NestJS with Prisma Project Structure

NestJS integrated with Prisma ORM for type-safe database queries, auto-generated types, and seamless migrations.

#nestjs #typescript #prisma #database #orm #postgresql

PNG PDF Copy Prompt

Project Directory

```
myproject/
├── src/ Application sou…
│   ├── main.ts Bootstrap app
│   ├── app.module.ts Root module
│   ├── prisma/ Prisma module
│   │   ├── prisma.module.ts Global module
│   │   └── prisma.service.ts Extends PrismaC…
│   ├── config/
│   │   ├── configuration.ts
│   │   └── database.config.ts
│   ├── common/
│   │   ├── filters/
│   │   │   └── prisma-exception.filter.ts Handle DB errors
│   │   └── interceptors/
│   │       └── transform.interceptor.ts
│   ├── modules/ Feature modules
│   │   ├── users/
│   │   │   ├── users.module.ts
│   │   │   ├── users.controller.ts
│   │   │   ├── users.service.ts Injects PrismaS…
│   │   │   └── dto/
│   │   │       ├── create-user.dto.ts
│   │   │       └── update-user.dto.ts
│   │   └── posts/
│   │       ├── posts.module.ts
│   │       ├── posts.controller.ts
│   │       ├── posts.service.ts
│   │       └── dto/
│   │           └── create-post.dto.ts
│   └── prisma/ Prisma schema &…
│       ├── schema.prisma Database schema
│       ├── seed.ts Seed data script
│       └── migrations/ Migration histo…
│           └── .gitkeep
├── test/
│   ├── app.e2e-spec.ts
│   └── jest-e2e.json
├── nest-cli.json
├── tsconfig.json
├── tsconfig.build.json
├── package.json
├── .env.example DATABASE_URL he…
├── .eslintrc.js
├── .prettierrc
├── .gitignore
└── README.md
```

Why This Structure?

Prisma generates TypeScript types from your schema—no manual entity definitions. The `PrismaService` wraps `PrismaClient` for proper NestJS lifecycle handling. A global `PrismaModule` makes it injectable anywhere.

Key Directories

`prisma/schema.prisma` - Single source of truth for database models

`prisma/migrations/` - Version-controlled database changes

`src/prisma/` - NestJS module wrapping PrismaClient

`src/modules/*/` - Features inject PrismaService directly

Getting Started

1. `npm install`
2. `cp .env.example .env` (set DATABASE_URL)
3. `npx prisma migrate dev`
4. `npx prisma db seed`
5. `npm run start:dev`

Best Practices

- Define relations in `schema.prisma`, not service code
- Use `@Transaction()` decorator for multi-table writes
- Run `prisma generate` in CI after migrations
- Keep `PrismaService` stateless—no business logic
- Use Prisma's `select` and `include` to avoid over-fetching

Trade-offs

`No repository layer` - Services use Prisma directly—simpler but less abstract

`Schema lock-in` - Prisma schema syntax, not raw SQL migrations

`Query complexity` - Complex joins may need raw SQL via `$queryRaw`