Zapolu

Audyt wydajności: deep dive (przykładowy raport)

> **To jest wzór.** „Acme Outdoor s.r.o.” to fikcyjny klient. Ustalenia są kompozytami z prawdziwych audytów, które dostarczyliśmy dla innych sklepów, ze zmienionymi szczegółami i liczbami. Struktura i poziom szczegółu odpowiadają temu, co dostaje płacący klient.

Klient	Acme Outdoor s.r.o. (fikcyjny)
Sklep	acme-outdoor.example (Magento 2.4.6, frontend Luma)
Stack	Varnish 7, Redis 7, MySQL 8, 2 nody aplikacyjne za load balancerem
Okno danych	28 dni danych z realnego ruchu, 14 dni trace’ów APM, jeden test obciążeniowy
Narzędzia	New Relic, CrUX, Lighthouse, k6, varnishstat, MySQL slow log
Autor	Luboš Zápotočný, Zapolu s.r.o.
Zlecenie	Deep dive, 7 dni roboczych, stała cena

Dla szybkiego przeglądu przeczytaj rozdział 1 i rozdział 5. Wszystko pomiędzy nimi to materiał dowodowy.

1. Podsumowanie

Spowolnienie sklepu ma źródło po stronie serwera, nie w zbyt słabym sprzęcie. LCP z realnego ruchu na stronach katalogu to **4,6 s na p75** (cel: ≤ 2,5 s), a stoi za tym **TTFB 1,9 s**. TTFB jest wysokie, bo full-page cache jest faktycznie wyłączony: Varnish obsługuje tylko **34%** żądań katalogowych, a 38% całego czasu serwera idzie na renderowanie stron produktowych, średnio 2,2 s na żądanie. Podczas majowego wieczoru kampanii checkout p95 sięgnął **6,1 s**, a **3,4% żądań kończyło się błędem**, podczas gdy CPU serwerów aplikacyjnych nie przekroczyło 12%.

Dalej jest jedenaście ustaleń. Sześć z nich kosztuje dzień pracy lub mniej, a dwa z tych sześciu dotyczą tylko plików konfiguracyjnych. Wykonanie wierszy 1–5 planu napraw powinno sprowadzić LCP katalogu poniżej 2,5 s na p75 i zatrzymać awarie checkoutu w szczycie; wiersz 6 sprowadza potem checkout p95 poniżej 2 s w trakcie kampanii. Pozostałe wiersze są uszeregowane według zwrotu, więc plan można uciąć w dowolnym miejscu.

W przeliczeniu na pieniądze: w ciągu trzech godzin kampanii 12 maja błędem zakończyło się około 230 checkoutów. Przy średniej wartości zamówienia 1 800 CZK (≈ 72 €) to około 415 000 CZK w zamówieniach, które się nie udały, przed doliczeniem klientów, którzy porzucili checkout po sześciu sekundach oczekiwania. Pozostałe ustalenia podajemy w milisekundach; kwota w pieniądzach byłaby przy nich spekulatywna, więc ją pomijamy.

2. Co i jak mierzyliśmy

- **Metryki realnych użytkowników:** Core Web Vitals z 28-dniowego zapytania do CrUX, segmentowane po typie strony (strona główna / kategoria / produkt / checkout). Syntetyczne przebiegi Lighthouse służyły tylko

do reprodukcji ustaleń, nie do ich oceniania. Jak sklep wypada na tle progów Google:

Metryka (p75, dane z realnego ruchu)	Sklep	Próg	Werdykt
LCP	4,6 s	< 2,5 s	nie zdaje
INP	280 ms	< 200 ms	nie zdaje
CLS	0,04	< 0,1	zdaje
FCP	2,9 s	< 1,8 s	nie zdaje
TTFB	1,9 s	< 0,8 s	nie zdaje

- **Trace po stronie serwera:** profil transakcji z New Relic za 14 dni, czyli dokąd naprawdę idzie czas serwera:

Transakcja	Udział w czasie serwera	Śr. odpowiedź
`catalog/product/view`	38%	2,2 s
`catalog/category/view`	17%	1,8 s
checkout REST (koszyk + dostawa)	11%	1,4 s (6,1 s p95 w szczycie)

  Do tego MySQL slow query log, profil wywołań usług zewnętrznych i statystyki hit/miss Varnisha per wzorzec URL z `varnishstat`.

- **Test obciążeniowy:** replay kształtu ruchu z kampanii 12 maja (0 → 420 req/s w 20 minut) na stagingu, z tym samym stanem cache co na produkcji.
- **Przegląd konfiguracji:** ustawienia PHP i OPcache, autoloader Composera, `env.php`, VCL Varnisha i ustawienia MySQL, wszystko porównane z udokumentowanymi zaleceniami dla Magento.
- **Audyt bundli i assetów:** co wysyła się przy pierwszym wyświetleniu strony produktu, co blokuje render, a co jest zbędnym obciążeniem.

Przy większości ustaleń cytujemy dokładne polecenie albo zapytanie, którego użyliśmy: po pierwsze, żeby Twój zespół mógł sprawdzić naszą pracę, po drugie dlatego, że te same sondy zadziałają na każdym sklepie Magento, także na Twoim. Przy każdej liczbie niżej napisano, skąd pochodzi; tam, gdzie efekt ustalenia jest szacunkiem, jest tak oznaczony.

3. Przegląd ustaleń

Wysoka: dotyka klientów albo zależy od niej przychód. **Średnia**: zespół odczuwa ją codziennie albo po cichu uszkadza dane. **Niska**: niewielki nakład, a usuwa realne ryzyko operacyjne.

#	Ustalenie	Waga	Praca	Oczekiwany efekt
F1	Full-page cache jest faktycznie wyłączony i nie da się go unieważnić	Wysoka	1-2 dni	–700 ms TTFB na p75 w katalogu
F2	PHP runtime i autoloader źle skonfigurowane pod Magento	Wysoka	0,5 dnia	–400 ms na każdym niecache'owanym żądaniu
F3	Checkout blokuje się na synchronicznym odpytaniu ERP o stany	Wysoka	3–5 dni	–2,8 s checkout p95 w szczycie; koniec błędów timeout
F4	Frontend wysyła 2,3 MB JavaScriptu, blokując render	Wysoka	projekt	–1,2 s LCP (rozwiązania przejściowe: –0,4 s)

#	Ustalenie	Waga	Praca	Oczekiwany efekt
F5	Zdjęcia produktów: 1,6 MB PNG, bez responsywnych rozmiarów	Średnia	1 dzień	−0,6 s LCP na stronach produktów
F6	Indeksery zostawione na „Update on Save”	Średnia	0,5 dnia	Zapis w adminie 30 s → ~2 s; koniec rozbieżności stanów magazynowych
F7	Pula PHP-FPM wyczerpana przez session locki w szczycie	Wysoka	2 dni	Checkout przeżywa ruch kampanii
F8	Varnish bez trybu grace	Niska	0,5 dnia	Amortyzuje falę po każdym flushu cache
F9	Dwa rozszerzenia nasłuchują tego samego eventu; jedno porzucone	Średnia	1 dzień	−300 ms na dodaniu do koszyka
F10	41 GB martwych tabel temp po indekserach w MySQL	Niska	0,5 dnia	Backupy mniejsze o 60%; restore w około połowę czasu
F11	Grid zamówień w adminie filtruje po niezindeksowanej kolumnie	Niska	0,5 dnia	Grid zamówień 12 s → poniżej 1 s

4. Ustalenia szczegółowo

F1: Full-page cache jest faktycznie wyłączony i nie da się go unieważnić (Wysoka)

Co zobaczyliśmy. Varnish raportuje na URL-ach katalogowych hit rate 34%. Dla katalogu tej wielkości, przy tym udziale crawlerów, normą jest ponad 90%. Każda strona kategorii i produktu niesie `X-Magento-Cache-Debug: MISS`. Przyczyny są dwie, niezależne od siebie:

1. Zmodyfikowany `default.xml` oznacza jeden blok w nagłówku (przełącznik store view czytający sesję klienta) jako `cacheable="false"`. W Magento jeden niecache’owalny blok w layoucie czyni niecache’owalną **całą stronę**. Blok pojawił się w listopadowej aktualizacji rozszerzenia; hit rate załamał się w tym samym tygodniu.
2. Varnish jest włączony w adminie (`system/full_page_cache/caching_application = 2`), ale w `env.php` nie ma sekcji `http_cache_hosts`, więc Magento w ogóle nie umie wysyłać żądań purge do Varnisha. Zespół obchodzi to restartem Varnisha po każdym deployu; właśnie dlatego deploye są problematyczne (zob. F8).

Jak to zweryfikowaliśmy:

```
$ curl -sI https://acme-outdoor.example/namioty | grep -i cache-debug
X-Magento-Cache-Debug: MISS
# tak samo na każdym sprawdzonym URL-u

$ grep -R 'cacheable="false"' app/design app/code | wc -l
1
# przełącznik store view w default.xml

$ php -r 'var_export(array_key_exists("http_cache_hosts", (include "app/etc/env.php")));'
false
# Magento nie umie purge'ować Varnisha
```

Naprawa. Renderować przełącznik przez private content (customer data JS), żeby sama strona pozostała cache’owalna (udokumentowany wzorzec dla fragmentów zależnych od sesji), i dodać `http_cache_hosts` do

`env.php`, żeby działało unieważnianie po tagach, a restarty mogły się skończyć.

Praca / efekt / ryzyko. 1–2 dni z testami regresyjnymi. Oczekiwane –700 ms TTFB na p75 katalogu (zmierzone: cache'owana odpowiedź katalogowa wraca w 80 ms, niecache'owana w 1,9 s). Ryzyko: niskie; obie zmiany są ograniczone i odwracalne per deploy.

F2: PHP runtime i autoloader źle skonfigurowane pod Magento (Wysoka)

Co zobaczyliśmy. Codebase zawiera ~125 000 plików PHP; OPcache jest ustawiony na 10 000. Autoloader Composera nigdy nie był optymalizowany: `vendor/composer/autoload_static.php` nie zawiera class mapy, a trace'y New Relic pokazują całe sekundy spędzone w rozwiązywaniu `ObjectManager` factory na zimnych ścieżkach. Obecne kontra zalecane:

Ustawienie	Obecne	Zalecane
`opcache.max_accelerated_files`	10 000	100 000
`opcache.memory_consumption`	128	512
`opcache.validate_timestamps`	On, rewalidacja co 2 s	Off; reset przy deployu
`opcache.enable_cli`	Off	On (indeksery i crony to też PHP)
`realpath_cache_size`	4M	32M
`realpath_cache_ttl`	120	7200
Autoloader Composera	nieoptymalizowany	`dump-autoload --optimize` w buildzie

Jak to zweryfikowaliśmy:

```
$ find . -type f -name '*.php' | wc -l
124862
# plików PHP w codebase

$ php -i | grep 'opcache.max_accelerated_files'
opcache.max_accelerated_files => 10000
# mieści 8% z nich

$ grep -c 'classMap' vendor/composer/autoload_static.php
0
# autoloader nigdy nie był optymalizowany
```

Dlaczego. Przy tych wartościach PHP przy każdym żądaniu od nowa czyta z dysku i kompiluje dużą część codebase; profile przypisują ponad połowę czasu niecache'owanego renderu ładowaniu klas i składaniu strony, a nie logice biznesowej.

Naprawa. Jedna zmiana konfiguracji plus jeden krok w buildzie. Jedyne sprzężenie do uszanowania: przy wyłączonym `validate_timestamps` pipeline deployu musi przy release resetować OPcache. Dokładny krok dla Twojego obecnego pipeline'u opisaliśmy w planie napraw.

Praca / efekt / ryzyko. 0,5 dnia. Szacunkowo –400 ms na każdym niecache'owanym żądaniu (plus szybsze crony i indeksery jako efekt uboczny). Ryzyko: niskie, o ile krok w deployu wyląduje razem z konfiguracją.

F3: Checkout blokuje się na synchronicznym odpytaniu ERP o stany (Wysoka)

Co zobaczyliśmy. Na kroku dostawy trace pokazuje synchroniczne wywołanie HTTPS do endpointu stanów ERP z 3-sekundowym timeoutem i jednym ponowieniem; profil usług zewnętrznych przypisuje temu jednemu endpointowi 92% całego czekania na systemy zewnętrzne. Pod obciążeniem kampanii ERP zwalnia, wywołania wybierają cały timeout, a żądania checkoutu zaczynają się piętrzyć. Dokładnie stąd bierze się p95 6,1 s i 3,4% błędów. Test obciążeniowy odtwarza to na stagingu.

Dlaczego. Potwierdzanie stanów w czasie rzeczywistym doszło po zeszłorocznym incydencie z nadsprzedażą. Zamiar jest słuszny, tylko wkłada latencję cudzego systemu w każdy checkout.

Naprawa. Wyprowadzić uzgadnianie stanów poza ścieżkę żądania: rezerwować na lokalnym magazynie, potwierdzać asynchronicznie i alarmować przy rozbieżności. Nadsprzedaż pozostaje pod kontrolą, a checkout przestaje czekać na ERP. (To ten sam wzorzec, który opisujemy w tekście o synchronizacji stanów między ERP a sklepem; odnośnik udostępnimy na życzenie.)

Praca / efekt / ryzyko. 3–5 dni z alertem rozbieżności stanów. Zdejmuje ~2,8 s z checkout p95 w szczycie i całkowicie usuwa błędy timeout (jedno i drugie zmierzone w teście obciążeniowym z zaślepionym wywołaniem). Ryzyko: średnie; wymaga pisemnej akceptacji nowego okna nadsprzedaży (szacunkowo < 0,1% zamówień kampanijnych, wobec dzisiejszych 3,4% nieudanych checkoutów).

F4: Frontend wysyła 2,3 MB JavaScriptu (Wysoka)

Co zobaczyliśmy. Pierwsze wyświetlenie strony produktu ładuje 2,3 MB JavaScriptu w 61 plikach; blokowanie głównego wątku na średniej klasy telefonie to 2,9 s. Cztery z czternastu tagów marketingowych nie mają już za sobą działającego konta.

Dlaczego. Po części architektura frontendu Luma, po części osiem lat przyrastania tagów; żaden pojedynczy tag nie przeważa w sumie.

Naprawa, krótkoterminowo (ten kwartał). Usunąć cztery martwe tagi, odroczyć bundle analityki, leniwie ładować widget opinii pod foldem. Szacunkowo −0,4 s LCP, 2 dni.

Naprawa, właściwa (osobna decyzja). Przebudowa frontendu na Hyvä. Na porównywalnych katalogach sprowadza LCP do zieleni na p75 i obniża koszt utrzymania frontendu; to projekt na 6–10 tygodni i zasługuje na własną wycenę, nie na wiersz na liście napraw. Wspominamy o nim tutaj, aby krótkoterminowy szacunek powyżej nie był traktowany jako osiągalne maksimum.

F7: Pula PHP-FPM wyczerpana przez session locki w szczycie (Wysoka)

Co zobaczyliśmy. Podczas testu obciążeniowego błędy zaczynają się około 280 req/s, podczas gdy CPU siedzi na 12%. Strona statusu FPM pokazuje wszystkie workery zajęte; trace'y pokazują czekanie na session locki w Redisie: klienci odpytujący stronę potwierdzenia zamówienia trzymają blokady, które kolejkują każde kolejne żądanie tej samej sesji.

Naprawa. Wyłączyć blokowanie sesji dla endpointów pollingu, podnieść liczbę workerów do tego, na co realnie pozwala zapas pamięci (obecna liczba pochodzi z mniejszego typu instancji), i dodać alert nasycenia FPM; ta awaria pozostała niezauważona, bo wykresy CPU nie budziły podejrzeń.

Praca / efekt / ryzyko. 2 dni z powtórką testu obciążeniowego. Razem z F3 staging przeżywa 420 req/s z checkout p95 1,9 s. Ryzyko: niskie.

F5, F6, F8, F9, F10, F11 w skrócie

- **F5 Zdjęcia:** hero produktów to 1,6 MB PNG serwowane w jednym rozmiarze każdemu urządzeniu. Konwersja do WebP + responsywne rozmiary przez CDN: 1 dzień, −0,6 s LCP na stronach produktów (zmierzone na przekonwertowanej próbce).
- **F6 Indeksery:** przełączone na „Update on Save” przy lutowym imporcie masowym i nigdy z powrotem. Zapis produktu w adminie trwa 30 s, a rozbieżność między stanami w sklepie a rzeczywistością magazynu wynika z wyścigów przy reindeksie. Przywrócić indeksowanie z harmonogramu: 0,5 dnia.
- **F8 Varnish grace:** każdy flush cache wysyła dziś całą falę ruchu na PHP. Tryb grace serwuje stare obiekty, dopóki cache się nie zapełni na nowo: 0,5 dnia.
- **F9 Zduplikowane observery:** dwa rozszerzenia subskrybują ten sam event checkoutu; jedno jest porzucone przez vendora od 2023 roku i przy każdym wywołaniu od nowa ładuje quote. Usunąć je (jego funkcja jest nieużywana): 1 dzień z weryfikacją, −300 ms na dodaniu do koszyka.
- **F10 Porządki w bazie:** baza niesie 41 GB tabel `catalogrule_product__temp*` (pozostałości przerwanych przebiegów indekserów ze starszej wersji Magento, która po sobie nie sprzątała) plus własną tabelę kolejki z 12 milionami wierszy bez żadnej retencji. Znajduje je jedno zapytanie do `information_schema`:

  ```
  mysql> SELECT COUNT(*) tables, ROUND(SUM(data_length+index_length)
      -> /1024/1024/1024, 1) gb FROM information_schema.tables
      -> WHERE table_name LIKE 'catalogrule\_product\_\_temp%';
  +--------+------+
  | tables | gb   |
  +--------+------+
  |     96 | 41.2 |
  +--------+------+
  ```

  Usunięcie pozostałości i dodanie czyszczenia: 0,5 dnia. Backupy maleją z 68 GB do 27 GB, a restore trwa mniej więcej połowę czasu, co liczy się głównie w dniu, w którym restore jest naprawdę potrzebny.
- **F11 Grid zamówień w adminie:** kolumna dodana dla zespołu wysyłek filtruje po niezindeksowanym atrybucie; grid ładuje się 12 s i za każdym razem wiąże jednego workera. Jedna migracja indeksu: 0,5 dnia.

5. Plan posortowany według zwrotu z nakładu

Kolejność	Pozycje	Praca	Co dostajesz
1	F2 + F8	1 dzień	Każde niecache’owane żądanie −400 ms; flushe cache nie wywołują już skoku obciążenia
2	F1	1–2 dni	TTFB katalogu −700 ms; deploye nie wymagają już restartu Varnisha
3	F6 + F10 + F11	1,5 dnia	Admin znowu używalny; koniec rozbieżności stanów magazynowych; backupy o połowę
4	F7	2 dni	Checkout przestaje zawodzić w szczycie
5	F5 + F9	2 dni	LCP strony produktu −0,9 s łącznie
6	F3	3–5 dni	Checkout p95 poniżej 2 s w trakcie kampanii
7	F4 krótkoterminowo	2 dni	LCP −0,4 s, zanim zapadnie decyzja o froncie

Kolejność	Pozycje	Praca	Co dostajesz
8	F4 przebudowa	osobna wycena	LCP w zieleni na stałe

Wiersze 1–6 to razem 10–14 dni inżynierskich i rozwiązują każdą awarię, którą klient może poczuć. Po dowolnym wierszu możesz skończyć, a praca do tego momentu broni się sama. Wiersz 6 jako jedyny dotyka architektury ścieżki zamówienia; niezależnie od tego, kto go wykona, zleć przegląd projektu doświadczonemu inżynierowi.

Po wierszu 6 zmierz ponownie tym samym zapytaniem o dane z realnego ruchu, od którego ten raport się zaczął. Jeśli liczby nie ruszą się zgodnie z przewidywaniem, skontaktuj się z nami, a ustalimy przyczynę.

6. Czego nie sprawdzaliśmy

Bezpieczeństwo, SEO, dostępność i jakość kodu poza profilowanymi gorącymi ścieżkami są poza zakresem tego zlecenia. Audyt obejmuje domyślny store view; store view B2B dzieli stack, ale nie był osobno mierzony. Jedno ograniczenie dostępów: nasz użytkownik audytowy nie miał w oknie pomiarowym dostępu do admin socketu Varnisha, więc liczniki runtime Varnisha pochodzą ze snapshotów `varnishstat` wyeksportowanych przez hosting, a nie z żywej inspekcji.

7. Kolejne kroki

Plan napraw wykona każdy kompetentny zespół, łącznie z Twoim: każde ustalenie nazywa zmianę, nie tylko objaw. Jeśli mamy go wykonać my, wiersze 1–6 mieszczą się w dwu-, trzytygodniowym zleceniu za stałą cenę na naszych standardowych warunkach współpracy.

Pytania o poszczególne ustalenia: info@zapolu.com, albo umów rozmowę i przynieś własne liczby; podczas rozmowy powiemy Ci, czy audyt ma sens.