🧱 Microservizi Base - Schema e Struttura Modulare

Questo schema descrive un'architettura **monorepo** in TypeScript con Fastify, pensata per riutilizzare microservizi fondamentali in più progetti (CRM, SaaS, AI tools, dashboard, ecc.).

💣 Struttura Progetto (es. con `pnpm` o `turborepo`)

```
microservices-monorepo/
├── packages/
│   ├── core/                  # Plugin condivisi, logger, env, error
handler
│   ├── sdk/                   # Client REST o gRPC per i microservizi
├── services/
│   ├── auth-service/          # Login, JWT, ruoli
│   ├── user-service/          # Profili, team, preferenze
│   ├── file-service/          # Upload/download file (S3, locale)
│   ├── notification-service/  # Email, push, webhook
│   ├── queue-service/         # BullMQ, Redis, job scheduler
│   ├── permission-service/    # RBAC, ABAC, access policy
├── gateway/                   # Fastify API Gateway centralizzato
├── .env                       # Variabili condivise (con supporto per
override)
└── README.md
```

🔌 Descrizione Servizi

🔐 `auth-service`

- Registrazione, login
- Refresh token
- JWT con Fastify + Zod
- Storage utenti base (DB)

`user-service`

- CRUD utenti, ruoli, team
- Avatar, profili pubblici/privati

📁 `file-service`

- Upload con firma JWT
- Salvataggio file locale o S3
- API: upload, download, delete

🔔 `notification-service`

- Email async via job queue
- Webhook o Firebase (push)
- Template HTML/email support

📩 `queue-service`

- Job asincroni via BullMQ o Fastify Cron
- Retry, scheduling, logs job

🔐 `permission-service`

- Gestione ruoli, policy
- ACL o ABAC
- Middleware verifiche per Fastify Gateway

🚪 Gateway (API centrale)

- Fastify con `@fastify/http-proxy` o router diretto
- Middleware globale (auth, logging)
- Swagger + logging centralizzato

🧰 Pacchetti condivisi (`packages/`)

- `core`: logger, validatori, errori, middleware comuni
- `sdk`: client REST per ogni servizio (es. `AuthClient`, `UserClient`)

🛠️ Roadmap (moduli futuri)

-

📦 Come usarli

1. Ogni microservizio può vivere **standalone** o **collegato**
2. L'API Gateway li espone unificati
3. I client (`sdk/`) ti permettono di usarli da frontend o altri backend

---

🕑 Obiettivo finale

Costruire una **base condivisa e scalabile** per:

- SaaS multi-tenant

• agenti AI orchestrati
• CRM, dashboard, gestionali

Questa struttura ti permette di riusare lo stesso core per 5+ progetti diversi. Aggiungi moduli solo quando servono!