AnalyticsExportService

September 4, 2026

AnalyticsExportService

Kind: Service

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

Generates analytics exports in PDF and CSV format.

PDF: built with pdfkit, branded cover page + KPIs grid + top tables.
Returns a Buffer so the controller can stream it as a download or
attach it to an email without writing to disk.

CSV: streams rows directly from Postgres via a Readable wrapper so we
can export 100k+ row datasets without loading them into memory.

`AnalyticsExportService` generates analytics exports as branded PDF reports and high-volume CSV files for download or email delivery. It builds PDFs in-memory using `pdfkit` (cover page, KPI grid, and top tables) and returns a `Buffer` so callers can stream without touching disk. For CSV, it streams rows directly from Postgres through a `Readable` wrapper to handle 100k+ rows without loading results into memory.

Methods

Method	Signature	Returns	Description
`generatePDF`	`generatePDF(opts: {		

```
projectId: string;
period: number;
sections?: ExportSection[];
```

}) | Promise<{ buffer: Buffer; filename: string }> | Build a branded analytics report PDF for a project. | | generateCSV | generateCSV(opts: {
projectId: string;
period: number;

table: 'pageviews' | 'searches';
}) | Promise<{ stream: Readable; filename: string }>` | Stream a CSV export of raw analytics rows. |

Dependencies

- `PrismaService`
- `AnalyticsService`

Where it refuses work

- `AnalyticsExportService` stops the work with an early return when `opts.table === 'pageviews'`.

When something fails

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

Diagram

```
sequenceDiagram
  autonumber
  participant Client
  participant Controller as AnalyticsExportController
  participant Service as AnalyticsExportService
  participant DB as Postgres
  participant PDF as pdfkit
  participant Stream as Readable Stream

  Client->>Controller: Request export (format=pdf|csv, filters)
  Controller->>Service: generateExport(params)

  alt PDF export
    Service->>PDF: Build branded cover + KPI grid + top tables
    PDF-->>Service: Buffer (in-memory)
    Service-->>Controller: Buffer
    Controller-->>Client: Stream Buffer as download / email attachment
  else CSV export
    Service->>DB: Execute query with cursor/streaming
    DB-->>Service: Row stream
    Service->>Stream: Wrap rows into Readable (CSV lines)
    Stream-->>Controller: Stream<CSV>
    Controller-->>Client: Stream CSV download (no buffering)
  end
```

Usage

```
import { Controller, Get, Query, Res } from '@nestjs/common';
import type { Response } from 'express';
import { AnalyticsExportService } from './analytics-export.service';

@Controller('/analytics/exports')
export class AnalyticsExportController {
  constructor(private readonly exports: AnalyticsExportService) {}

  @Get()
  async export(
    @Query('format') format: 'pdf' | 'csv',
    @Query('from') from: string,
    @Query('to') to: string,
    @Res() res: Response,
  ) {
    const params = { from, to }; // add any filters/grouping required by your service

    if (format === 'pdf') {
      const pdfBuffer = await this.exports.generatePdf(params);

      res.setHeader('Content-Type', 'application/pdf');
      res.setHeader('Content-Disposition', `attachment; filename="analytics-${from}-to-${to}.pd
      res.send(pdfBuffer);
      return;
    }

    const csvStream = await this.exports.generateCsvStream(params);

    res.setHeader('Content-Type', 'text/csv; charset=utf-8');
    res.setHeader('Content-Disposition', `attachment; filename="analytics-${from}-to-${to}.csv"

    // Pipe directly to response to avoid buffering large datasets
    csvStream.pipe(res);
  }
}
```

AI Coding Instructions

- Keep PDF generation fully in-memory (`Buffer` output); do not write temporary files—controllers should stream the buffer to HTTP or attach it to email.
- For CSV, preserve true streaming from Postgres through a `Readable`/pipe chain; avoid `.map()`/array accumulation or `await`-ing full result sets.
- Ensure headers and backpressure are handled correctly when piping streams; always set `Content-Type` / `Content-Disposition` before piping to the response.

- When adding new fields/tables to exports, update both the PDF layout (cover/KPI grid/top tables) and CSV column order/escaping consistently; test with large datasets (100k+ rows) for memory safety.
- Integration points: controller for HTTP streaming, email sender for PDF attachments, and DB query layer for cursor/streaming access—changes in query shape can impact both formats.

Relationships

- DEPENDS_ON → `PrismaService`
- DEPENDS_ON → `AnalyticsService`

Referenced By

- `AnalyticsModule` (MODULE_PROVIDES)
- `AnalyticsModule` (MODULE_EXPORTS)
- `AnalyticsExportController` (DEPENDS_ON)
- `ScheduledReportsService` (DEPENDS_ON)