ReconcilerService

September 4, 2026

ReconcilerService

Kind: Service

Source: `atloria-monorepo/apps/api/src/docs-repo/reconciler.service.ts`

ReconcilerService — the DB-wins rebuild that makes a project's docs repo DISPOSABLE.

Postgres is the authority; the repo is a derived, rebuildable projection of it. `rebuild` loads the CURRENTLY-SERVED doc set (the same resolution public serving uses — the ACTIVE
version per audience scope, else the latest PUBLISHED) and writes every published+public
page to docs///.md, then commits as atloria-generator. `driftCheck`
proves the invariant by diffing the repo tree against a fresh render.

Version resolution mirrors ProjectService.resolvePublicDocVersionId (default path). It is
replicated here against Prisma rather than importing ProjectModule so the substrate stays
a leaf with no module cycle; the query is intentionally identical.

`ReconcilerService` rebuilds a project's documentation repository from Postgres, treating the database as the source of truth and the Git repository as a disposable derived projection. It renders the currently served public documentation set into `docs/<scope>/<lang>/<slug>.md`, commits generated changes as `atloria-generator`, and can verify repository drift against a fresh database render. Version selection mirrors the public-serving resolution path: active versions for an audience scope, otherwise the latest published version.

Methods

Method	Signature	Returns	Description
`rebuild`	`rebuild(projectId: string, opts: RebuildOptions)`	`Promise<RebuildResult>`	Rebuild the whole repo from the served DB doc set.
`backfillVersion`	`backfillVersion(projectId: string, versionId: string)`	`Promise<{ tag: string; commit: string } | null>`	Back-fill EVERY un-tagged PUBLISHED (rollback-worthy) version of a project.
`backfillProject`	`backfillProject(projectId: string)`	`Promise<{ backfilled: number; skipped: number; total: number }>`	Back-fill EVERY un-tagged PUBLISHED (rollback-worthy) version of a project.
`driftCheck`	`driftCheck(projectId: string)`	`Promise<DriftResult>`	Compare the repo's committed tree (under docs/) against a fresh DB render.

Dependencies

- `PrismaService`
- `DocsRepoService`

Where it refuses work

- `ReconcilerService` stops the work with an early return when `!version`.
- `ReconcilerService` stops the work with an early return when `version.gitTag`.
- `ReconcilerService` stops the work with an early return when `docs.length === 0`.
- `ReconcilerService` stops the work with an early return when `headVersion !== null && headVersion !== current`.
- `ReconcilerService` stops the work with an early return when `!Array.isArray(m?.sourceScreenSlugs)`.
- `ReconcilerService` stops the work with an early return when `!Array.isArray(toc)`.

When something fails

- `ReconcilerService` handles failure in 1 place: it logs it and continues in all 1.

Diagram

```
sequenceDiagram
  participant Caller as Admin/Job Caller
  participant Reconciler as ReconcilerService
  participant DB as Postgres (Prisma)
  participant Git as Docs Git Repository

  Caller->>Reconciler: rebuild(projectId)
  Reconciler->>DB: Resolve currently served doc versions
  DB-->>Reconciler: Active or latest published versions

  Reconciler->>DB: Load published + public pages
  DB-->>Reconciler: Pages and content
  Reconciler->>Git: Write docs/<scope>/<lang>/<slug>.md
  Reconciler->>Git: Commit generated tree as atloria-generator
  Git-->>Reconciler: Commit SHA
  Reconciler-->>Caller: RebuildResult

  Caller->>Reconciler: driftCheck(projectId)
  Reconciler->>DB: Render fresh expected docs tree
  Reconciler->>Git: Read current repository tree
  Reconciler-->>Caller: DriftResult
```

Usage

```
import { ReconcilerService } from './reconciler.service';

@Injectable()
export class DocsMaintenanceService {
  constructor(private readonly reconciler: ReconcilerService) {}

  async rebuildProjectDocs(projectId: string) {
    const result = await this.reconciler.rebuild(projectId);

    return {
      commit: result.commit,
      writtenFiles: result.written,
    };
  }

  async verifyProjectDocs(projectId: string) {
    const drift = await this.reconciler.driftCheck(projectId);
```

```
    if (drift.hasDrift) {
      // Schedule or trigger a rebuild to restore the generated projection.
      await this.reconciler.rebuild(projectId);
    }

    return drift;
  }

  async backfillVersion(projectId: string, versionId: string) {
    const result = await this.reconciler.backfillVersion(projectId, versionId);

    return result
      ? `Backfilled ${result.tag} at commit ${result.commit}`
      : 'No backfill was required.';
  }
}
```

AI Coding Instructions

- Treat Postgres as authoritative: never use repository contents to determine document state, visibility, or version resolution.
- Keep version-resolution logic aligned with `ProjectService.resolvePublicDocVersionId`; this service intentionally duplicates the Prisma query to avoid a module dependency cycle.
- Only render pages that are both published and public, using the canonical path format: `docs/<scope>/<lang>/<slug>.md`.
- Generated repository commits must use the `atloria-generator` identity so automated reconciliation changes remain distinguishable from user edits.
- Use `driftCheck()` before debugging repository inconsistencies; it compares the current Git tree with a freshly rendered expected tree.

Relationships

- DEPENDS_ON → `PrismaService`
- DEPENDS_ON → `DocsRepoService`

Referenced By

- `DocsRepoModule` (MODULE_PROVIDES)
- `DocsRepoModule` (MODULE_EXPORTS)