ProjectService

September 4, 2026

ProjectService

Kind: Service

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

`ProjectService` encapsulates project lifecycle and membership management for the API. It provides operations for creating, retrieving, updating, and deleting projects, as well as managing project members, roles, and project-level settings.

Methods

Method	Signature	Returns	Description
`create`	`create(dto: CreateProjectDto, user: JwtPayload)`	`Promise<Project>`	
`list`	`list(user: JwtPayload)`	`Promise<Project[]>`	
`get`	`get(id: string, user: JwtPayload)`	`Promise<Project>`	
`update`	`update(id: string, dto: UpdateProjectDto, user: JwtPayload)`	`Promise<Project>`	
`delete`	`delete(id: string, user: JwtPayload)`	`Promise<void>`	
`getMembers`	`getMembers(id: string, user: JwtPayload)`	`Promise<ProjectMember[]>`	
`addMember`	`addMember(id: string, dto: AddMemberDto, user: JwtPayload)`	`Promise<ProjectMember>`	
`updateMemberRole`	`updateMemberRole(id: string, userId: string, dto: UpdateMemberRoleDto, user: JwtPayload)`	`Promise<ProjectMember>`	
`removeMember`	`removeMember(id: string, userId: string, user: JwtPayload)`	`Promise<void>`	
`getSettings`	`getSettings(id: string, user: JwtPayload)`	`Promise<Record<string, any>>`	
`updateSettings`	`updateSettings(id: string, dto: UpdateSettingsDto, user: JwtPayload)`	`Promise<Project>`	

Method	Signature	Returns	Description
`getStats`	`getStats(id: string, user: JwtPayload)`	`unknown`	
`docsHealth`	`docsHealth(id: string, user: JwtPayload)`	`unknown`	Consolidated "Docs Health" summary for the owner project page (P1.4).
`publish`	`publish(id: string, user: JwtPayload)`	`Promise<PublishedProjectResult>`	Publish a project making it publicly accessible
`resolveAvailableSlug`	`resolveAvailableSlug(desired: string, selfId: string)`	`Promise<string>`	Normalize a desired subdomain slug and guarantee it's available: reserved- word safe + globally unique across projects (excluding `selfId`).
`checkSlugAvailability`	`checkSlugAvailability(id: string, desired: string, user: JwtPayload)`	`unknown`	Is a desired subdomain slug available for this project?
`setPublishedSlug`	`setPublishedSlug(id: string, desired: string, user: JwtPayload)`	`unknown`	Set/rename the project's subdomain slug (the {slug}.atloria.app host).
`unpublish`	`unpublish(id: string, user: JwtPayload)`	`Promise<Project>`	Unpublish a project making it private
`getPublishedByUrlId`	`getPublishedByUrlId(urlId: string)`	`Promise<Project>`	Get a published project by URL ID (public access, no auth required)
`getPublishedBySlug`	`getPublishedBySlug(publishedSlug: string)`	`Promise<Project>`	Get a published project by its subdomain slug (public access, no auth required) Used by *.atloria.app

Method	Signature	Returns	Description
			subdomain routing
`getProjectAudiences`	`getProjectAudiences(id: string, docVersionId: string	string[])`	`unknown`
`getAudienceDocuments`	`getAudienceDocuments(projectId: string, audienceSlug: string, docVersionId: string, hiddenDocIds: Set	null, lang: string)`	`unknown`
`getPublishedDocument`	`getPublishedDocument(projectId: string, audienceSlug: string, documentSlug: string, docVersionId: string, hiddenDocIds: Set	null, lang: string)`	`Promise<{

```
id: string;
title: string;
slug: string;
content: string;
toc: TocEntry[];
order: number | null;
categoryId: string | null;
pageType: string | null;
updatedAt: Date;
category: {
  id: string;
  name: string;
  slug: string;
  icon: string | null;
  color: string | null;
} | null;
/**
 * Set when the underlying screen/code-entity this page documented was removed in a
 * later delta (metadata.screenRemoved / techEntityRemoved). The page is retained
 * during the true-up grace window so links don't 404; the reader shows a
 * "no longer part of the product" banner instead of pretending it's current.
 */
removed: { at: string | null } | null;
/**
 * D3 viewport realism: desktop-blob-URL → its responsive/zoom companion set
 * ({ desktop, tablet?, mobile?, crops?[] }), so the reader can offer a
 * viewport picker on images that have variants. Only these public CDN URLs
 * leave the service — never the rest of the internal metadata blob.
 */
imageVariants: Record<string, unknown> | null;
/** B6: the language the page body is actually served in. */
language: string;
/** B6: set when a published overlay variant is being served instead of the source body. */
translation: { language: string; machineTranslated: boolean; stale: boolean } | null;
/** B6: set when the reader asked for a language this page has no variant in (rung 3). */
untranslated: { requestedLanguage: string } | null;
```

}> `| Get a specific published document by slug for a project and audience Used for Level 3 document viewing in public project pages |` | renderPreview | renderPreview(audienceSlug: string, rawContent: string) | { content: string; toc: TocEntry[] } `| B2 preview: render RAW proposed markdown through the exact same audience-`

`filter + ToC path a published read uses (getPublishedDocument above), so a change-re… |`
| getRelatedDocuments | getRelatedDocuments(projectId: string, audienceSlug: string, documentSlug: string, docVersionId: string, limit: unknown, hiddenDocIds: Set | null) | Promise< Array<{
id: string;
slug: string;
title: string;
category: { name: string; slug: string } | null;
}>

> `| Related / "See also" pages for a published document. |`
> | resolvePublicDocVersionId | resolvePublicDocVersionId(projectId: string, opts: { version?: string; lang?: string; audienceScope?: string }) | Promise<string | null> | `Resolve the doc-set version served for public reads. |` | recentChanges | recentChanges(projectId: string, sinceIso: string | undefined, lang: string, hiddenDocIds: Set | null) | Promise<{ count: number; since: string | null; pages: Array<{ slug: string; title: string; updatedAt: Date }> }> | `Resolve the currently-served version of EACH audience scope (user manual + technical set). |`
> | getActiveScopeDocVersionIds | getActiveScopeDocVersionIds(projectId: string, lang: string) | Promise<string[]> | | | resolveVersion | resolveVersion(projectId: string, versionNumber: number) | unknown | `Phase 3: Resolve a version number to a DocVersion record for public access. |` | getPublishedVersions | getPublishedVersions(projectId: string) | unknown | `Phase 3: Get all publicly accessible versions for the version switcher. |`
> | getPublishedLanguages | getPublishedLanguages(projectId: string) | unknown` | Languages available for the public locale switcher. |

Dependencies

- `PrismaService`
- `UrlService`
- `TocBuilderService`
- `TechDocsRagService`
- `AuditService` *(optional)*
- `SnippetsService` *(optional)*

Where it refuses work

- `ProjectService` stops the work with `NotFoundException` when `!member` — “Project member not found”, in 2 places.
- `ProjectService` stops the work with `NotFoundException` when `!project` — “Project not found”, in 2 places.
- `ProjectService` stops the work with `BadRequestException` when `!validation.valid`.
- `ProjectService` stops the work with `ForbiddenException` when `!allowedRoles.includes(user.role as UserRole)` — “Insufficient permissions to create projects”.

- `ProjectService` stops the work with `ConflictException` when `existingProject` — “A project with this slug already exists in your organization”.
- `ProjectService` stops the work with `NotFoundException` when `!_project` — “Project not found”.

When something fails

- `ProjectService` handles failure in 1 place: it discards it silently in all 1.

Diagram

```
sequenceDiagram
    participant Client
    participant Controller as Project Controller
    participant Service as ProjectService
    participant Database

    Client->>Controller: Create or manage project request
    Controller->>Service: create/list/get/update/delete(...)
    Service->>Database: Read or write project data
    Database-->>Service: Project result
    Service-->>Controller: Project response
    Controller-->>Client: HTTP response

    Client->>Controller: Manage project member
    Controller->>Service: addMember/updateMemberRole/removeMember(...)
    Service->>Database: Update membership records
    Database-->>Service: Member result
    Service-->>Controller: Member response
    Controller-->>Client: HTTP response
```

Usage

```
import { Injectable } from '@nestjs/common';
import { ProjectService } from './project.service';

@Injectable()
export class ProjectController {
  constructor(private readonly projectService: ProjectService) {}

  async createProject() {
    const project = await this.projectService.create();

    return project;
  }

  async getProjectMembers() {
    const members = await this.projectService.getMembers();

    return members;
  }

  async addProjectMember() {
    const member = await this.projectService.addMember();

    return member;
```

```
  }

  async changeMemberRole() {
    return this.projectService.updateMemberRole();
  }

  async removeProjectMember(): Promise<void> {
    await this.projectService.removeMember();
  }

  async getProjectSettings() {
    return this.projectService.getSettings();
  }
}
```

AI Coding Instructions

- Keep project, membership, and settings operations within `ProjectService`; controllers should delegate business logic to this service.
- Preserve the existing return contracts: project methods return `Project`, membership methods return `ProjectMember`, and removal methods resolve to `void`.
- Apply authorization and tenant/project access validation before exposing project data or modifying members.
- When changing member roles, ensure role transitions and membership uniqueness are validated consistently with the application's authorization model.
- Update related controllers, DTOs, persistence models, and tests when adding new project fields, settings, or membership behaviors.

Relationships

- DEPENDS_ON → `PrismaService`
- DEPENDS_ON → `UrlService`
- DEPENDS_ON → `TocBuilderService`
- DEPENDS_ON → `TechDocsRagService`
- DEPENDS_ON → `AuditService`
- DEPENDS_ON → `SnippetsService`

Referenced By

- `ProjectController` (DEPENDS_ON)
- `ProjectModule` (MODULE_PROVIDES)
- `ProjectModule` (MODULE_EXPORTS)
- `PublicProjectController` (DEPENDS_ON)
- `TrialService` (DEPENDS_ON)