Fastify + TypeScript – Template Modulare

Questa guida descrive l’architettura, i plugin, la struttura dei file e l’uso del template Fastify generato.

1. Architettura Modulare

Il progetto è organizzato per moduli. Ogni feature è incapsulata in una cartella (`modules/`), separando controller, routes e index. I plugin comuni (es. JWT, Swagger) sono in `src/plugins`. L'app viene avviata da `src/server.ts`, che istanzia Fastify e carica `app.ts`.

```
Esempio struttura:
src/ ■■■ server.ts → Entry point ■■■ app.ts → Plugin + autoload ■■■ plugins/ → JWT, Swagger ■ ■■■ jwt.ts ■ ■■■ swagger.ts ■■■ modules/ ■ ■■■ user/ ■ ■■■ controller.ts ■ ■■■ routes.ts ■ ■■■ index.ts
```

2. Plugin Inclusi

- @fastify/jwt: per autenticazione stateless. - @fastify/swagger + swagger-ui: documentazione OpenAPI. - @fastify/autoload: carica automaticamente moduli route/plugin. - dotenv: gestisce variabili da file `.env`.

3. Avvio del progetto

1. Rinomina `.env.example` in `.env`. 2. Installa le dipendenze con `npm install`. 3. Avvia il server in dev: `npm run dev`

4. Rotte di esempio

Il modulo `user` espone: - POST /users → Crea nuovo utente - GET /users → Lista utenti (protetta da JWT) Autenticazione con header: Authorization: Bearer

5. Suggerimenti di Estensione

- Aggiungi Prisma per connettere PostgreSQL. - Configura test con Vitest + Supertest. - Definisci schema Zod per l’input API. - Aggiungi Dockerfile + docker-compose.