DashboardService

September 4, 2026

DashboardService

Kind: Service

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

`DashboardService` is a NestJS backend service that aggregates data for dashboard views. It provides overall statistics, project-level metrics, and user activity data for API controllers or other application services.

Methods

Method	Signature	Returns
`getStats`	`getStats(user: JwtPayload)`	`unknown`
`getProjectStats`	`getProjectStats(user: JwtPayload)`	`unknown`
`getUserActivity`	`getUserActivity(user: JwtPayload)`	`unknown`

Dependencies

- `PrismaService`

Diagram

```
sequenceDiagram
    participant Client
    participant Controller as DashboardController
    participant Service as DashboardService
    participant Data as Data Sources

    Client->>Controller: Request dashboard data
    Controller->>Service: getStats()
    Service->>Data: Query aggregate metrics
    Data-->>Service: Statistics data
    Service-->>Controller: Dashboard statistics
```

```
    Controller-->>Client: JSON response

    Client->>Controller: Request project/user insights
    Controller->>Service: getProjectStats() / getUserActivity()
    Service->>Data: Query related records
    Data-->>Service: Project metrics or activity
    Service-->>Controller: Structured dashboard data
    Controller-->>Client: JSON response
```

Usage

```
import { Controller, Get } from '@nestjs/common';
import { DashboardService } from './dashboard.service';

@Controller('dashboard')
export class DashboardController {
  constructor(private readonly dashboardService: DashboardService) {}

  @Get('stats')
  getStats() {
    return this.dashboardService.getStats();
  }

  @Get('projects')
  getProjectStats() {
    return this.dashboardService.getProjectStats();
  }

  @Get('activity')
  getUserActivity() {
    return this.dashboardService.getUserActivity();
  }
}
```

AI Coding Instructions

- Keep dashboard aggregation logic in `DashboardService`; controllers should only handle request routing and response delivery.
- Define explicit return types or DTOs for `getStats`, `getProjectStats`, and `getUserActivity` instead of leaving public method results as `unknown`.
- Ensure data returned by dashboard methods is scoped to the authenticated user or organization where applicable.
- Prefer efficient aggregate queries and avoid per-record queries that can cause N+1 performance issues.

- Update related controllers, DTOs, and tests when adding new dashboard metrics or changing response shapes.

Relationships

- DEPENDS_ON → `PrismaService`

Referenced By

- `DashboardController` (DEPENDS_ON)
- `DashboardModule` (MODULE_PROVIDES)
- `DashboardModule` (MODULE_EXPORTS)