TechnicalDocsGenerationService

September 4, 2026

TechnicalDocsGenerationService

Kind: Service

Source: `atloria-monorepo/apps/api/src/technical-docs/technical-docs-generation.service.ts`

The technical-docs generation UNIT OF WORK (clone → parse → snapshot → materialize).

Extracted from the controller so it runs OFF the API request path: a queue worker (its own
deployment, sized for parsing) calls process(), not the api pod. This is what keeps a large-
repo parse — which can consume several GB — from OOM-ing the pod that serves users. The method
is self-contained and records progress/outcome on the DocumentationJob so it never throws into
the worker; the worker just needs to run it.

`TechnicalDocsGenerationService` executes the technical documentation generation unit of work: clone a repository, parse its source, create a snapshot, and materialize generated documentation. It is invoked by a dedicated queue worker rather than the API request path, isolating memory-intensive repository parsing from user-serving API pods. The service records progress and final outcomes on the associated `DocumentationJob` and handles failures internally so they do not escape to the worker.

Methods

Method	Signature	Returns
`process`	`process(jobId: string	null, options: TechnicalDocsJobPayload)`

Dependencies

- `PrismaService`
- `TechnicalDocsParserService`
- `TechnicalDocsMaterializerService`
- `TechnicalDocsService`
- `OpsAlertService`
- `AIUsageService`
- `NotificationService`
- `EmailService`
- `L10nAutomationService` *(optional)*

Where it refuses work

- `TechnicalDocsGenerationService` stops the work with an early return when `!jobId`.
- `TechnicalDocsGenerationService` stops the work with an early return when `!user`.
- `TechnicalDocsGenerationService` stops the work with an early return when `already`.

When something fails

- `TechnicalDocsGenerationService` handles failure in 4 places: it logs it and continues in 3, and turns it into a return value in 1.

Diagram

```
sequenceDiagram
    participant Worker as Queue Worker
    participant Service as TechnicalDocsGenerationService
    participant Job as DocumentationJob
    participant Git as Git Repository
    participant Parser as Source Parser
    participant Storage as Docs/Snapshot Storage

    Worker->>Service: process(jobId)
    Service->>Job: Mark running / update progress
    Service->>Git: Clone repository
    Git-->>Service: Local repository checkout
    Service->>Parser: Parse source code
    Parser-->>Service: Parsed entities and relationships
    Service->>Storage: Create snapshot
    Storage-->>Service: Snapshot reference
    Service->>Storage: Materialize generated docs
    Service->>Job: Mark completed / save result
    Service-->>Worker: Resolve without throwing
```

```
    Note over Service,Job: On failure, record failed status and error details
```

Usage

```
import { Injectable } from '@nestjs/common';
import { TechnicalDocsGenerationService } from './technical-docs-generation.service';

@Injectable()
export class TechnicalDocsWorker {
  constructor(
    private readonly technicalDocsGenerationService: TechnicalDocsGenerationService,
  ) {}

  async handleGenerationJob(documentationJobId: string): Promise<void> {
    // The service owns progress updates and failure handling.
    // Do not run this work in an API controller request handler.
    await this.technicalDocsGenerationService.process(documentationJobId);
  }
}
```

AI Coding Instructions

- Keep `process()` self-contained: update `DocumentationJob` status and progress throughout every major generation phase.
- Do not let clone, parse, snapshot, or materialization errors escape the service; persist failure details on the job instead.
- Invoke this service only from the dedicated queue worker deployment, never directly from an API request handler.
- Treat repository parsing as memory-intensive work; avoid adding API-pod dependencies or request-scoped behavior.
- Preserve the generation order: clone repository → parse source → create snapshot → materialize documentation.

Relationships

- DEPENDS_ON → `PrismaService`
- DEPENDS_ON → `TechnicalDocsParserService`
- DEPENDS_ON → `TechnicalDocsMaterializerService`
- DEPENDS_ON → `TechnicalDocsService`
- DEPENDS_ON → `OpsAlertService`

- DEPENDS_ON → `AIUsageService`
- DEPENDS_ON → `NotificationService`
- DEPENDS_ON → `EmailService`
- DEPENDS_ON → `L10nAutomationService`

Referenced By

- `TechnicalDocsQueue` (DEPENDS_ON)
- `TechnicalDocsModule` (MODULE_PROVIDES)