie zmodyfikuj request konta `public-only`, dopisując grupę. Oczekiwany wynik to odrzucenie wejścia, nie ciche zwiększenie dostępu. Próba musi być wykonana w WordPress; test listy grup w lokalnym Python nie potwierdza mapowania capabilities w PHP.

PROMPT DO CODEXA / Codex: test macierzy uprawnień

Zaprojektuj test stagingu dla endpointu `ask`.

Cel: Zaprojektuj test stagingu dla endpointu `ask`.

Dane: Dwa konta, `read_wdl_knowledge`, `explicit read_wdl_internal_knowledge`, `cookie` i `X-WP-Nonce`.

Wynik: Macierz żądania, statusu i dozwolonych źródeł; także request z ręcznie dopisaną grupą.

KONTROLA: Nie traktuj `nonce` jako `capability`. Oddziel zaliczone testy Python od niewykonanej próby WordPress i superadministratora.

Ranking i wynik podobieństwa

Cosine porządkuje kandydatów. Nie poświadcza aktualności, prawdziwości ani kompletności dokumentu.

Po filtrze ACL wektor pytania jest porównywany z wektorami dozwolonych fragmentów. Wyniki są sortowane malejąco, a przy remisie pomaga stabilny `chunk_id`. W naszym profilu zwracamy najwyżej trzy fragmenty powyżej progu demonstracyjnego. Daje to ograniczony kontekst, który można łatwo obejrzeć ręcznie.

Wysoki wynik nie odpowiada na pytanie, czy procedura obowiązuje obecnie. Dokument może być przestarzały albo zawierać błąd autora. Ten problem rozwiązuje proces redakcyjny, wersjonowanie i wycofywanie, a nie matematyczny ranking. Również niski wynik nie dowodzi braku wiedzy: mock może nie dopasować odmiany lub synonimu.

Oceń ranking na parze pytanie–oczekiwany fragment. Zapisz, czy właściwy dokument znalazł się w pierwszej trójce oraz czy wybrane okno zawiera warunek potrzebny do odpowiedzi. Odrębnie sprawdź niepożądane źródła. Jeśli wyniki są podobne, większy kontekst może pomagać człowiekowi, lecz nie powinien automatycznie zwiększać liczby danych przesyłanych dostawcy.

Fragment poniżej pokazuje sedno zabezpieczenia. Dynamiczna liczba placeholderów wynika wyłącznie z długości zwalidowanej listy grup, a ich wartości trafiają jako parametry. Zgodność `embedding_space` jest sprawdzana wcześniej. Po zmianie mock/openai, modelu albo wymiarów usługi wymaga przebudowy indeksu. Błąd niezgodności nie powinien zostać przedstawiony użytkownikowi jako zwykła informacja „brak dokumentów”.

KOD W PRAKTYCE / PYTHON / Filtr SQL przed obliczeniem cosine

```
candidates = db.execute(f"""
    SELECT c.*, d.title, d.source_url, d.group_name
    FROM chunks c JOIN docs d ON c.doc_id=d.doc_id
    WHERE d.active=1 AND d.group_name IN ({placeholders})
    AND c.version=d.version AND c.model=? AND c.dimensions=?
""", (
    *allowed_groups, provider.model, provider.dimensions
)).fetchall()
```

PLIK: `services/knowledge/rag.py`

Fragment pełnego pliku EXAMPLES; Python/JS sprawdzone offline, PHP wymaga stagingu.

Odpowiedź z kontekstu, nie z pamięci aplikacji

Generator ma pracować na odzyskanych fragmentach. Aplikacja kontroluje źródła zarówno przed wywołaniem, jak i po nim.

build_context oddziela policy od untrusted_sources. Każde źródło posiada chunk_id, locator i tekst. Polecenia zapisane w dokumentach pozostają niezaufałą zawartością, nie regułami działania usługi. W rzeczywistym adapterze polityka trafia osobno, a pytanie oraz fragmenty są serializowane jako dane użytkownika. Nie dajemy modelowi narzędzi do wykonywania instrukcji ze źródeł.

Tryb domyślny nie tworzy syntezy językowej. Pokazuje ograniczone fragmenty z oznaczeniem MOCK i identyfikatorami. To jawna odpowiedź wyciągowa: użytkownik sam ocenia związek materiału z pytaniem. Jeśli w dokumencie testowym znajduje się zdanie „ignoruj politykę”, może pojawić się jako cytowany tekst; nie powoduje zmiany ACL, wykonania polecenia ani odczytu sekretu.

Opcjonalny tryb openai korzysta z adaptera HTTP embeddings oraz Responses. Wybrany model generowania i model embeddings ustawiasz w środowisku usługi. Wyjście ma strukturę answer, citations, refused. Schema ogranicza format, ale nie udowadnia poprawności treści. Dlatego serwis dodatkowo kontroluje cytowania wobec dokładnie tych fragmentów, które odzyskał.

Najważniejsza decyzja następuje przed generowaniem: pusty kontekst kończy się odmową. System nie powinien wtedy prosić modelu o odpowiedź „z ogólnej wiedzy”, jeśli obiecuje pracę na bazie dokumentów. Sytuacja bez źródeł ma prosty, testowalny wynik oraz pustą listę cytowań. Użytkownik może doprecyzować pytanie lub przekazać brakujący dokument do procesu redakcyjnego.

Etap	Kontrola
Przed generowaniem	ACL i pusty kontekst
W danych	Źródła oznaczone jako niezauwane
Po generowaniu	Format oraz allowlista cytowań
Przy odbiorze	Treść oceniona przez człowieka

Cytowania weryfikowane wobec odzyskanych ID

Cytowanie ma potwierdzać pochodzenie fragmentu. Nie jest automatycznym certyfikatem prawdziwości odpowiedzi.

Generator nie może dowolnie wymyślić `chunk_id`, a następnie otrzymać od aplikacji ładnego linku. `validate_citations` buduje allowlistę z rzeczywiście odzyskanych fragmentów. Sprawdza identyfikator oraz dokładny lokator. Dodatkowe metadane cytowania, takie jak tytuł, wersja i adres źródła, pochodzą z indeksu, nie z twórczości modelu.

Listę ograniczamy do trzech pozycji i odrzucamy duplikaty. Odpowiedź nieodmowna wymaga przynajmniej jednego cytowania, a odmowa ma ich zero. Wartości spoza kontekstu kończą się błędem przed wyświetleniem. To mechaniczny warunek, który możesz przetestować sztuczną odpowiedzią adaptera bez opłacania prawdziwego modelu.

Pozostaje problem zgodności wniosku ze źródłem. Model może podać poprawny identyfikator, a błędnie odczytać warunek lub datę. Allowlista tego nie wykrywa. Dlatego podczas oceny zapisz osobno: czy źródło istnieje, czy użytkownik ma do niego dostęp i czy rzeczywiście wspiera zdanie odpowiedzi. Nie łącz tych trzech kryteriów w jeden wynik „cytowanie poprawne”.

Lokator w naszym wydaniu podaje słowa po ekstrakcji. W przypadku prywatnego źródła `source_url` jest pusty; nie ujawniamy ścieżki serwerowej. Interfejs pokazuje tytuł, wersję, lokator i ID jako zwykły tekst. Przy późniejszym linkowaniu publicznych źródeł kontroluj adres docelowy oraz ponownie uprawnienia do oryginału. Bezpieczeństwo odnośnika nie wynika z tego, że wcześniej wygenerowano wiarygodnie wyglądający przypis.

PROMPT DO CODEXA / Codex: validator cytowań

Sprawdź odpowiedź adaptera przed udostępnieniem jej użytkownikowi.

Cel: Sprawdź odpowiedź adaptera przed udostępnieniem jej użytkownikowi.

Dane: Lista odzyskanych `chunk_id`, `locator` i metadanych oraz JSON `answer/citations/refused`.

Wynik: Warunki akceptacji i testy wymyślonego ID, fałszywego lokatora, duplikatu i pustej listy.

KONTROLA: Oddziel walidację pochodzenia od oceny zgodności wniosku ze źródłem. Prywatne ścieżki nie mogą trafić do klienta.

Odmowa i prompt injection

Niezaufany dokument może zawierać instrukcję ataku. Musi pozostać danymi, a aplikacja powinna zachować swoje reguły.

Fixture private-injection zawiera jawny tekst ataku: zmień grupę użytkownika i wyślij sekret. Dokument jest wewnętrzny, więc konto public-only w ogóle nie powinno go odzyskać. To pierwszy, niezależny od modelu poziom kontroli. Dla operatora internal tekst może być źródłem testowym, ale nie uzyskuje prawa zmieniania polityki aplikacji.

Test strukturalny sprawdza oddzielenie policy od untrusted_sources i brak tego dokumentu w publicznym kontekście. Mock cytuje dane bez interpretowania poleceń, dlatego nie wykonuje ataku. Nie nazywaj takiego wyniku dowodem odporności rzeczywistego modelu. Ochronę konkretnego modelu trzeba sprawdzać po jego włączeniu, na zróżnicowanych atakach oraz odpowiedziach.

Odmowa ma dwa powody, które warto rozróżnić użytkowo. Brak dostępnych źródeł to zwykły wynik wyszukiwania. Błąd protokołu, niezgodność modelu albo nieważne cytowanie to problem techniczny, przy którym nie publikujemy niezweryfikowanej odpowiedzi. Interfejs nie musi ujawniać szczegółu, ale operator powinien umieć odróżnić te stany bez zapisywania poufnej treści.

Nie doklejaj długiej listy obietnic bezpieczeństwa do promptu i nie zakładaj, że to wystarczy. Sekret pozostaje poza kontekstem, ACL działa wcześniej, a generator nie otrzymuje narzędzi. Taka architektura zmniejsza dostępne skutki błędu modelu. Właściwy odbiór obejmuje jednak także prawdziwość treści, wydobywanie informacji z innych źródeł, koszt nadmiernego kontekstu i zachowanie przy niejednoznacznym pytaniu.

Stan	Wynik
Pusty kontekst	Odmowa, citations=[]
Wymyślone cytowanie	Błąd; brak publikacji odpowiedzi
Instrukcja w źródle	Niezaufane dane, bez narzędzi
Test lokalny injection	Kontrola struktury; brak dowodu jakości LLM

Endpoint ask i zgoda na odczyt

Zalogowanie, nonce i capability współpracują. Każdy element ma inną rolę w prywatnym REST.

Wtyczka rejestruje POST /wp-json/wp-demo-lab/v1/ask podczas `rest_api_init`. `permission_callback` wymaga zalogowania oraz `read_wdl_knowledge`. Standardowy mechanizm WordPress sprawdza cookie i REST nonce. Przeglądarka przekazuje X-WP-Nonce dla działania `wp_rest`, ponieważ formularz działa wewnątrz sesji WordPress, a nie jako niezależny klient serwerowy.

Callback dodatkowo ogranicza body do 4096 bajtów i akceptuje tylko question. Tekst musi być poprawnym UTF-8 oraz zawierać od jednego do pięciuset znaków po usunięciu skrajnych spacji. Limity w JavaScript poprawiają doświadczenie użytkownika, lecz decyzja bezpieczeństwa pozostaje po stronie serwera. Nie polegaj wyłącznie na `maxlength` w polu.

Wartość `WDL_RAG_TOKEN` pochodzi ze środowiska procesu PHP. Jeśli konfiguracji brakuje, gateway zwraca kontrolowany błąd, zamiast wysłać żądanie bez zabezpieczenia. WordPress wylicza `public` i opcjonalne `internal`; JSON klienta nie jest źródłem tej decyzji. Aktywacja nadaje podstawową capability administratorowi, a prawa wewnętrzne trzeba przypisać świadomie.

Prawdziwy odbiór wymaga prób bez cookie, bez nonce, z wygasłą sesją i bez capability. Zobacz również nagłówki odpowiedzi przy błędach. Nonce nie jest jednorazowym biletem do dowolnego dokumentu ani substytutem autoryzacji. W tym środowisku nie uruchomiono PHP ani rdzenia WordPress, więc opis działania wtyczki jest instrukcją integracji, a nie raportem z już zaliczonego stagingu.

PROMPT DO CODEXA / Codex: review prywatnego endpointu

Przejrzyj gateway ask względem kontraktu WP DEMO LAB.

Cel: Przejrzyj gateway ask względem kontraktu WP DEMO LAB.

Dane: PHP 8.2+, cookie REST, X-WP-Nonce, dwie capabilities, question-only JSON i stały upstream.

Wynik: Lista konkretnych kontroli wejścia, uprawnień, sekretu i błędów z nazwą funkcji.

KONTROLA: Nie zgłaszaj PHP ani WordPress jako przetestowanych bez uruchomienia. Sprawdź błędy auth i Cache-Control.

WP DEMO LAB • TOM 05

Stały upstream i sekret tylko na serwerze

URI lokalnej usługi jest stałą kodu. Użytkownik nie wskazuje miejsca, z którego serwer ma pobierać dane.

Gateway wywołuje `http://127.0.0.1:8787/ask`. Nie przyjmuje pola URL ani parametru wybierającego dostawcę. Ta prosta decyzja usuwa z formularza całą klasę problemów związanych z pobieraniem dowolnego adresu przez serwer. `source_url` dokumentu również pozostaje metadany, a nie celem automatycznego fetch.

`wp_remote_post` otrzymuje ograniczony timeout, zakaz przekierowań, limit odpowiedzi oraz JSON body. Authorization zawiera Bearer ze środowiska. Cookie użytkownika nie jest przekazywane lokalnej usłudze; usługa otrzymuje tylko pytanie i grupy wyliczone przez gateway. Nie potrzebuje hasła konta WordPress ani znajomości jego roli.

Loopback zakłada, że PHP i Python działają w tej samej przestrzeni sieciowej. W dwóch kontenerach 127.0.0.1 odnosi się do dwóch różnych środowisk. Nie obchodzi tego przez wystawienie usługi na publiczny port bez kontroli. Zmiana na profil produkcyjny wymaga zaprojektowania prywatnej sieci, TLS i zasad dopuszczania ruchu, a potem oddzielnych testów.

Ustal rotację wspólnego sekretu. Oba procesy muszą otrzymać zgodną wartość, a zmiana powinna być przetestowana w oknie wdrożeniowym. Sekret dostawcy OpenAI jest czymś innym: należy tylko do usługi Python i nie zastępuje Bearer gateway. Żaden z tych sekretów nie trafia do odpowiedzi, logu pytań, pliku HTML ani repozytorium. Przykładowy env zawiera wyłącznie placeholder i opis konfiguracji.

Dwie relacje serwerowe

01 Przeglądarka → WordPress

sesja i nonce; pytanie bez sekretu

02 WordPress → loopback

stały URI i `WDL_RAG_TOKEN`

03 Python → opcjonalny dostawca

`OPENAI_API_KEY` tylko w trybie openai

04 Powrót przez gateway

walidowany JSON; bez headers i body błędu upstream

Walidacja odpowiedzi i brak cache współdzielonego

Prawidłowy HTTP status nie wystarcza. Gateway sprawdza strukturę danych przed zwróceniem odpowiedzi użytkownikowi.

`wp_remote_post` może zwrócić `WP_Error` albo odpowiedź o statusie innym niż 200. W obu przypadkach użytkownik dostaje krótki komunikat techniczny, bez oryginalnego body i nagłówków usługi. Sekret nie powinien pojawić się nawet wtedy, gdy upstream odpowie nieprzewidywalnym tekstem. Kontrolowana obsługa błędu jest częścią kontraktu, nie dodatkiem estetycznym.

Po sukcesie gateway dekoduje JSON i sprawdza `answer`, `mode`, `citations` oraz `refused`. Kontroluje typy, rozmiary i zgodność odmowy z pustymi cytowaniami. Odrzuca niepoprawne wersje oraz adresy poza domeną demonstracyjną. Serwis wcześniej sprawdził ID i lokatory wobec odzyskanych fragmentów; walidacja PHP stanowi drugi poziom kontroli formatu na granicy systemów.

Wszystkie odpowiedzi `ask` otrzymują `Cache-Control: private, no-store`. Filtr `rest_post_dispatch` obejmuje również błędy uwierzytelnienia i `permission`. Strona ze shortcode otrzymuje nagłówki wyłączające cache. Konfiguracja CDN i page cache musi mimo tego wykluczyć prywatną stronę oraz endpoint. Jeśli cache odpowie zanim uruchomi PHP, kod wtyczki nie naprawi takiej konfiguracji.

Przy odbiorze wykonaj dwa żądania z różnych kont, a potem wyloguj się i ponów próbę. Sprawdź body, nagłówki oraz zachowanie warstwy cache. Własna odpowiedź internal nie może zostać pokazana komuś innemu przez wspólny cache. Test lokalnego serwera potwierdza jego nagłówek, lecz nie dowodzi konfiguracji CDN ani ustawień hostingu WordPress.

KOD W PRAKTYCE / PHP / Jawne przypisanie grupy internal

```
$groups = array('public');
$user = wp_get_current_user();
if (!empty($user->allcaps['read_wdl_internal_knowledge'])
    && current_user_can('read_wdl_internal_knowledge')) {
    $groups[] = 'internal';
}
```

PLIK: `wp-content/plugins/wp-demo-knowledge/wp-demo-knowledge.php`

Fragment pełnego pliku EXAMPLES; Python/JS sprawdzone offline, PHP wymaga stagingu.

Formularz, który pokazuje wynik jako tekst

Źródła mogą zawierać HTML i instrukcje. Klient ma je wyświetlać jako treść, bez wykonywania.

Shortcode `wdl_knowledge` tworzy etykietowane pole pytania, przycisk, komunikat stanu oraz osobne miejsce na odpowiedź i listę cytowań. Użytkownik może wysłać pytanie dopiero po walidacji długości. Serwer nadal wykonuje tę samą kontrolę niezależnie od klienta. Formularz nie zawiera pola wyboru grupy ani konfiguracji upstream.

Fetch używa credentials: same-origin, przesyła JSON i nagłówek X-WP-Nonce. cache: no-store jest ustawieniem klienta, które nie zastępuje polityki odpowiedzi serwera. AbortController ogranicza czas oczekiwania interfejsu; gateway ma własny timeout. Po błędzie przycisk wraca do normalnego stanu, a komunikat nie ujawnia szczegółów infrastruktury.

Odpowiedź trafia do `textContent`. Cytowania powstają przez `createElement` i również korzystają z `textContent`. Nie używamy `innerHTML` do renderowania danych z dokumentu lub modelu. Dzięki temu fragment wyglądający jak znacznik nie staje się wykonywanym HTML. Jeśli w przyszłości dodasz format Markdown, potrzebujesz odrębnej decyzji i sprawdzonego mechanizmu sanitizacji.

Zadbaj o informację zwrotną. Pole ma label, a status `role=status` oraz `aria-live=polite`. Komunikat mocka mówi o fragmentach, wynik rzeczywistego adaptera zachęca do sprawdzenia źródeł. Nie oznacza to zakończonego audytu dostępności: należy jeszcze przetestować fokus, obsługę klawiatury, zawijanie długiej odpowiedzi i czytelność na telefonie. Składnia JavaScript została sprawdzona, ale prawdziwe zachowanie w WordPress wymaga uruchomienia strony.

KOD W PRAKTYCE / JAVASCRIPT / Renderowanie bez innerHTML

```
answer.textContent = value.answer;
for (const citation of value.citations) {
  const row = document.createElement('li');
  row.textContent = `${citation.title}; wersja `
    + `${citation.version}; ${citation.locator}; `
    + `${citation.chunk_id}`;
  citations.append(row);
}
```

PLIK: `wp-content/plugins/wp-demo-knowledge/assets/knowledge.js`

Fragment pełnego pliku EXAMPLES; Python/JS sprawdzone offline, PHP wymaga stagingu.

Eksport biblioteki i brak automatycznego pobierania

Publikacja rekordu oznacza dostępność opisu. Załącznik wymaga osobnego, kontrolowanego etapu parsowania.

Wtyczka dodaje polecenie WP-CLI `wdl knowledge-export`. Odczytuje wyłącznie opublikowane `wdl_material` bez hasła. WP_Query używa `has_password=false`, a dodatkowa kontrola `post_password` pomija chroniony rekord przed odczytem treści. Zachowuje stabilne ID i wersję, a opis zamienia na tekst. Wynik jest listą JSON zgodną z importerem. Pole `_wdl_file_id` jest metadaniem publicznego załącznika, nie instrukcją pobrania go z przypadkowego URL.

Eksport nie czyta automatycznie PDF ani DOCX wskazanego przez załącznik. Jeśli opis stanowi pełne źródło wiedzy, możesz użyć go bez dalszych kroków. Jeśli wiedza znajduje się w publicznym pliku, operator kopiuje plik do kontrolowanego katalogu manifestu i wskazuje `source_file`. Parser lokalny zajmuje się formatem, a `source_url` pomaga później zidentyfikować publiczny rekord.

Publiczną listę łączysz z osobnym manifestem `internal` przed uruchomieniem pełnego snapshot. Nie doklejaj prywatnego pliku do biblioteki tylko po to, aby uzyskać wygodny eksport. W tym etapie katalog poza `webroot` oraz osobna klasyfikacja są bardziej istotne niż jednolity ekran zarządzania każdym rodzajem dokumentu.

Dla samodzielnego startu pakiet zawiera `support/minimal-library.php`. Rejestruje CPT, taksonomię i wymagane meta. To minimalny kontrakt, nie pełne zarządzanie plikami z tomu 04. Wybierz jeden wariant biblioteki. Dwie aktywne implementacje tych samych funkcji mogą wprowadzić konflikty rejestracji oraz niejasne prawa. Czytaj README przed skopiowaniem plików i zapisz, którą konfigurację zainstalowano na stagingu.

Źródło	Droga do manifestu
Opis publish bez hasła	Eksport WP-CLI jako text
Publiczny PDF/DOCX	Lokalna kopia i <code>source_file</code>
Prywatna instrukcja	Osobny katalog i grupa <code>internal</code>
Draft	Pominięty w publicznym eksporcie

Na stagingu porównaj eksport publicznego rekordu bez hasła, `publish` z hasłem, `draft` i `private`. Chroniony opis oraz jego marker testowy mają być nieobecne. To próba ręczna, której tutaj nie wykonano.

Uruchomienie całości na stagingu

Integracja jest ukończona dopiero wtedy, gdy właściwe konto otrzymuje właściwe źródła przez prawdziwy WordPress.

Najpierw uruchom usługę offline, indeksując fixture do bazy poza webroot. Wygeneruj lokalny sekret bez wypisywania go do dziennika i ustaw tę samą wartość dla procesów Python oraz PHP. Wtyczkę skopiuj do wp-content/plugins/wp-demo-knowledge, a resztę przykładu pozostaw poza publicznym katalogiem. Nie publikuj całego EXAMPLES jako statycznej witryny.

Aktywuj gateway oraz wybrany wariant biblioteki. Utwórz stronę ze shortcode, trzy publiczne rekordy demonstracyjne i dwa konta do macierzy dostępu. Pierwsze pytanie może dotyczyć publikacji materiału. Kolejne powinno sprawdzać LAMPION i widoczność procedury Latarnia. Oczekiwany wynik zależy od jawnej capability, nie od tytułu roli użytkownika.

Zatrzymaj usługę i zobacz kontrolowany błąd. Następnie ustaw niewłaściwy Bearer w jednym z procesów i sprawdź, że odpowiedź nie ujawnia sekretu ani body upstream. Dodaj niedozwolone pole w JSON oraz pytanie przekraczające limit. Wreszcie skontroluj nagłówki i wyłączenie endpointu oraz strony z cache hostingu.

Zapisz konkretne wersje rdzenia, PHP, systemu i konfiguracji w dzienniku odbioru. Ten pakiet nie dostarcza rdzenia WordPress i nie wykonał integracji PHP. Nie przepisuj wyniku testów Python do rubryki „WordPress zaliczony”. Rozdzielenie tych dowodów ułatwia później znalezienie problemu: parser, ranking, autoryzacja, transport i interfejs mają oddzielne obowiązki oraz oddzielne warunki potwierdzenia.

PROMPT DO CODEXA / Codex: dziennik stagingu

Przygotuj odtwarzalny scenariusz odbioru całej integracji.

Cel: Przygotuj odtwarzalny scenariusz odbioru całej integracji.

Dane: README, wtyczka, usługa loopback, pięć fixture i dwa konta o różnych capabilities.

Wynik: Checklist z krokiem, oczekiwanym wynikiem, miejscem zapisu dowodu i polem wersji środowiska.

KONTROLA: Nie wpisuj wyników z góry. Ujmij zły nonce, brak capability, timeout, manipulację grupą i współdzielony cache.

Nowa treść unieważnia wcześniejsze fragmenty

Numer wersji oraz hash zabezpieczają spójność indeksu. Odpowiedź nie może korzystać z poprzedniego okna po aktualizacji.

Źródło zmienia się częściej niż jego `doc_id`. Redaktor poprawia warunek, aktualizuje datę lub zastępuje listę kroków. Przed ponownym importem zachowaj stabilną tożsamość dokumentu i zwiększ `version`. Importer oblicza nowy hash z wydobytego tekstu; identyfikatory `chunk` zawierają ten hash, dlatego stare fragmenty nie mogą udawać aktualnego stanu.

Nawet jeśli autor pozostawi ten sam numer wersji, zmiana treści nadal unieważni wcześniejsze ID. To mechanizm obrony spójności, nie zachęta do pomijania wersjonowania redakcyjnego. Cofnięcie numeru wersji jest odrzucane. Cały zapis pozostaje w transakcji, więc błąd jednego dokumentu nie powinien pozostawić innych w częściowo zmienionym stanie.

W trybie `openai embedding` przygotowujemy przed zapisami. Dzięki temu awaria dostawcy nie otwiera długiej transakcji z częściową treścią. Udań obliczenie wektorów nadal może zostać rozliczone przez dostawcę, nawet gdy późniejsza walidacja wersji zatrzyma import. Nie obiecujemy atomowości między rachunkiem API a lokalną bazą.

Po aktualizacji zadaj wcześniej zapisane pytanie kontrolne. Sprawdź wersję i `chunk_id` odpowiedzi, a nie tylko podobieństwo tekstu. Dodaj również próbę porównania starego cytowania z nowym odzyskanym kontekstem: powinno zostać odrzucone. W dalszej rozbudowie można zachować historię osobno do audytu, lecz wersje archiwalne nie mogą samoczynnie wracać do aktywnego rankingu. Ustal też, kto decyduje, że nowa treść została zatwierdzona do użycia.

Aktualizacja bez starych cytowań

01 Nowe źródło

ten sam `doc_id`, aktualna `version` i tekst

02 Przygotowanie

hash, fragmenty i wektory przed zapisami

03 Transakcja

zastąpienie poprzednich fragmentów w SQLite

04 Kontrola

nowe ID; stare cytowanie poza allowlistą

Wycofanie, purge i retencja

Brak źródła w rankingu nie oznacza usunięcia go ze wszystkich miejsc przechowywania.

Snapshot może oznaczyć `active=false` albo pominąć dokument. Importer wtedy wycofuje źródło i usuwa jego fragmenty. Metadane dokumentu pozostają w tabeli docs jako nieaktywne. Takie logiczne wycofanie jest przydatne do kontroli synchronizacji, ale nie jest pełnym wymazaniem danych. Nieaktywny rekord nie uczestniczy w pobraniu kandydatów.

Polecenie `purge` usuwa konkretny `doc_id` oraz jego fragmenty i wykonuje `VACUUM` w SQLite. Nie kasuje pliku źródłowego, manifestu, eksportu, kopii zapasowej ani ewentualnych dzienników innych usług. Jeżeli dane były wysłane rzeczywistemu dostawcy, jego zasady retencji są dodatkową kwestią. Ustawienie `store=false` dla Responses nie rozstrzyga samo wszystkich zasad przechowywania.

Ustal listę miejsc oraz właściciela usunięcia. W małym laboratorium obejmuje źródła lokalne, manifest, indeks i kopie. W produkcji dochodzą monitoring, kolejki, wysyłka dostawcy i materiały pracy redakcyjnej. Zapisuj dowód usunięcia przez identyfikator i datę, bez ponownego kopiowania poufnego tekstu do protokołu.

Przywrócenie kopii indeksu może odtworzyć wcześniej wycofane źródło. Dlatego proces odtwarzania musi ponownie uwzględnić aktualny rejestr usunięć i zatwierdzony manifest. Dostęp do kopii nie powinien być szerszy niż do danych źródłowych. Nie traktuj nowego hasła gateway jako sposobu usunięcia informacji: rotacja sekretu zmienia dostęp transportowy, a nie zawartość bazy czy backupu. Odbiór retencji wymaga sprawdzenia tych działań osobno.

PROMPT DO CODEXA / Codex: procedura wycofania źródła

Rozpisz odwołanie dostępu i usunięcie dokumentu bez obietnic niewykonanych działań.

Cel: Rozpisz odwołanie dostępu i usunięcie dokumentu bez obietnic niewykonanych działań.

Dane: `doc_id`, `active`, `snapshot`, CLI `purge`, źródło poza `webroot` oraz kopie indeksu.

Wynik: Tabela miejsca przechowywania, czynności, właściciela i dowodu wykonania.

KONTROLA: Oddziel brak w rankingu od pełnego usunięcia. Uwzględnij odtworzenie backupu i retencję dostawcy w trybie `openai`.

Koszty oraz obserwacja bez logowania treści

Koszt bazy wiedzy wynika z utrzymania źródeł, indeksowania i pytań. Mock nie daje prognozy rachunku dostawcy.

W trybie offline płacisz czasem operatora, CPU, dyskiem i utrzymaniem WordPress. Nie pojawia się rachunek zewnętrznego API. Mimo tego warto mierzyć czas importu, liczbę dokumentów, liczbę fragmentów i czas zapytania. Taki pomiar pomaga zauważyć wzrost bazy, lecz nie może zostać przedstawiony jako benchmark prawdziwego modelu.

Tryb openai dodaje koszt embeddings oraz generowania. Bieżący adapter przelicza fragmenty snapshot i wysyła po jednym fragmencie na request. To czytelny profil odniesienia, który przed większą skalą wymaga batchingu, cache zależnego od hash, budżetu oraz limitów liczby pytań. Nie zakładaj darmowej aktualizacji tylko dlatego, że version nie zmieniła się w WordPress.

Limit trzech fragmentów ogranicza kontekst, ale jego ilość nie jest jedynym źródłem kosztu. Znaczenie mają długość pytania, odpowiedzi, model, częstotliwość importów i błędy prowadzące do ponownych prób. Ten pakiet nie wykonuje automatycznego retry dostawcy. Zwiększenie timeoutu ani dodanie retry nie powinno następować bez oceny budżetu i niejednoznacznych skutków.

Logowanie pytań oraz treści dokumentów nie jest potrzebne do zwykłej obserwacji dostępności. Usługa wyłącza standardowy log body; zapisuj statusy, czas i identyfikator operacji bez sekretów. Dla produkcji zaplanuj ograniczanie ruchu, liczbę równoczesnych zadań, alarmy i kontrolę błędów. Lokalny ThreadingHTTPServer oraz pełny ranking Python nie stanowią gotowego serwera o potwierdzonej przepustowości. Wartości budżetu i czasu potwierdzisz we własnym środowisku.

Pomiar	Interpretacja
Liczba fragmentów	Rozmiar indeksu i kontekstu pracy
Czas offline	Koszt lokalnego mechanizmu
Koszt dostawcy	Osobny rzeczywisty pomiar
Statusy bez body	Dostępność bez kopiowania danych

Opcjonalny adapter rzeczywistego modelu

Przełączenie trybu wymaga konfiguracji, pełnego indeksu w zgodnej przestrzeni i osobnego testu jakości.

WDL_AI_MODE domyślnie ma wartość mock. Opcjonalna wartość openai uruchamia dołączony adapter urllib. Usługa odczytuje OPENAI_API_KEY, WDL_EMBED_MODEL, WDL_EMBED_DIMENSIONS oraz WDL_GENERATION_MODEL wyłącznie z własnego środowiska. Adres API jest stały, przekierowania są zablokowane, odpowiedź ma limit rozmiaru i każde wywołanie ograniczony czas.

Embeddings zwracają reprezentację sprawdzaną pod względem modelu, długości, skończonych liczb oraz niezerowej normy. Model i wymiary trafiają do settings i rekordów chunk. Zmiana przestrzeni wymaga pełnego reindeksu; query przy niezgodnym indeksie kończy się błędem. Nie mieszaj wektorów mocka z rzeczywistym modelem ani nie włączaj nowego modelu tylko po stronie pytania.

Generowanie używa Responses API z text.format ustawionym na json_schema. Parser przegląda output oraz content, szukając output_text. Odrzuca incomplete, odmowę protokołu i niepoprawny JSON. Następnie kontroluje cytowania względem odzyskanych ID i lokatorów. Format odpowiedzi jest sprawdzony lokalnym serwerem udającym dostawcę, bez prawdziwego klucza i wywołania API.

To kompletny adapter protokołu, lecz nie raport jakości semantycznej. Przed rzeczywistym uruchomieniem wybierz dostępny model obsługujący wymagany format, potwierdź przetwarzanie danych i budżet. Oddzielnie oceń synonimy, odpowiedzi niejednoznaczne, odmowę, injection i prawdziwość wniosków. Nie obiecujemy dostępności wybranego modelu na każdym koncie ani odporności na ataki po zaliczeniu lokalnych testów HTTP.

Tryb	Potwierdzone tutaj
mock	Offline, deterministyczny przepływ
openai adapter	Lokalne testy request/response
Rzeczywista jakość	Niewykonana; osobny odbiór
WordPress/PHP	Niewykonane; staging wymagany

Po takim przełączeniu ponownie uruchom macierz dostępu. Nowa reprezentacja może zmienić ranking, lecz nie może zmienić zasad widoczności dokumentów.

Testy, które sprawdzają właściwe granice

Raport testowy powinien mówić dokładnie, co wykonano. Zielony wynik protokołu nie zastępuje integracji i jakości odpowiedzi.

Pakiet zawiera dwadzieścia sześć testów unittest. Jeden z nich statycznie sprawdza filtr hasła eksportu i pominięcie chronionego rekordu przed odczytem treści; nie wykonuje WordPress ani WP-CLI. Sprawdzają ACL przed odczytem wektora i kontekstem, odmowę bez wyników, zmianę wersji i hash, wycofanie brakującego źródła oraz rollback. Kontrolują wymyślone cytowanie, fałszywy lokator, niedozwoloną grupę i strukturalne oddzielenie instrukcji od źródeł. To konkretne mechanizmy, nie ogólny test „AI działa”.

Testy HTTP lokalnej usługi obejmują wymagany Bearer, poprawny JSON, limit body i długości pytania oraz private, no-store. Parser TXT i prosty DOCX uruchomiono naprawdę. Opcjonalny pdftotext odczytał tekst z tymczasowego PDF. Ten wynik nie poświadcza OCR ani pełnej zgodności formatów. Składnia JavaScript i kompilacja plików Python również zostały sprawdzone.

Siedem testów adaptera używa wyłącznie lokalnego serwera udającego OpenAI. Kontrolują schema Structured Outputs, modele i wymiary, brak generation przy pustym kontekście, nieważne cytowanie oraz incomplete/refusal. Dane testowe nie potwierdzają rozumienia dokumentów ani ochrony rzeczywistego modelu. Nie wykonano tutaj żadnego płatnego wywołania dostawcy.

TEST_STATUS.md zapisuje komendy oraz ograniczenia. WordPress, PHP, aktywacja wtyczki, capability mapowanie, nonce i cache hostingu pozostają do testu stagingu. Potem wykonujesz oddzielny pomiar prawdziwego dostawcy, jeśli go wybierzesz. Nie przepisuj automatycznie liczby testów do obietnicy gotowości produkcyjnej. Odbiór ma wskazać granicę między wykonanym mechanizmem, planowanym wdrożeniem i niewykonanym pomiarem.

PROMPT DO CODEXA / Codex: uczciwy protokół odbioru

Złóż raport potwierdzony wynikami komend i prób środowiskowych.

Cel: Złóż raport potwierdzony wynikami komend i prób środowiskowych.

Dane: TEST_STATUS.md, 26 testów offline, node --check, py_compile oraz niewykonane PHP/WordPress/API.

Wynik: Tabela wymagania, testu, dowodu, wyniku i ograniczenia; osobna lista dalszych prób.

KONTROLA: Nie rozciągaj lokalnego mocka na jakość modelu, CDN ani E2E. Każdy PASS musi mieć rzeczywiście wykonany dowód.

Projekt przyjęty oraz kolejny bezpieczny etap

Pierwszy wynik to baza, którą potrafisz wyjaśnić i odtworzyć. Rozbudowę wybierasz według zmierzonego problemu.

Gotowy etap laboratoryjny łączy kontrolowane dokumenty z prywatną wyszukiwarką. Znasz miejsce źródeł, regułę klasyfikacji, kontrakt manifestu i sposób unieważniania fragmentów. WordPress jest właścicielem sesji oraz capabilities. Lokalna usługa wybiera tylko aktywne i dozwolone dane, a odpowiedź pokazuje sprawdzone cytowania albo kończy się odmową.

Przyjmij projekt po odtworzeniu komend oraz macierzy użytkowników na własnym stagingu. Zachowaj dziennik wersji, konfigurację cache, sposób przekazania sekretu i procedurę aktualizacji. Wskaż osobę zatwierdzającą dokumenty i osobę reagującą na błędy. Bez tych odpowiedzialności indeks stopniowo zamienia się w kolekcję źródeł, których aktualności nikt nie potrafi potwierdzić.

Dalszy etap zależy od potrzeby. Słabe dopasowanie synonimów uzasadnia ocenę rzeczywistych embeddings. Duża liczba fragmentów może wymagać innego silnika, batchingu lub pracy importu w tle. Potrzeba dokładnych stron PDF wymaga lepszego parsera i mapowania lokatorów. Nie dodawaj tych elementów tylko dlatego, że systemy AI często o nich opowiadają.

W każdym rozszerzeniu zachowaj granice: prywatne oryginały poza uploads, grupy z WordPress, ACL przed kontekstem, sekret wyłącznie na serwerze i brak współdzielonego cache. Oceń także nowe skutki: zewnętrzne przetwarzanie, koszt, retencję i narzędzia modelu. Tom kolejny może wykorzystać tę bazę do audytu eksploatacji. Dobry odbiór pokaże zarówno to, co system robi wiarygodnie, jak i sytuacje, w których świadomie odmawia odpowiedzi albo potrzebuje interwencji człowieka.

Dalsza potrzeba	Następny dowód
Synonimy	Pomiar prawdziwego embeddingu
Większa baza	Własny benchmark i budżet
Precyzyjne lokatory	Porównanie parsera z oryginałem
Produkcja	Staging, backup, monitoring i rollback

Bibliografia i granice potwierdzenia

Źródła służą weryfikacji kontraktu implementacji. Nie zastępują testu konkretnej instalacji ani pomiaru rzeczywistego modelu.

Dokumentację oficjalną odczytano 13 października 2026. Przed wdrożeniem ponownie sprawdź zgodność z wersją rdzenia, środowiskiem PHP oraz wybranym modelem. Ten tom nie zawiera rekomendacji stałych cen ani zapewnienia dostępności modelu na każdym koncie. Konfiguracja pozostaje decyzją operatora i właściciela danych.

Sześć pozycji poniżej wspiera opis REST, transportu, nonce oraz opcjonalnego adaptera. Pełne przykłady i protokół testów są w EXAMPLES. Wszystkie dokumenty WP DEMO LAB są fikcyjne; tryb domyślny jest niesemantycznym mockiem. Lokalny test HTTP OpenAI potwierdza strukturę komunikacji, nie odpowiedzi rzeczywistego modelu. Integracja PHP i WordPress wymaga własnego stagingu, a ochrona oryginalnych plików pozostaje oddzielna od ACL indeksu. Zapisz własny wynik odbioru zamiast traktować tę bibliografię jako certyfikat gotowości produkcyjnej.

Authentication – REST API Handbook | Developer.WordPress.org

<https://developer.wordpress.org/rest-api/using-the-rest-api/authentication/>

Cookie, X-WP-Nonce oraz obowiązek odpowiedniej capability. Oficjalna dokumentacja sprawdzona 13.10.2026.

wp_remote_post() – Function | Developer.WordPress.org

https://developer.wordpress.org/reference/functions/wp_remote_post/

HTTP POST, WP_Error oraz konfiguracja żądania. W projekcie URI jest stały i lokalny.

register_rest_route() – Function | Developer.WordPress.org

https://developer.wordpress.org/reference/functions/register_rest_route/

Rejestracja route w rest_api_init i osobna permission_callback; wymagany odbiór w WordPress.

Nonces – Common APIs Handbook | Developer.WordPress.org

<https://developer.wordpress.org/apis/security/nonces/>

Nonce wspiera ochronę żądania, ale nie zastępuje uwierzytelnienia ani autoryzacji.

Create embeddings | OpenAI API Reference

<https://developers.openai.com/api/reference/resources/embeddings/methods/create>

Oficjalny kontrakt HTTP embeddings, model, wymiary i wynik. Bez rzeczywistego wywołania dostawcy.

Structured model outputs | OpenAI API

<https://developers.openai.com/api/docs/guides/structured-outputs?api-mode=responses>

Responses, text.format i json_schema. Test protokołu lokalnie; prawdziwa jakość pozostaje niezmierna.

Zachowaj datę ponownej weryfikacji w protokole własnego wdrożenia.