# Część 1: User Experience — RUM Angular (web‑vitals, błędy JS, korelacja z APM)

Scenariusz prowadzącego (15–20 min) — stack: .NET Core + Angular + Node.js

## 0) Cel i efekty

• Zrozumieć, jak w Kibanie interpretować metryki RUM (Core Web Vitals) dla Angulara i jak wykrywać regresje UX.

• Nauczyć się identyfikować i analizować błędy JavaScript z poziomu RUM.

• Opanować praktyczną korelację RUM → APM (.NET/Node) → Infra (Metricbeat) w celu szybkiego triage.

## 1) Przygotowanie „dzień wcześniej” (poza sesją)

• Dostępy: uczestnicy mają Observability (User Experience, APM, Infrastructure, Uptime), Discover/Logs, Alerts & Rules, Cases.

• Instrumentacja RUM: skrypt APM RUM JS w Angular (prod/stage) + ustawione service.name, service.environment, service.version.

• Ruch testowy: upewnij się, że w wybranym oknie czasu są wizyty (w razie potrzeby wygeneruj), by widzieć LCP/INP/CLS i błędy JS.

• Spójne nazewnictwo: w APM (.NET/Node) używasz tych samych wartości service.name/service.environment co w RUM (łatwiejsza korelacja).

• Import: miej wgrane Saved Objects (NDJSON) i Saved Queries (z poprzedniej części) — będą pod ręką w trakcie pokazu.

• Źródła map: jeżeli chcesz czytelne stacktrace’y błędów JS, zadbaj o sourcemaps (upload/serwowanie z mapą).

• Polityki: CORS i distributed tracing dla RUM ↔ backend (nagłówki tracujące) włączone na reverse proxy (IIS/Nginx/Traefik).

## 2) Start sesji (ustawienia i filtr)

• Ustaw zakres czasu na Last 24 hours (lub 7 days, jeśli mało ruchu).

• Włącz auto‑refresh (1–5 min) na czas pokazu.

• Na pasku filtrów dodaj: service.environment="prod" (lub „staging”), ewentualnie release/service.version.

## 3) Widok „User Experience” — co tu jest

• Core Web Vitals: p75 LCP, p75 INP, p75 CLS — szybki barometr UX. Sprawdź trend vs poprzedni okres i segmentacje (urządzenia/przeglądarki/region).

• Najgorsze widoki/ścieżki (Worst pages): wykrywanie tras z regresją. Dobrze rozróżniać page‑load vs route‑change (SPA).

• Błędy JS/AJAX: liczność i tempo występowania, przegląd stacktrace’ów, user agent i wpływ na użytkowników.

• Segmentacje: według przeglądarki, urządzenia, kraju, wersji aplikacji (service.version). Używaj ich do zawężania źródła problemu.

• Nawigacja do APM: z problematycznego widoku możesz przejść do powiązanej usługi backendowej (lub ręcznie do APM).

## 4) Scenariusz pokazu – krok po kroku (demo)

• Krok A: Otwórz Observability → User Experience. Sprawdź p75 LCP/INP/CLS. Jeśli któryś wskaźnik wysoki — zanotuj (np. LCP>2.5s).

• Krok B: Przejdź do „Worst pages” i wybierz trasę z regresją vs wczoraj. Zanotuj różnice w porównaniu do poprzedniego okresu.

• Krok C: Otwórz kartę błędów JS (Errors). Wybierz najczęstszy typ błędu (np. TypeError) i sprawdź stacktrace/UA/release.

• Krok D: Korelacja z APM — z widoku RUM kliknij link do APM (jeśli dostępny) lub otwórz APM → Services i wybierz odpowiadającą usługę backendu.

• Krok E: W APM sprawdź Transactions (p95/p99), Dependencies (DB/HTTP/Redis) oraz Errors (czy pokrywają się okna czasowe).

• Krok F: Jeśli w APM widać symptomy przeciążenia — otwórz Infrastructure: host/kontener/reverse proxy/DB i zweryfikuj CPU/memory/5xx/evictions.

• Krok G: Otwórz Discover i użyj Saved Query „RUM – Błędy JS (24h)” oraz „APM – Błędy (24h)” dla potwierdzenia korelacji zdarzeń.

• Krok H: Załóż Case z krótką notatką (ścieżka, godzina, wpływ, hipoteza) i przypisz właściciela.

## 5) Analiza błędów JS – praktyczne wskazówki

• Używaj sourcemaps, by stacktrace był czytelny (plik .map dostępny w momencie analizy).

• Grupuj błędy po typie/wiadomości i sprawdzaj, czy dotyczą konkretnej wersji (service.version) lub przeglądarki.

• Weryfikuj, czy błędy JS występują równolegle z regresją web‑vitals — np. duże INP może korelować z ciężkimi handlerami zdarzeń.

• Sprawdzaj, czy błędy nie pochodzą z zewnętrznych skryptów (third‑party); jeśli tak — rozważ lazy‑loading/odseparowanie.

## 6) Korelacja RUM ↔ APM – jak to robić pewnie

• Spójne pola: trzymaj ten sam service.name/service.environment między RUM i backendem (APM .NET/Node).

• Distributed tracing: włącz propagację nagłówków trace‑context między przeglądarką a backendem (konfiguracja CORS/proxy).

• Okno czasu: zawsze porównuj te same zakresy i podglądaj piki w obu widokach (RUM i APM) w identycznych ramach czasowych.

• Ścieżka: porównuj konkretne route (RUM) z odpowiadającymi transakcjami HTTP (APM) — czy p95 wzrosło jednocześnie?

## 7) Saved Queries – szybki dostęp (KQL)

• RUM – Błędy JS (24h)

service.framework.name : "angular" and processor.event : "error" and @timestamp >= now-24h

• RUM – Page-load i Route-change (24h)

service.framework.name : "angular" and transaction.type : ("page-load","route-change") and @timestamp >= now-24h

• APM – Transakcje HTTP (24h)

processor.event : "transaction" and transaction.type : "request" and @timestamp >= now-24h

• APM – Błędy (24h)

processor.event : "error" and @timestamp >= now-24h

## 8) SLO/SLA – progi referencyjne (do uzgodnienia w zespole)

• Przykładowo: p75 LCP ≤ 2.5 s, p75 INP ≤ 200 ms, p75 CLS ≤ 0.1. Traktuj to jako punkt wyjścia i dopasuj do aplikacji.

• Zdefiniuj progi alertów RUM; eskaluj tylko stałe przekroczenia (np. 3 kolejne okna 15‑min).

## 9) Lab – praktyka dla uczestników

• Ćw. 1: Znajdź najgorszą trasę (Worst pages) z ostatnich 24 h i sprawdź RUM web‑vitals per przeglądarka.

• Ćw. 2: Odszukaj najczęściej występujący błąd JS; zapisz wpływ (liczba sesji) i wersję (service.version).

• Ćw. 3: W APM sprawdź, czy backend tej ścieżki miał wzrost p95/error rate w tym samym oknie czasu.

• Ćw. 4: Załóż Case, wklej linki do widoków RUM/APM i zaproponuj hipotezę przyczyny.

## 10) Pułapki i dobre praktyki

• Mały ruch = niestabilne p75/p95. Analizuj 7 dni lub agreguj po większym oknie.

• Ad‑blockery i polityki prywatności mogą redukować próbki RUM — miej to na uwadze.

• Brak sourcemaps utrudnia debug JS – zapewnij dostępne .map dla produkcji (z ochroną dostępu).

• SPA routing: upewnij się, że route‑change jest poprawnie rozpoznawany (konfiguracja instrumentacji RUM).

• Unikaj dublowania metryk (RUM/OTel/Prometheus); trzymaj jednolitą ścieżkę danych.

## 11) Mini‑ściąga dla prowadzącego

• Ustaw Last 24 hours + auto‑refresh 1–5 min; filtr service.environment.

• User Experience: sprawdź p75 LCP/INP/CLS → Worst pages → Errors (JS).

• Korelacja: APM (Transactions/Dependencies/Errors) → Infrastructure (CPU/5xx/DB).

• Discover: użyj Saved Queries (RUM/APM).

• Załóż Case i przypisz właściciela.