ParsingService

September 4, 2026

ParsingService

Kind: Service

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

`ParsingService` manages parsing jobs for the current project, including starting a new parse, monitoring progress, cancelling active work, and listing project-level parsing jobs. It serves as the backend orchestration layer between API consumers and the underlying parsing workflow.

Methods

Method	Signature	Returns	Description
`startParsing`	`startParsing(projectId: string, dto: ParseProjectDto)`	`Promise<ParseStatusDto>`	Start parsing a project
`getParseStatus`	`getParseStatus(projectId: string, jobId: string)`	`Promise<ParseStatusDto>`	Get parse job status
`cancelParsing`	`cancelParsing(projectId: string, jobId: string)`	`Promise<{ success: boolean; message: string }>`	Cancel a parsing job
`getProjectParsingJobs`	`getProjectParsingJobs(projectId: string, limit: number)`	`Promise<ParseStatusDto[]>`	Get all parsing jobs for a project

Dependencies

- `PrismaService`
- `ParsingQueue`

Where it refuses work

- `ParsingService` stops the work with `NotFoundException` when `!job` — “Parse job not found”, in 2 places.

- `ParsingService` stops the work with `NotFoundException` when `!project` — “Project not found”.
- `ParsingService` stops the work with `Error` when `!project.repoUrl` — “No repository URL provided and project has no stored URL”.

Diagram

```
sequenceDiagram
  participant Client
  participant Controller as Parsing Controller
  participant Service as ParsingService
  participant Parser as Parsing Worker/Queue

  Client->>Controller: POST /parsing/start
  Controller->>Service: startParsing()
  Service->>Parser: Create and start parsing job
  Parser-->>Service: ParseStatusDto
  Service-->>Controller: ParseStatusDto
  Controller-->>Client: Job status

  Client->>Controller: GET /parsing/status
  Controller->>Service: getParseStatus()
  Service-->>Controller: ParseStatusDto
  Controller-->>Client: Current status

  Client->>Controller: POST /parsing/cancel
  Controller->>Service: cancelParsing()
  Service->>Parser: Cancel active job
  Service-->>Controller: { success, message }
  Controller-->>Client: Cancellation result
```

Usage

```
import { Injectable } from '@nestjs/common';
import { ParsingService } from './parsing.service';

@Injectable()
export class ProjectSetupService {
  constructor(private readonly parsingService: ParsingService) {}

  async parseProject() {
    const status = await this.parsingService.startParsing();

    if (status.status === 'completed') {
      return status;
    }

    return {
      jobId: status.jobId,
      status: status.status,
```

```
      message: 'Parsing started successfully.',
    };
  }

  async getJobs() {
    return this.parsingService.getProjectParsingJobs();
  }

  async stopParsing() {
    const result = await this.parsingService.cancelParsing();

    if (!result.success) {
      throw new Error(result.message);
    }

    return result;
  }
}
```

AI Coding Instructions

- Inject `ParsingService` through NestJS dependency injection; do not instantiate it directly.
- Use `startParsing()` to create a parsing job, then query `getParseStatus()` or `getProjectParsingJobs()` for progress and history.
- Treat parsing as asynchronous work: do not assume `startParsing()` means parsing has completed.
- Call `cancelParsing()` only for an active job and always handle the returned `success` flag and message.
- Preserve `ParseStatusDto` as the status contract across controllers, clients, and any queue or worker integrations.

Relationships

- DEPENDS_ON → `PrismaService`
- DEPENDS_ON → `ParsingQueue`

Referenced By

- `ParsingController` (DEPENDS_ON)
- `ParsingModule` (MODULE_PROVIDES)
- `ParsingModule` (MODULE_EXPORTS)