ProjectOrgGuard

September 4, 2026

ProjectOrgGuard

Kind: Service

Source: `atloria-monorepo/apps/api/src/auth/guards/project-org.guard.ts`

ProjectOrgGuard — enforce that the authenticated user's organization OWNS the project named
by `:projectId`. OrganizationGuard only checks an explicit `organizationId` in the request,
so a project-scoped route (which carries only a `projectId`) sails past it — a user from org
A could act on org B's project. This guard closes that hole: it loads the project, 404s when
it does not exist, and 403s unless project.organizationId === user.organizationId.

Fail-closed: an unauthenticated request or a missing project context is rejected, never allowed through.

`ProjectOrgGuard` is a NestJS authorization guard for project-scoped routes. It loads the project identified by `:projectId`, returns `404` when the project does not exist, and rejects requests with `403` unless the authenticated user belongs to the project's owning organization. It fails closed when authentication or project context is missing.

Methods

Method	Signature	Returns
`canActivate`	`canActivate(context: ExecutionContext)`	`Promise<boolean>`

Dependencies

- `PrismaService`

Where it refuses work

- `ProjectOrgGuard` stops the work with `ForbiddenException` when `!user` — “User not authenticated”.
- `ProjectOrgGuard` stops the work with `ForbiddenException` when `!projectId` — “Project context required”.
- `ProjectOrgGuard` stops the work with `NotFoundException` when `!project`.
- `ProjectOrgGuard` stops the work with `ForbiddenException` when `project.organizationId !== user.organizationId` — “Access denied: this project belongs to another organization”.

Diagram

```
sequenceDiagram
  participant Client
  participant Guard as ProjectOrgGuard
  participant ProjectService
  participant Controller

  Client->>Guard: Request with :projectId and authenticated user
  Guard->>Guard: Read user.organizationId and params.projectId

  alt Missing user, organization, or projectId
    Guard-->>Client: Reject request
  else Request context is valid
    Guard->>ProjectService: Find project by projectId

    alt Project does not exist
      ProjectService-->>Guard: null
      Guard-->>Client: 404 Not Found
    else Project exists
      ProjectService-->>Guard: project.organizationId

      alt Organization IDs do not match
        Guard-->>Client: 403 Forbidden
      else Organization IDs match
        Guard->>Controller: Allow route handler
        Controller-->>Client: Response
      end
    end
  end
```

Usage

```
import { Controller, Delete, Param, UseGuards } from '@nestjs/common';
import { JwtAuthGuard } from '../auth/guards/jwt-auth.guard';
import { ProjectOrgGuard } from '../auth/guards/project-org.guard';
import { ProjectsService } from './projects.service';

@Controller('projects')
@UseGuards(JwtAuthGuard, ProjectOrgGuard)
export class ProjectsController {
  constructor(private readonly projectsService: ProjectsService) {}

  @Delete(':projectId')
  async removeProject(@Param('projectId') projectId: string) {
    return this.projectsService.remove(projectId);
  }
}
```

`ProjectOrgGuard` expects the authentication guard to populate `request.user`, including the authenticated user's `organizationId`.

AI Coding Instructions

- Apply `ProjectOrgGuard` to every route scoped by `:projectId`, especially mutation endpoints such as update, delete, member management, and project settings.
- Ensure an authentication guard runs before this guard and attaches a user with a valid `organizationId` to `request.user`.
- Preserve fail-closed behavior: missing authentication, missing organization context, or missing `projectId` must reject the request rather than allow it.
- Keep project lookup behavior consistent: nonexistent projects should produce `404 Not Found`, while organization ownership mismatches should produce `403 Forbidden`.
- Do not rely on `OrganizationGuard` alone for project routes; project ownership must be resolved from the stored project record.

Relationships

- DEPENDS_ON → `PrismaService`