ScheduledReportsService

September 4, 2026

ScheduledReportsService

Kind: Service

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

ScheduledReportsService — CRUD for saved scheduled reports + the cron evaluator that fires due reports every 5 minutes.

The evaluator is intentionally simple: every tick it queries enabled reports, computes their next-run time from `cron` and `lastRunAt`, and fires any that are overdue. Failed reports record their error in `lastStatus` but never crash the loop.

`ScheduledReportsService` manages CRUD operations for saved scheduled reports and evaluates enabled schedules on a five-minute cron tick. For each due report, it calculates the next execution time from its cron expression and `lastRunAt`, triggers the export workflow, and records success or failure status without allowing one failed report to interrupt the evaluator.

Methods

Method	Signature	Returns	Description
`list`	`list(projectId: string)`	`unknown`	
`get`	`get(projectId: string, id: string)`	`unknown`	
`create`	`create(projectId: string, userId: string, dto: CreateScheduledReportDto)`	`unknown`	
`update`	`update(projectId: string, id: string, dto: UpdateScheduledReportDto)`	`unknown`	

Method	Signature	Returns	Description
`remove`	`remove(projectId: string, id: string)`	`unknown`	
`runNow`	`runNow(projectId: string, id: string)`	`unknown`	Manually trigger a saved report.
`evaluateScheduledReports`	`evaluateScheduledReports()`	`unknown`	Runs every 5 minutes.

Dependencies

- `PrismaService`
- `AnalyticsExportService`
- `EmailService`

Where it refuses work

- `ScheduledReportsService` stops the work with `NotFoundException` when `!report` — “Scheduled report not found”.
- `ScheduledReportsService` stops the work with `BadRequestException` when `!Array.isArray(recipients) || recipients.length === 0` — “At least one recipient email is required”.
- `ScheduledReportsService` stops the work with `BadRequestException` when `invalid.length > 0`.
- `ScheduledReportsService` stops the work with an early return when `enabled.length === 0`.
- `ScheduledReportsService` stops the work with an early return when `due.length === 0`.

When something fails

- `ScheduledReportsService` handles failure in 5 places: it logs it and continues in 3, turns it into a return value in 1, and lets it reach the caller in 1.

Diagram

```
sequenceDiagram
  participant Cron as NestJS Cron Scheduler
  participant Service as ScheduledReportsService
```

```
  participant DB as Scheduled Reports Repository
  participant Export as Report Export Service

  Cron->>Service: evaluateDueReports() every 5 minutes
  Service->>DB: Find enabled scheduled reports
  DB-->>Service: Enabled reports

  loop Each enabled report
    Service->>Service: Compute next run from cron + lastRunAt
    alt Report is due
      Service->>Export: Generate and deliver report
      alt Export succeeds
        Export-->>Service: Export result
        Service->>DB: Update lastRunAt and lastStatus
      else Export fails
        Export-->>Service: Error
        Service->>DB: Record failure in lastStatus
      end
    end
  end
```

Usage

```
import { Injectable } from '@nestjs/common';
import { ScheduledReportsService } from './scheduled-reports.service';

@Injectable()
export class AnalyticsAdminService {
  constructor(
    private readonly scheduledReportsService: ScheduledReportsService,
  ) {}

  async createWeeklySalesReport(userId: string) {
    return this.scheduledReportsService.create({
      name: 'Weekly sales summary',
      cron: '0 9 * * 1', // Every Monday at 09:00
      enabled: true,
      reportType: 'sales-summary',
      createdById: userId,
      recipients: ['analytics@example.com'],
      filters: {
        period: 'previous-week',
      },
    });
  }

  async pauseReport(reportId: string) {
    return this.scheduledReportsService.update(reportId, {
      enabled: false,
    });
```

```
  }

  async listActiveReports() {
    return this.scheduledReportsService.findAll({
      enabled: true,
    });
  }
}
```

AI Coding Instructions

- Keep scheduled-report CRUD and execution concerns in this service; delegate report generation and delivery to the existing export/report workflow.
- Treat cron evaluation as fault-isolated: catch and persist errors per report so one failed export never stops the five-minute evaluator.
- When updating execution state, correctly maintain `lastRunAt` and `lastStatus`; these fields determine whether a report is considered overdue on subsequent ticks.
- Validate cron expressions and schedule configuration during create/update operations to prevent invalid schedules from reaching the evaluator.
- Preserve enabled-report filtering in the cron loop; disabled schedules must never be evaluated or executed.

Relationships

- DEPENDS_ON → `PrismaService`
- DEPENDS_ON → `AnalyticsExportService`
- DEPENDS_ON → `EmailService`

Referenced By

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