RUM JavaScript dla Angular 15 — dokumentacja wdrożeniowa (PL)

Integracja z Kibana APM / Applications (frontend RUM + distributed tracing)  
*Data: 2025-08-21 06:05*

# 1) Cel i korzyści

* Pomiar rzeczywistego UX w przeglądarce: page-load, route-change, Web Vitals‑like sygnały, błędy JS, long tasks.
* Pełna ścieżka żądania z distributed tracing (traceparent/tracestate) — od frontu do backendu.
* Szybkie wnioski w Kibanie: p50/p95/p99 latencji, error rate, mapa zależności, ślady.

# 2) Wymagania i przygotowanie

* Angular 15 (działa również z ≥12) i Node/npm.
* APM Server z włączonym RUM oraz poprawnym CORS.

Minimalne ustawienia APM Server (przykładowo w apm-server.yml / przez Fleet):

apm-server.rum.enabled: true  
# Dozwolone źródła (ustaw NAZWY domen — nie '\*' w produkcji):  
apm-server.rum.allow\_origins: ["https://app.twojadomena.pl"]  
# (Opcjonalnie) ograniczenie usług i limity dla anon. auth  
# apm-server.auth.anonymous.allow\_service: ["angular-app"]

# 3) Instalacja pakietu dla Angulara

Zainstaluj oficjalną integrację Angular:

npm install @elastic/apm-rum-angular --save

Pakiet zapewnia ApmModule/ApmService oraz ApmErrorHandler.

# 4) Instrumentacja aplikacji

## 4A) Projekt z AppModule

import { NgModule, ErrorHandler } from '@angular/core';  
import { BrowserModule } from '@angular/platform-browser';  
import { Routes, RouterModule } from '@angular/router';  
import { ApmModule, ApmService, ApmErrorHandler } from '@elastic/apm-rum-angular';  
import { AppComponent } from './app.component';  
  
const routes: Routes = [  
 { path: 'contact', loadComponent: () => import('./contact-list.component') },  
 { path: 'contact/:id', loadComponent: () => import('./contact-detail.component') }  
];  
  
@NgModule({  
 imports: [BrowserModule, ApmModule, RouterModule.forRoot(routes)],  
 declarations: [AppComponent],  
 providers: [  
 ApmService,  
 { provide: ErrorHandler, useClass: ApmErrorHandler }  
 ],  
 bootstrap: [AppComponent]  
})  
export class AppModule {  
 constructor(apmService: ApmService) {  
 const apm = apmService.init({  
 serviceName: 'angular-app',  
 serverUrl: 'https://<APM\_SERVER>:8200',  
 environment: 'prod'  
 });  
 apm.setUserContext({ username: 'foo', id: 'bar' }); // opcjonalnie  
 }  
}

## 4B) Aplikacja standalone (Angular 15+)

// main.ts  
import { bootstrapApplication } from '@angular/platform-browser';  
import { provideRouter } from '@angular/router';  
import { APP\_INITIALIZER, ErrorHandler } from '@angular/core';  
import { ApmService, ApmErrorHandler } from '@elastic/apm-rum-angular';  
import { AppComponent } from './app/app.component';  
import { routes } from './app/app.routes';  
  
function initApm(apm: ApmService) {  
 return () => apm.init({  
 serviceName: 'angular-app',  
 serverUrl: 'https://<APM\_SERVER>:8200',  
 environment: 'prod'  
 });  
}  
  
bootstrapApplication(AppComponent, {  
 providers: [  
 ApmService,  
 { provide: APP\_INITIALIZER, multi: true, useFactory: initApm, deps: [ApmService] },  
 { provide: ErrorHandler, useClass: ApmErrorHandler },  
 provideRouter(routes)  
 ]  
});

# 5) Kluczowe opcje konfiguracyjne agenta (RUM JS)

* serviceName (wymagane), serverUrl, environment, serviceVersion (dla sourcemaps).
* Distributed tracing: distributedTracingOrigins: ['https://api.twojadomena.pl']; opcjonalnie propagateTracestate: true.
* Sampling/limity: transactionSampleRate (0.0–1.0), eventsLimit.
* Redukcja szumów: ignoreTransactions (np. /assets/), disableInstrumentations (fetch, xmlhttprequest, click, itp.).
* Long tasks: monitorLongtasks: true — zapisuje spany „longtask”.

# 6) Konfiguracja APM Server (RUM, CORS, bezpieczeństwo)

* Włącz RUM i zdefiniuj Allowed Origins na swoje domeny (bez '\*').
* Dla DT Cross-Origin zezwól nagłówki: traceparent, tracestate (preflight OPTIONS).
* Rozważ ograniczenie allow\_service i limity anon. auth; loguj do osobnego policy/ILM jeśli potrzeba.

# 7) Korelacja z backendem (distributed tracing)

1) W RUM ustaw distributedTracingOrigins dla domen API.

2) Backend (API) musi w CORS dodać Access-Control-Allow-Headers: traceparent, tracestate.

3) Backendowe agenty APM muszą obsługiwać W3C Trace Context — większość wspieranych wersji to zapewnia.

# 8) Source maps (zalecane)

* Ustaw serviceVersion w konfiguracji RUM (np. wersja z package.json).
* Zbuduj sourcemapy (np. Webpack devtool: 'source-map').
* Wyślij sourcemapy do Kibany endpointem /api/apm/sourcemaps w trakcie deployu (API key).

# 9) Kibana — gdzie patrzeć i co wnioskować

* APM / Applications → Services → angular-app: latencja page-load/route-change, przepustowość, błędy JS, zależności.
* Traces: pełna ścieżka od RUM do backendu (gdy DT skonfigurowane).
* Long tasks: „freezy” UI (>50 ms) — koreluj z żądaniami HTTP i znacznikami User Timing.

Przykładowe filtry KQL (Discover / Lens na apm-\*):

service.name:"angular-app" and processor.event:"error" # błędy JS

service.name:"angular-app" and processor.event:"transaction" and transaction.type:"route-change" and transaction.duration.us >= 2000000 # trasy SPA >2s

service.name:"angular-app" and processor.event:"span" and span.name : "longtask\*" # long tasks (jeśli włączone)

# 10) Alerty i SLO — przykłady

* Page-load p95 > 3 s przez 5 min (environment: prod).
* Error rate JS > 2% (5–10 min), grupowanie po transaction.name.
* Piki long tasks: liczba/łączny czas na transakcję > próg.

# 11) Dobre praktyki

* Prywatność: nie wysyłaj wrażliwych danych; używaj ignoreTransactions/disableInstrumentations.
* Sampling: zacznij od transactionSampleRate 0.3–0.5; zwiększ na czas debugowania.
* Nazewnictwo: spójne serviceName, environment, serviceVersion (pomaga przy sourcemaps).
* CORS/DT: jawne Allowed Origins i nagłówki traceparent/tracestate po stronie API.

# 12) Troubleshooting

* Brak danych: sprawdź apm-server.rum.enabled: true, Allowed Origins i właściwy serverUrl w init().
* Brak korelacji z backendem: dopisz domeny API do distributedTracingOrigins i nagłówki w CORS.
* Błędy JS nie pojawiają się: skonfiguruj ApmErrorHandler jako globalny ErrorHandler.
* Ad-blocker blokuje intake: użyj serverUrl przez reverse proxy (np. /intake/vX/rum/events).

# 13) Szybki checklist wdrożenia (TL;DR)

* Włącz RUM i CORS (Allowed Origins) w APM Server/Fleet.
* npm i @elastic/apm-rum-angular i wczesna inicjalizacja ApmService + ApmErrorHandler.
* Skonfiguruj distributedTracingOrigins i nagłówki CORS na backendzie.
* Dodaj serviceVersion i upload sourcemaps przy deployu.
* Zweryfikuj w APM → Services i dołóż alerty (p95, error rate, long tasks).