AnalyticsService

September 4, 2026

AnalyticsService

Kind: Service

Source: `atloria-monorepo/apps/api/src/analytics/analytics.service.ts`

`AnalyticsService` is a NestJS backend service responsible for recording user activity and aggregating analytics data for reporting. It tracks page views and search events, then exposes dashboard-oriented metrics such as traffic trends, popular documents, search queries, device usage, sources, and geographic breakdowns.

Methods

Method	Signature	Returns	Description
`trackPageView`	`trackPageView(data: {		

```
projectId: string;
docVersionId?: string;
documentId?: string;
documentSlug: string;
sessionId: string;
referrer?: string;
userAgent?: string;
country?: string;
timeOnPage?: number;
scrollDepth?: number;
fingerprint?: string;
pathname?: string;
timestamp?: string;
type?: string;
```

}, req: any) | unknown | `Persist a page-view event.` | | trackSearch | trackSearch(data: {
projectId: string;
docVersionId?: string;

query: string;
resultsCount: number;
clickedDocId?: string;
sessionId: string;
}) | unknown | | | getOverview | getOverview(projectId: string, days: unknown) | unknown | High-level overview metrics for the analytics dashboard. |
getVersionTraffic	getVersionTraffic(projectId: string, days: unknown)	unknown				getTopDocuments	getTopDocuments(projectId: string, days: unknown, limit: unknown)	unknown			getTopSearchQueries	getTopSearchQueries(projectId: string, days: unknown, limit: unknown)	unknown	
getDailyTraffic	getDailyTraffic(projectId: string, days: unknown)	unknown	Daily page view + unique visitor counts for the time-series chart.
getDeviceBreakdown	getDeviceBreakdown(projectId: string, days: unknown)	unknown	Group page views by device class.
getSourceBreakdown	getSourceBreakdown(projectId: string, days: unknown)	unknown	Group page views by normalized referrer source.
getGeoBreakdown	getGeoBreakdown(projectId: string, days: unknown)	unknown	Group page views by country.
getZeroResultSearches	getZeroResultSearches(projectId: string, days: unknown, limit: unknown)	unknown	Search queries that returned no results.
getGhostDocs	getGhostDocs(projectId: string, days: unknown, limit: unknown)	unknown	Published documents with zero page views in the last N days.
getTrendingDocs	getTrendingDocs(projectId: string, days: unknown, limit: unknown)	unknown	Top documents with week-over-week growth.
getScrollDepthHistogram	getScrollDepthHistogram(projectId: string, days: unknown)	unknown`	Scroll-depth histogram — buckets all page views in the window by the furthest scroll percentage reached.

Dependencies

- `PrismaService`

Where it refuses work

- `AnalyticsService` stops the work with an early return when `isBot(data.userAgent)`.
- `AnalyticsService` stops the work with an early return when `publishedDocs.length === 0`.

When something fails

- `AnalyticsService` handles failure in 2 places: it logs it and continues in all 2.

Diagram

```
sequenceDiagram
    participant Client
    participant Controller
    participant AnalyticsService
    participant AnalyticsStore as Analytics Database

    Client->>Controller: Page view or search request
    Controller->>AnalyticsService: trackPageView() / trackSearch()
    AnalyticsService->>AnalyticsStore: Persist analytics event
    AnalyticsStore-->>AnalyticsService: Event recorded
    AnalyticsService-->>Controller: Tracking result

    Client->>Controller: Request analytics dashboard data
    Controller->>AnalyticsService: getOverview() / breakdown query
    AnalyticsService->>AnalyticsStore: Aggregate event data
    AnalyticsStore-->>AnalyticsService: Metrics and trends
    AnalyticsService-->>Controller: Analytics response
    Controller-->>Client: Dashboard data
```

Usage

```
import { Controller, Get, Query } from '@nestjs/common';
import { AnalyticsService } from './analytics.service';

@Controller('analytics')
export class AnalyticsController {
  constructor(private readonly analyticsService: AnalyticsService) {}

  @Get('overview')
  async getOverview() {
    return this.analyticsService.getOverview();
  }

  @Get('top-documents')
  async getTopDocuments() {
    return this.analyticsService.getTopDocuments();
  }

  @Get('searches')
  async getTopSearchQueries() {
    return this.analyticsService.getTopSearchQueries();
  }
```

```
  @Get('track/search')
  async trackSearch(@Query('query') query: string) {
    return this.analyticsService.trackSearch();
  }
}
```

AI Coding Instructions

- Inject `AnalyticsService` through NestJS dependency injection; do not instantiate it directly.
- Use `trackPageView()` and `trackSearch()` at request/event boundaries so analytics events are captured consistently.
- Use the dedicated aggregation methods (`getDailyTraffic()` , `getDeviceBreakdown()` , `getGeoBreakdown()` , and similar) instead of duplicating reporting queries in controllers.
- Validate and sanitize incoming tracking metadata, especially search terms, referrers, device data, and location-related fields.
- Keep analytics collection non-blocking where possible so tracking failures do not prevent primary user workflows from completing.

Relationships

- DEPENDS_ON → `PrismaService`

Referenced By

- `AnalyticsController` (DEPENDS_ON)
- `AnalyticsModule` (MODULE_PROVIDES)
- `AnalyticsModule` (MODULE_EXPORTS)
- `AnalyticsExportService` (DEPENDS_ON)